How to make a simple shellcode (The basics)

Shell-code is a piece of object codes that can be injected into the executable stack to get the execution access...Shell-code is so called because it is basically used to get a shell (/bin/bash).. We'll see how make a simple exit shell-code..

This article assumes basic knowledge of Assembly x86 as prerequisites for this article

Shell-Codding

First lets just type the basic exit routine in assembly (x86 , intel format)..

shell.asm

Code:

```
section .text

global _start

_start :

        mov eax,1

        mov ebx,7

        int 0x80
```

Assembling and linking

Code:

```
aneesh@aneesh-laptop:~/articles/ASM$ nasm -f elf32  shell.asm -o shell.o

aneesh@aneesh-laptop:~/articles/ASM$ ld shell.o -o shell
```

Lets run the code and check the exit status … so that we know that it runs without errors..

Code:

```
aneesh@aneesh-laptop:~/articles/ASM$ ./shell

aneesh@aneesh-laptop:~/articles/ASM$ echo $?

7
```

Perfect!!

Now lets dump the shellcode's opcodes that are of main concern to us as we need opcodes to attach the shell code to the executable stack..

lets dump the code with objdump

Code:

```
aneesh@aneesh-laptop:~/articles/ASM$ objdump -d shell

shell:     file format elf32-i386

Disassembly of section .text:

08048060 <_start>:

 8048060:       b8 01 00 00 00          mov    $0x1,%eax

 8048065:       bb 07 00 00 00          mov    $0x7,%ebx

 804806a:       cd 80                   int    $0x80
```

Just run that command for now.. I'll write a tutorial on objdump soon!!...

Now as we see there are lots and lots of nuls out there in the opcodes..

So we need to remove that because as we will be using this shellcode to run it in a executable stack so..The program will be reading the opcodes only till it finds a null (assume the functionallity like that of strcpy()).. As it finds a null it will return to the main program..

So our shell-code will not work..

Now lets try to remove the nulls...

New shell.asm

Code:

```
section .text

global _start

_start :

        xor eax,eax ; zero the eax

        mov al,1
     ; move 1 to lower bit of eax which is 1 byte so we'll loose the null
        xor ebx,ebx
 ;
        mov bl,7
     ;
        int 0x80
      ; call the kernel..
```

Assembling and linking :-

Code:

```
aneesh@aneesh-laptop:~/articles/ASM$ nasm -f elf32  shell.asm -o shell.o
```

```
aneesh@aneesh-laptop:~/articles/ASM$ ld shell.o -o shell
```

Testing

Code:

```
aneesh@aneesh-laptop:~/articles/ASM$ ./shell

aneesh@aneesh-laptop:~/articles/ASM$ echo $?

7
```

Dump the opcodes

Code:

```
aneesh@aneesh-laptop:~/articles/ASM$ objdump -d shell

shell:     file format elf32-i386

Disassembly of section .text:

08048060 <_start>:

 8048060:       31 c0                   xor    %eax,%eax

 8048062:       b0 01                   mov    $0x1,%al

 8048064:       31 db                   xor    %ebx,%ebx

 8048066:       b3 07                   mov    $0x7,%bl

 8048068:       cd 80                   int    $0x80
```

Yupi... We eliminated all the NULL's..This makes the Shell-Code Complete..

Source: http://www.go4expert.com/articles/simple-shellcode-basics-t24907/