MRLAB

Windows ToolHelp32 kitabxanasından yayınma 2. hissə - Proses yolunun gizlədilməsi

Azərbaycan Respublikası Xüsusi Rabitə və İnformasiya Təhlükəsizliyi Dövlət Xidməti - Kompüter İnsidentlərinə qarşı Mübarizə Mərkəzi - Malware Research Lab - S. Abasov - 17 Aprel 2022

TlHelp32 icra oluna bilən proqram təminatları haqqında informasiya əldə edilməsi üçün istifadə edilən funksiyalardan ibarət kitabxanadır. Bu kitabxana köməkliyi ilə sistemdə fəaliyyət göstərən proseslər, onları istifadə etdikləri modullar (dll), aktiv heap yaddaşları, icra olunan threadlar vs haqqında məlumat əldə etmək mümkündür. Daha ətraflı məlumat üçün : https://docs.microsoft.com/en-us/windows/win32/api/_toolhelp/

Məqalədə göstərilən metod "*proof of concept*" olaraq qəbul edilməlidir. Müdaxilə edilən prosesin düzgüz işləyib işləməyəcəyi haqqında tam olaraq test fəaliyyəti həyata keçirilməmişdir. Məqalənin 1. qismində sizlərə TlHelp32 kitabxanasından yayınaraq yüklənilən modulun necə gizlədiləcəyi haqqında məlumat vermişdim. Bu yazıda isə sizlərə icra edilən prosesin imagepathın -da ediləcək dəyişiklik ilə antivirus (ClamAV) proqram təminatından necə yayınmaq olar bu haqda bəhs edəcəyəm. Ilk öncə bizə bypass edəcəyimiz antivirus haqqında bəzi məlumatlar lazımdır. ClamAV memory scan opsiyası ilə proseslər və onlara aid modulların axtarışını həyata keçirir. Virtual maşında olmadığım üçün məcbur özüm test sample yaradıb onun hash summası ilə şəxsi signature baza yaradıb məhz bu faylın axtarışını edəcəyəm.

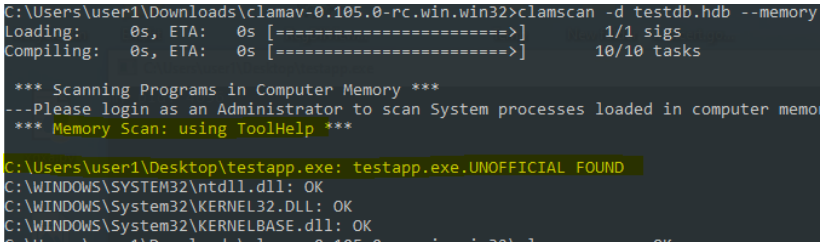
```
#include <windows.h>
#include <stdio.h>

int main(void)
{
    printf("Current Process ID %d\n",
        GetCurrentProcessId());
    getchar();
}
```

Daha sonra en yeni clamav mühərriki ilə testapp prosesi üzərində axtarış edəcəyəm. Lakin ilk öncə imza bazası yaratmalıyıq. ClamAV istifadəçilərə öz şəxsi bazalarını yaratmaq imkanı tanıyır. Bunun üçün sigtool alətindən istifadə edirəm.

```
C:\Users\user1\Downloads\clamav-0.105.0-rc.win.win32>sigtool.exe --md5 c:
\Users\user1\Desktop\testapp.exe > testdb.hdb

C:\Users\user1\Downloads\clamav-0.105.0-rc.win.win32>type testdb.hdb
c688f764b2c3ebc138ef38a004c011a7:104960:testapp.exe
```

```
C:\Users\user1\Downloads\clamav-0.105.0-rc.win.win32>clamscan.exe -d testdb
.hdb c:\Users\user1\Desktop\testapp.exe
Loading:     0s, ETA:   0s [========================>]        1/1 sigs
Compiling:   0s, ETA:   0s [========================>]       10/10 tasks

C:\Users\user1\Desktop\testapp.exe: testapp.exe.UNOFFICIAL FOUND

----------- SCAN SUMMARY -----------
Known viruses: 1
Engine version: 0.105.0-rc
Scanned directories: 0
Scanned files: 1
Infected files: 1
Data scanned: 0.10 MB
Data read: 0.10 MB (ratio 1.00:1)
Time: 0.024 sec (0 m 0 s)
Start Date: 2022:04:05 15:48:38
End Date:   2022:04:05 15:48:38
```

Gördüyünüz kimi testapp faylımızı virus olaraq görür.
İndi eyni əməliyyatı **testapp.exe**-ni işə salıb yoxlayaq.

