



Practical Malware Analysis & Triage

Malware Analysis Report

Unknown_RE1012018.zip

Feb 2024 | Amna Jasser | v1.0



Table of Contents

Table of Contents	2
Executive Summary	3
High-Level Technical Summary	4
Malware Composition.....	6
Stage2.exe	6
Basic Static Analysis.....	7
Basic Dynamic Analysis	10
Advanced Static Analysis.....	12
Advanced Dynamic Analysis.....	19
Indicators of Compromise	22
Network Indicators	22
Host-based Indicators	23
Appendices	24
A. Yara Rules	24
B. Callback URLs	24

Executive Summary

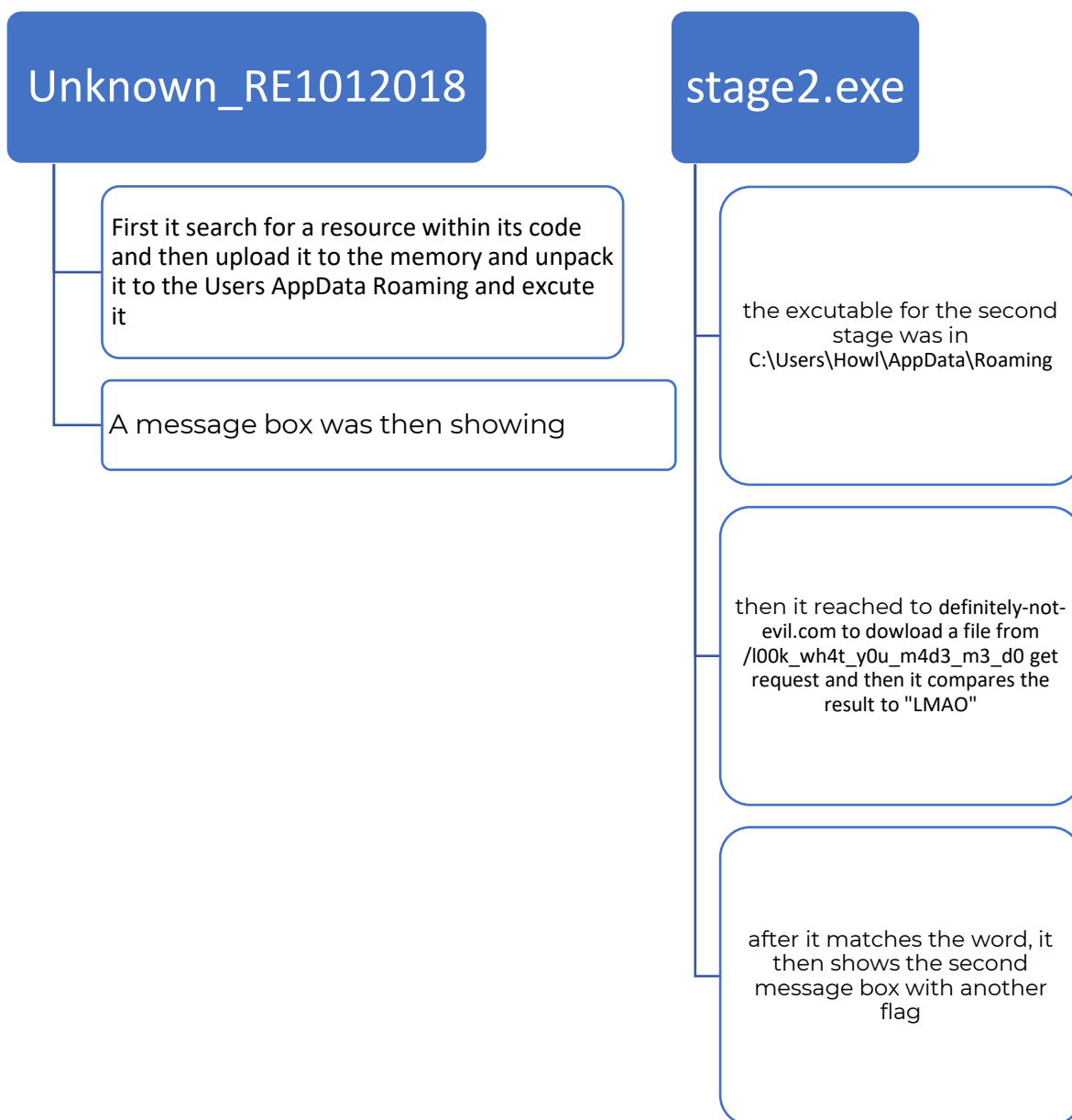
SHA256 hash	EC905CB2CB8E9F74790ADDE2C138807F3A6CDBECD5735FA5035B547280D7DB79
-------------	--

