A taste of our own medicine : How SmokeLoader is deceiving configuration extraction by using binary code as bait

int0xcc.svbtle.com/a-taste-of-our-own-medicine-how-smokeloader-is-deceiving-dynamic-configuration-extraction-by-using-binary-code-as-bait

September 18, 2018
A taste of our own medicine : How smokeloader is deceiving dynamic configuration extraction by using binary code as bait

Recently an interesting smoke loader sample caught my eye ,and moreover I had to put smoke loader monitoring under scrutiny , as my monitoring script found it hard to locate a live c2 . Then suddenly something strange I noticed on the dashboard , the output c2's from the configuration extraction script and the generated pcap were different

URL	http://185.35.137.147/mlp/

Output From config extraction

Port	HTTP Host	Method	URI
80	www.msftncsi.com	GET	/ncsi.txt
80	185.61.148.224	POST	/p/
80	www.msftncsi.com	GET	/ncsi.txt

Pcap generated output

Notice the subtle difference between two outputs ?

A configuration extraction script is essentially an instrumenting script (using windbg or a memory acquisition tool) to extract configuration (c2's , keys , campaigns, etc) from a running malware binary . It's sole purpose is to capture a pattern in a binary to extract certain parameters like DWORD's , constants or pointers to memory region . Generally there is a long sleep call between consecutive attempts to connects multiple c2's , which is essentially a way though which it keeps its secondary c2's hidden , as mostly only one of the few c2's gets listed in a sandbox report .
The smoke loader configuration happens to be a list of c2's and encryption keys (DWORD)

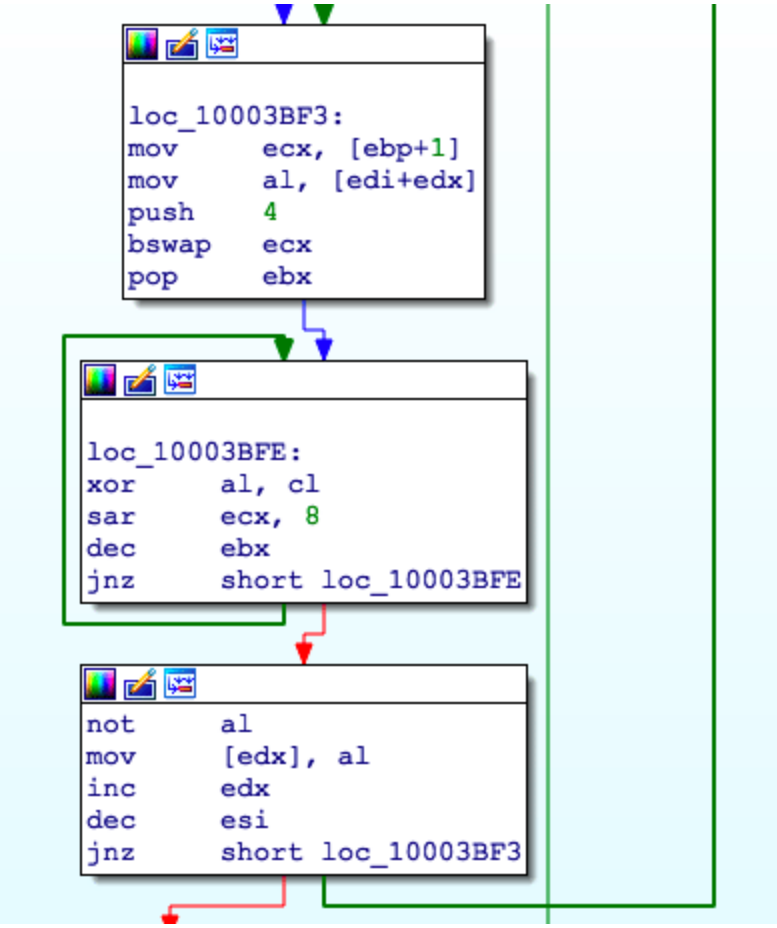
```
GenerateC2 proc near
mov     ecx, dword ptr NumC2
xor     eax, eax
cmp     ecx, 2
cmovz   ecx, eax
mov     dword ptr NumC2, ecx
mov     ecx, C2BufferArray[ecx*4]
jmp     DecodeC2
GenerateC2 endp
```

This subroutine that generates a hidden c2, roughly translates to following stream in opcode .

```
33 C0 83 F9 02 0F 44 C8 89 0D 80 6C 00 10 8B 0C 8D E8 12 00 10
```

Extracting Numc2 and C2BufferArray (encoded c2 list buffer) would be a matter of creating a regex

```
RegEx = \x33\xC0\x83\xF9(.)\x0F\x44\xC8\x89\x0D.{4}\x8B\x0C\x8D(.{4})
```



But unpacking a particular sample mentioned earlier , revealed another side of the story . Although the code to load encoded c2 buffer was there , but the coding routine was a clever choice of deception, which feeds a fake encoded c2 buffer , though decoded buffer is a valid http resource , but instead chooses to take the c2 buffer from a plain text value in between the subroutine

```
DecodeC2buf     proc near               ; CODE XREF: Installer+154↑p
                                        ; InjectPayload+63↑p ...
                call    $+5
                pop     eax
                add     eax, 12h
                mov     ecx, eax
                nop
                nop
                jmp     short locret_10003C41
; ---------------------------------------------------------------------------
                align 10h
                dd 0
                db    0
                align 2
aHttp1856114822 db 'http://185.61.148.224/p/',0
                align 10h
                dd 7 dup(0)
                db 5 dup(90h)
; ---------------------------------------------------------------------------

locret_10003C41:                        ; CODE XREF: DecodeC2buf+D↑j
                retn
DecodeC2buf     endp
```

But the fact to notice is , not only it would fools scripts , but difference between the real and the fake c2 is so subtle , that it deceives the eyes of the beholder as well.

SmokeLoader has suffered considerably a lot due to immediate c2 takedown , its no surprise that they were looking for a quick and a smart way to tackle this problem , but seldom it goes unnoticed

17

Kudos

17

Kudos