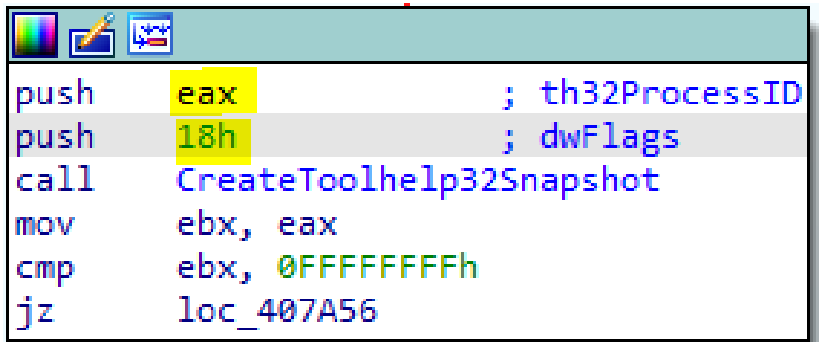


Memory scan əməliyyatıda faylı uğurla aşkar etdi. Clamav mühərrikin scan əmliyyatlarında hansı funksiyalardan istifadə etdiyini görmək üçün clamscan.exe nə static analiz etmək lazımdır.

IDAPro ilə disasm etdim və ClamAV-də eyni part1 -də olduğu kimi modulların siyahısını götürür və scan əməliyyatı modulların pathı üzərindən həyata keçirilir.

```
push    0               ; th32ProcessID
push    2               ; dwFlags
call    CreateToolhelp32Snapshot
```

Burada clamscan flag olaraq 2 (**TH32CS_SNAPPROCESS**) və **PID** 0 parametrləri ilə **CreateToolhelp32Snapshot** funksiyasını çağırır. PID 0 sistemdə olan bütün proseslərin siyahısını götürür. Daha sonra **Proces32** funksiyalarının köməkliyi ilə bu prosesləri enum edir.
Burada PROCESSENTRY32W strukturundan **th32ProcessId**-ni götürüb yenidən **CreateToolhelp32Snapshot** funksiyasını bu PID dəyəri əsasında yenidən çağırır.



```
push    eax             ; th32ProcessID
push    18h             ; dwFlags
call    CreateToolhelp32Snapshot
mov     ebx, eax
cmp     ebx, 0FFFFFFFFh
jz      loc_407A56
```

Bu dəfə isə flag olaraq 0x18 (**TH32CS_SNAPMODULE | TH32CS_SNAPMODULE32**) parametrini CreateToolhelp32Snapshot funksiyasına göndərir.
Bundan sonra isə fayl path üzərində axtarış əməliyyatı aparır. Biz isə yenidən **PEB** (ProcessEnvironmentBlock) struktuna müraciət etməliyik. Burada clamav **Module32** funksiyalarından istifadə etdiyi üçün ilk olaraq əlbətdə PEB_LDR_DATA strukturuna baxmalıyıq. Bizə lazım olan field FullDllName dir.

```
0:001> dt _LDR_DATA_TABLE_ENTRY 0xe43280
ntdll!_LDR_DATA_TABLE_ENTRY
   +0x000 InLoadOrderLinks : _LIST_ENTRY [ 0xe43198 - 0x77165d8c ]
   +0x008 InMemoryOrderLinks : _LIST_ENTRY [ 0xe431a0 - 0x77165d94 ]
   +0x010 InInitializationOrderLinks : _LIST_ENTRY [ 0x0 - 0x0 ]
   +0x018 DllBase          : 0x003e0000 Void
   +0x01c EntryPoint       : 0x003e12e1 Void
   +0x020 SizeOfImage      : 0x1d000
   +0x024 FullDllName      : _UNICODE_STRING "C:\Users\user1\Desktop\testapp.exe"
```

Burada diqqət çəkən bir məqam var. Bu field cari load edilən modulun full pathına işarə (memory pointer) edir. Buffer fieldinə baxdığım zaman adres qismi olduqca maraqlı gəldi.

```
dx -r1 ((ntdll!wchar_t *)0xe41fd8)
!wchar_t *)0xe41fd8)              : 0xe41fd8 : "C:\Users\user1\Desktop\testapp.exe"
```

Bu adres həmçinin bir başqa data strukturunda da qeyd edilib.