The malware `unknown_re1012018` is a PE (Portable Executable) file that intricately conceals another PE within its code. This embedded PE, identified as "stage2.exe," is systematically extracted, uploaded into the system's memory, and unpacked in the Users' AppData Roaming directory. Following this process, the unpacked resource is executed, revealing a multifaceted approach to its malicious operations. The subsequent steps involve communication with the server definitely-not-evil.com, where a file is downloaded using a GET request from the specified path "/l00k_wh4t_y0u_m4d3_m3_d0." The content of this downloaded file is then compared to the string "LMAO." Upon a successful match, a second message box is displayed, signifying the completion of the second stage and potentially providing a flag or pertinent information about the malware's activities. This layered execution strategy demonstrates the malware's sophistication in evading detection and executing malicious actions.

YARA signature rules are attached in Appendix A. Malware sample and hashes have been submitted to VirusTotal for further examination.

High-Level Technical Summary

Unknown_RE1012018.exe is a PE file that unpacked to another PE file which is the next stage.



- Step 1: Search for a resource “stage2.exe” within the code.
- Step 2: Upload the resource to the memory.
- Step 3: Unpack the resource to the Users' AppData Roaming directory.
- Step 4: Execute the unpacked resource.
- Step 5: A message box appears.
- Step 6: stage2.exe executes and connect to definitely-not-evil.com.
- Step 7: Download a file from /l00k_wh4t_y0u_m4d3_m3_d0 using a GET request.
- Step 8: Compare the result of the download to the string "LMAO."
- Step 9: After matching the word, display a second message box.



Malware Composition

Unknown_RE1012018.exe consists of the following components:

File Name	SHA256 Hash
Unknown_RE1012018	92730427321A1C4CCFC0D0580834DAEF98121EFA9BB 8963DA332BFD6CF1FDA8A
stage2.exe	3EDA6E2DAE6FA86245A688EB24E0A29BF206242560C71C9D7726E5DE02D4538A

Unknown_RE1012018

The initial PE file that is packed with another PE stage2.exe, in the code it holds a flag and a message box will show.

Stage2.exe

This file is unpacked from the first file, then it connects to definitely-not-evil.com and request from it / l00k_wh4t_y0u_m4d3_m3_d0 and compare it to “lmao” and if it’s the same it will show another message box with a flag.



Basic Static Analysis

{Screenshots and description about basic static artifacts and methods}

Looking at the **strings** output we see:

!This program cannot be run in DOS mode. (twice)
hflagh{i_shtay_hout_htoo_hlateh_goth_nothhingh_in_hmy_bhrainj}
00000A70 Im totally malware
00000A84 totally not malware
Dimmaletyoufinishbut
definitely-not-evil.com
100k_wh4t_y0u_m4d3_m3_d0
stage2.exe
00000E07 Dimmaletyoufinishbut
00001F10 100k_wh4t_y0u_m4d3_m3_d0

Since this !This program cannot be run in DOS mode. Was appeared twice then there is an indication that there is another PE file within the PE file.

PEStudio:

Notice the first 2 bytes are **MZ** meaning it's a PE Binary

pFile	Raw Data	Value
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	

file-type: executable



imports (18)	flag (5)	t
GetEnvironmentVariableA	x	!
WriteFile	x	!
CreateProcessA	x	!
TerminateProcess	x	!
GetCurrentProcess	x	!
FindResourceA	-	!
LoadResource	-	!
SizeofResource	-	!
LockResource	-	!
IsDebuggerPresent	-	!
IsProcessorFeaturePresent	-	!
CreateFileA	-	!
Sleep	-	!
SetUnhandledExceptionFilter	-	!
UnhandledExceptionFilter	-	!
lstrcatA	-	!
CloseHandle	-	!
MessageBoxA	-	!

- Notice that it is using [FindResourceA](#) and [CreateProcessA](#). This means the executable might be extracting something out of the resource section and run it as a new process.

It is also checking if it is running in a debugger:

ascii	64	.rsrc	-	import	exception	-	UnhandledExceptionFilter
ascii	17	.rdata	-	import	reconnaissance	T1082 System Information Discovery	IsDebuggerPresent
ascii	17	.rsrc	-	import	reconnaissance	T1082 System Information Discovery	IsDebuggerPresent
ascii	14	.rdata	-	import	resource	-	SizeofResource



And this is the import table:

pFile	Data	Description	Value
00000A00	00002124	Hint/Name RVA	0088 CreateFileA
00000A04	00002132	Hint/Name RVA	014B FindResourceA
00000A08	00002142	Hint/Name RVA	0341 LoadResource
00000A0C	00002152	Hint/Name RVA	0525 WriteFile
00000A10	0000215E	Hint/Name RVA	04B2 Sleep
00000A14	00002166	Hint/Name RVA	04B1 SizeofResource
00000A18	00002178	Hint/Name RVA	00A4 CreateProcessA
00000A1C	0000218A	Hint/Name RVA	053E lstrcatA
00000A20	00002196	Hint/Name RVA	01DB GetEnvironmentVariableA
00000A24	000021B0	Hint/Name RVA	0354 LockResource
00000A28	000021C0	Hint/Name RVA	0052 CloseHandle
00000A2C	00002258	Hint/Name RVA	0300 IsDebuggerPresent
00000A30	0000223A	Hint/Name RVA	04A5 SetUnhandledExceptionFilter
00000A34	000021F6	Hint/Name RVA	04C0 TerminateProcess
00000A38	0000220A	Hint/Name RVA	01C0 GetCurrentProcess
00000A3C	0000221E	Hint/Name RVA	04D3 UnhandledExceptionFilter
00000A40	0000226C	Hint/Name RVA	0304 IsProcessorFeaturePresent
00000A44	00000000	End of Imports	KERNEL32.dll
00000A48	000021DC	Hint/Name RVA	020E MessageBoxA
00000A4C	00000000	End of Imports	USER32.dll