PEB - RTL_USER_PROCESS_PARAMETERS.

```
0:001> dx -r1 ((ntdll!_RTL_USER_PROCESS_PARAMETERS *)0xe41b10)
((ntdll!_RTL_USER_PROCESS_PARAMETERS *)0xe41b10)
    [+0x000] MaximumLength    : 0x158a [Type: unsigned long]
    [+0x004] Length           : 0x158a [Type: unsigned long]
    [+0x008] Flags            : 0x4001 [Type: unsigned long]
    [+0x00c] DebugFlags       : 0x0 [Type: unsigned long]
    [+0x010] ConsoleHandle    : 0x8c [Type: void *]
    [+0x014] ConsoleFlags     : 0x0 [Type: unsigned long]
    [+0x018] StandardInput    : 0x94 [Type: void *]
    [+0x01c] StandardOutput   : 0x98 [Type: void *]
    [+0x020] StandardError    : 0x9c [Type: void *]
    [+0x024] CurrentDirectory [Type: _CURDIR]
    [+0x030] DllPath          [Type: _UNICODE_STRING]
    [+0x038] ImagePathName    [Type: _UNICODE_STRING]
```

```
0:001> dx -r1 (*((ntdll!_UNICODE_STRING *)0xe41b48))
(*((ntdll!_UNICODE_STRING *)0xe41b48))                [Type: _UNICODE_STRING]
    [+0x000] Length           : 0x44 [Type: unsigned short]
    [+0x002] MaximumLength    : 0x46 [Type: unsigned short]
    [+0x004] Buffer           : 0xe41fd8 : "C:\Users\user1\Desktop\testapp.exe"
```

Gördüyünüz kimi hər 2 bufferdə eyni adresi göstərir. Bu adreslərin birində edilən dəyişiklik digər buferədə təsir edəcəkdir.
Buna görə **LDR_DATA_TABLE_ENTRY** strukuturunda deyil birbaşa **RTL_USER_PROCESS_PARAMETR** strukturu üzərindən prosesə müdaxilə edəcəyəm. Burada ImagePathName üzərində edəcəyim dəyişiklik ilə ClamAV module list zamanı prosesin full pathını görməyəckdir. Daha dogrusu ImagePathName -i legitim bir fayl pathı ilə dəyişib ClamAV -nın başqa (legitim) faylı scan etməsini test edəcəyəm. İlk olaraq bizə legitim bir program təminatı lazımdır. Bünün üşün kiçik bir proqram təminatı yazdım və test**tpp**.exe olaraq **testapp.exe**-in olduğu qovluğa yazıram və **ImagePathName** bufferində olan yolu bu legitim fayl yolu ilə dəyişirəm.

```
C.:.\.U.s.e.r.s.\.u.s.e.r.1.\.D.e.
s.k.t.o.p.\.t.e.s.t.t.p.p...e.x.e.
```

Faylın hash summası isə tamami ilə fərqlidir.

```
>>> hashlib.md5(open("testtpp.exe", 'rb').read()).hexdigest()
'f97d8181dcfe04af716d3e3c57d39e65'
```

Yenidən **clamscan --memory** əmri ilə sistemdə fəaliyyət göstərən prosesləri scan etdim.

```
C:\Users\user1\Downloads\clamav-0.105.0-rc.win.win32>
Loading:     0s, ETA:   0s [========================>
Compiling:   0s, ETA:   0s [========================>

 *** Scanning Programs in Computer Memory ***
---Please login as an Administrator to scan System pr
 *** Memory Scan: using ToolHelp ***

C:\Users\user1\Desktop\testtpp.exe: OK
```

Gördüyünüz kimi əslində **testapp.exe** - ni scan etmək istəyən clamscan, etdiyim dəyişiklik ilə başqa legitim fayl üzərində axtarış əməliyyatı həyata keçirir və mühərrik tərəfindən təhlükəli fayl olaraq aşkar edilmir.

İstinadlar

https://docs.microsoft.com/en-us/windows/win32/api/winternl/ns-winternl-peb
https://www.geoffchappell.com/studies/windows/km/ntoskrnl/inc/api/ntldr/ldr_data_table_entry.htm