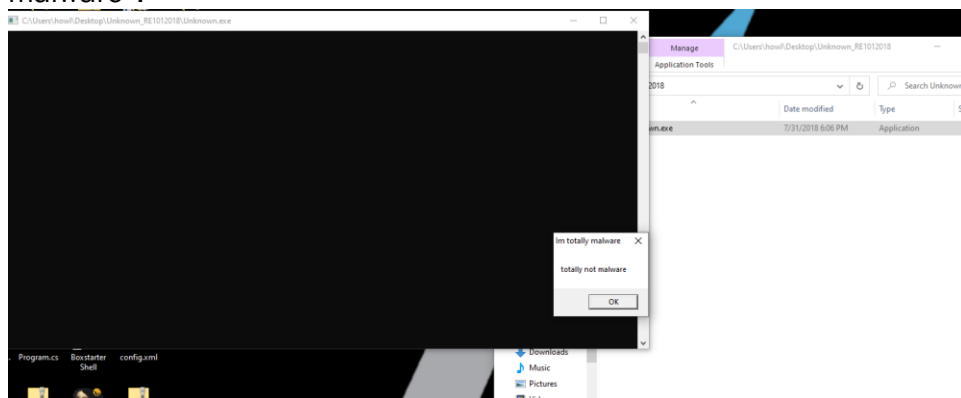


Basic Dynamic Analysis

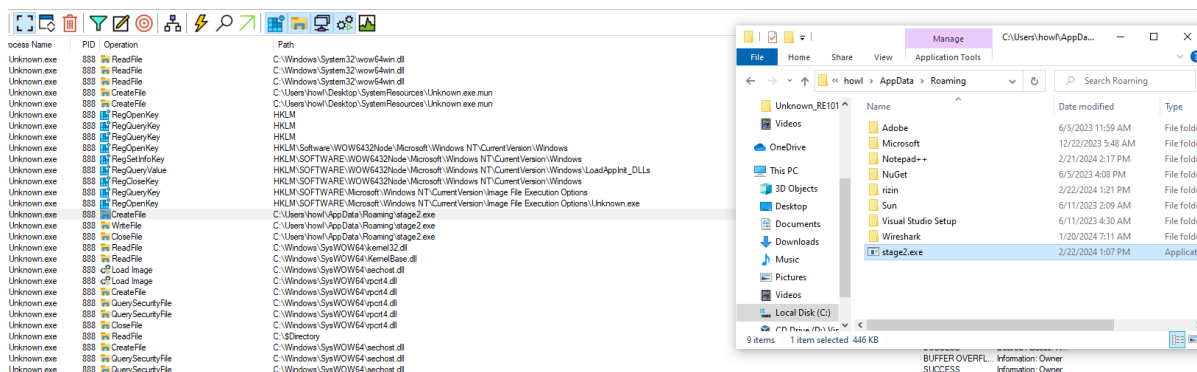
{Screenshots and description about basic dynamic artifacts and methods}

When running the file without internet connection:

- 1- It shows a black command line for a second and a message box with “totally not malware”:



msedge.exe (3228)	Microsoft Edge	C:\Program Files (...)	Microsoft Corporat...	DESKTOP-60ST I...	C:\P...
Unknown.exe (888)		C:\Users\howl\D...		DESKTOP-60ST...	"C:\U...
Conhost.exe (5604)	Console Window ...	C:\Windows\Syst...	Microsoft Corporat...	DESKTOP-60ST...	\??C...
stage2.exe (5588)		C:\Users\howl\Ap...		DESKTOP-60ST...	C:\Us...



Unknown_RE1012018.zip
Feb 2024
v1.0



When there is an internet connection:

Even with internet it is the same result, So we need to check with debugger.



Advanced Static Analysis

{Screenshots and description about findings during advanced static analysis} Could not

Looking at cutter info about this PE:

OVERVIEW

Info

File:	C:\Users\howl\Desktop\Unknown_RE1	FD:	3	Architecture:	x86
Format:	pe	Base addr:	0x00400000	Machine:	i386
Bits:	32	Virtual addr:	True	OS:	windows
Class:	PE32	Canary:	False	Subsystem:	Windows CUI
Mode:	r-x	Crypto:	False	Stripped:	False
Size:	452 kB	NX bit:	True	Relocs:	True
Type:	EXEC (Executable file)	PIC:	False	Endianness:	LE
Language:	c	Static:	False	Compiled:	Tue Jul 24 12:34:19 2018 UTC-8
		Relro:	N/A	Compiler:	N/A

Certificates

Version info

Hashes

MD5:	29f228f3375c489a8a6e31203ab25787
SHA1:	14d713a5c8a2fc01fa2f01d993a249b9fb292810
SHA256:	ec905cb2cb8e9f74790adde2c138807f3a6cdbecd5735fa5035b547280d7db79
CRC32:	5feec509
ENTROPY:	5.825133

Analysis info

Functions:	6
X-Refs:	69
Calls:	67
Strings:	3570
Symbols:	18
Imports:	18
Analysis coverage:	1083 bytes
Code size:	4096 bytes
Coverage percent:	26.4404%

Libraries

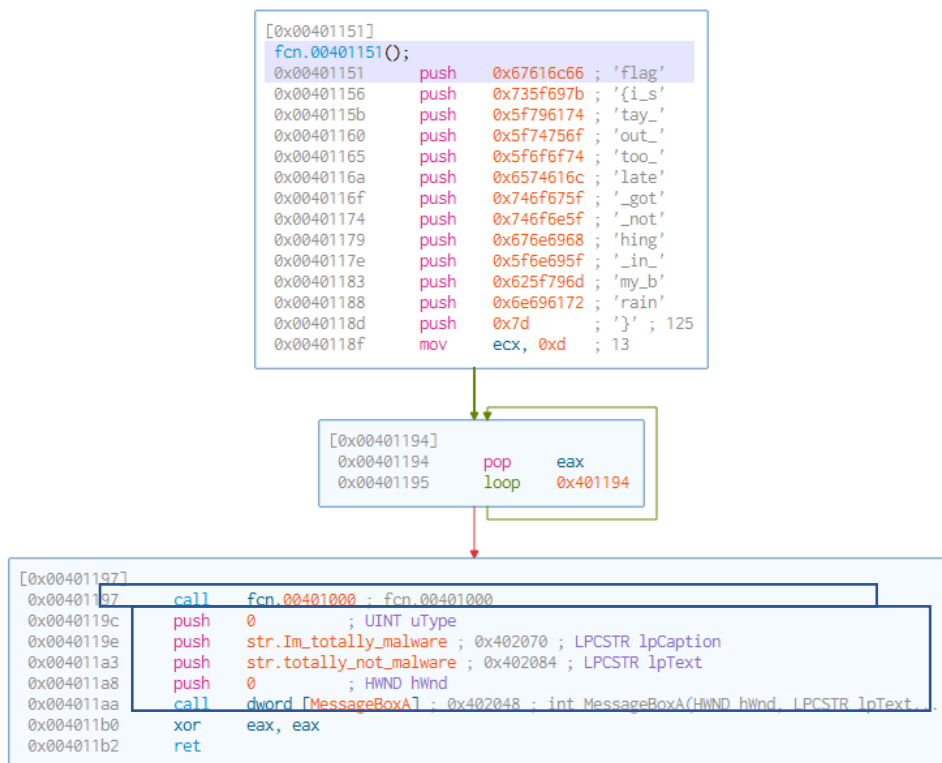
kernel32.dll
user32.dll

We now know it is built with C and x86 instruction.



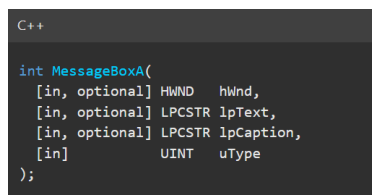
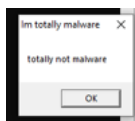
Looking at the main function of Unknown_RE1012018.exe:

Analysing Function **0x00401151**:



- 1- In the first we see the flagh{i_stay_out_too_late_got_nothing_in_my_brain}
- 2- Then it will loop 13 times as loop decrement ecx value.
- 3- Then it will call function **0x00401000**
- 4- Call MessageBoxA function:

The result is this



- 5- Xor eax, eax: This is a common idiom for zeroing out a register, set the `eax` register to 0 and then return from a subroutine.

Unknown_RE1012018.zip

Feb 2024

v1.0



Analyzing function **0x00401000**:

```
[0x0040104f]
0x0040104f  push    0xa          ; 10 ; LPCSTR lpType
0x00401051  push    1            ; 1 ; LPCSTR lpName
0x00401053  push    esi          ; HMODULE hModule
0x00401054  call    dword [FindResourceA] ; 0x402004 ; HRSRC FindResourceA(HMODULE hModule, L...
0x0040105a  mov     esi, eax
0x0040105c  push    esi          ; HRSRC hResInfo
0x0040105d  push    0            ; HMODULE hModule
0x0040105f  call    dword [SizeofResource] ; 0x402014 ; DWORD SizeofResource(HMODULE hModule,...
0x00401065  push    esi          ; HRSRC hResInfo
0x00401066  push    0            ; HMODULE hModule
0x00401068  mov     dword [nNumberOfBytesToWrite], eax
0x0040106b  call    dword [LoadResource] ; 0x402008 ; HGLOBAL LoadResource(HMODULE hModule, H...
0x00401071  mov     ebx, eax
0x00401073  push    ebx          ; HGLOBAL hResData
0x00401074  call    dword [LockResource] ; 0x402024 ; LPVOID LockResource(HGLOBAL hResData)
0x0040107a  test    esi, esi
0x0040107c  je      0x40113d
```

- 1- FindResourceA
- 2- SizeOfResource
- 3- LoadResource
- 4- LockResource

In this code: it Locate a resource within the executable and obtain a handle to the resource information. Then, determine its size and load it into memor(When you call `LoadResource`, it loads the specified resource (identified by the `HRSRC` handle) into memory, and it returns a handle (`HGLOBAL`) to the memory block where the resource is loaded. This handle allows you to access and manipulate the resource's data.).

Finally, obtain a pointer to the resource, allowing access to it and enabling its use.



```
[0x00401082]
0x00401082  push    edi
0x00401083  push    0xffff ; DWORD nSize
0x00401088  push    data.00403398 ; 0x403398 ; LPSTR lpBuffer
0x0040108d  push    str.AppData ; 0x402058 ; LPCSTR lpName
0x00401092  call    dword [GetEnvironmentVariableA] ; 0x402020 ; DWORD GetEnvironmentVariable...
0x00401098  mov     edi, dword [!strcata] ; 0x40201c
0x0040109e  neg     eax
0x004010a0  sbb     esi, esi
0x004010a2  push    data.00402060 ; 0x402060
0x004010a7  and     esi, data.00403398 ; 0x403398
0x004010ad  push    esi
0x004010ae  call    edi
0x004010b0  push    str.stage2.exe ; 0x402064
0x004010b5  push    esi
0x004010b6  call    edi
0x004010b8  push    0 ; HANDLE hTemplateFile
0x004010ba  push    0x80 ; 128 ; DWORD dwFlagsAndAttributes
0x004010bf  push    2 ; 2 ; DWORD dwCreationDisposition
0x004010c1  push    0 ; LPSECURITY_ATTRIBUTES lpSecurityAttributes
0x004010c3  push    0 ; DWORD dwShareMode
0x004010c5  push    0x40000000 ; DWORD dwDesiredAccess
0x004010ca  push    esi ; LPCSTR lpFileName
0x004010cb  call    dword [CreateFileA] ; 0x402000 ; HANDLE CreateFileA(LPCSTR lpFileName, DW...
0x004010d1  mov     edi, eax
0x004010d3  test    edi, edi
0x004010d5  je      0x40113c
```

```
[0x004010d7]
0x004010d7  mov     ecx, dword [nNumberOfBytesToWrite]
0x004010da  push    0 ; LPOVERLAPPED lpOverlapped
0x004010dc  lea     eax, [lpNumberOfBytesWritten]
0x004010df  push    eax ; LPDWORD lpNumberOfBytesWritten
0x004010e0  push    ecx ; DWORD nNumberOfBytesToWrite
0x004010e1  push    ebx ; LPCVOID lpBuffer
0x004010e2  push    edi ; HANDLE hFile
0x004010e3  call    dword [WriteFile] ; 0x40200c ; BOOL WriteFile(HANDLE hFile, LPCVOID lpBuf...
0x004010e9  test    eax, eax
0x004010eb  je      0x40113c
```

- 1- **GetEnviromentalVariableA**: First it get the environment variable in which the resource would be unpacked to which is in AppData, and we see that there is a buffer:

`call edi` is telling the processor to jump to the address stored in the `edi` register, treating it as the starting address of a subroutine. The return address (the address of the instruction following the `call edi` instruction) is pushed onto the stack, allowing the program to return to that address after the subroutine completes.

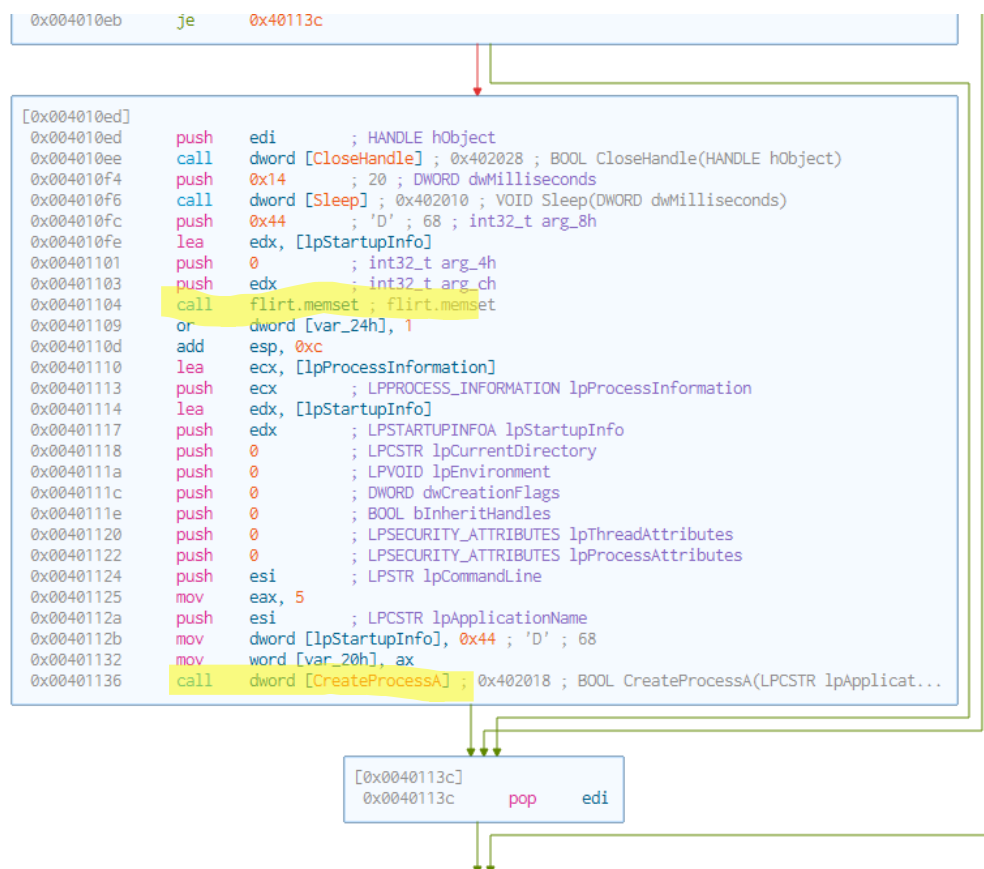


2- WriteFile and CreateFile: This code is creating a file which is stage2.exe and save it in as seeing from string in **C:/Users/Howl/AppData/Roaming**.

CreateFile is used for creating or opening files, obtaining a handle to them, and setting up the necessary parameters. Once you have a file handle obtained through CreateFile, you can use WriteFile to write data to that file. WriteFile is specifically focused on the act of writing data, while CreateFile is focused on obtaining a handle to a file or creating/opening a file.

Then the result of the creation file is stored in eax then to edi and tested on a jump function. If the ZF is set Jump if equal is taken, and only ZF is set when the result of the writefunction is 0, then ZF set to 1 and then the program is exited by following the condition call and jumped to **0x40113c**.

If the ZF is not set then the program continuous to execute:





- 1- It call flirt.memset
- 2- Create a process
- 3- Call a function to terminate any running functions.

Stage2.exe: Function 0x00401000:

```
[0x00401000]
;-- section..text:
fcn.00401000();
; var int32_t var_410h @ stack - 0x410
; var int32_t var_40ch @ stack - 0x40c
; var int32_t var_408h @ stack - 0x408
; var int32_t var_407h @ stack - 0x407
; var int32_t var_8h @ stack - 0x8
0x00401000  push    ebp ; [00] -r-x section size 4096 named .text
0x00401001  mov     ebp, esp
0x00401003  sub     esp, 0x40c
0x00401009  mov     eax, dword section..data ; 0x403000
0x0040100e  xor     eax, ebp
0x00401010  mov     dword [var_8h], eax
0x00401013  push    ebx
0x00401014  push    esi
0x00401015  push    0
0x00401017  push    0
0x00401019  push    0
0x0040101b  push    1 ; 1
0x0040101d  push    str.definitely_not_evil.com ; 0x4020a0
0x00401022  call    dword [InternetOpenA] ; 0x402088
0x00401028  test    eax, eax
0x0040102a  je      0x401140
```



C++

```
BOOL InternetReadFile(
    [in] HINTERNET hFile,
    [out] LPVOID lpBuffer,
    [in] DWORD dwNumberOfBytesToRead,
    [out] LPDWORD lpdwNumberOfBytesRead
);
```

In this function its open a connection to definitely_not_evil.com and its requesting an html page and then it reads the file and store it in ebx register and at the end it closes the handle. the result of the Read file function is saved into EAX



Advanced Dynamic Analysis

Running Stage2.exe in a debugger:

When reaching to the entry point, we saw a call to function 0x401000 and then we saw multiple comparison of the result of that function:

00401169	00BD	FCFEFFFF	00	lea	edi,dword ptr ss:[ebp-104]	
0040116F	C685	FCFEFFFF	00	mov	byte ptr ss:[ebp-104],0	
00401176	E8	85FEFFFF		call	stage2.401000	
0040117B	80BD	FCFEFFFF	6C	cmp	byte ptr ss:[ebp-104],6C	6C: 'l'
00401182	0F85	5E010000		jne	stage2.4012E6	
00401188	80BD	FDFFFFFF	6D	cmp	byte ptr ss:[ebp-103],6D	6D: 'm'
0040118F	0F85	51010000		jne	stage2.4012E6	
00401195	80BD	FEFEFFFF	61	cmp	byte ptr ss:[ebp-102],61	61: 'a'
0040119C	0F85	44010000		jne	stage2.4012E6	
004011A2	80BD	FFFEFFFF	6F	cmp	byte ptr ss:[ebp-101],6F	6F: 'o'
004011A9	0F85	37010000		jne	stage2.4012E6	
004011AF	8B3D	60204000		mov	edi,dword ptr ds:[&LoadIconA]	00402060: "``iuA"
004011B5	33DB			xor	ebx,ebx	

When entering 0x401000 function we saw it open connection and requestion definitely-not-evil.com/100k_wh4t_y0u_m4d3_m3_d0



00401010	8945 FC	mov dword ptr ss:[ebp-4],eax	
00401013	53	push ebx	
00401014	56	push esi	
00401017	6A 00	push 0	
00401019	6A 00	push 0	
00401018	6A 01	push 1	
0040101D	68 A0204000	push stage2.4020A0	4020A0:"definitely-not-evil.com"
00401022	FF15 88204000	call dword ptr ds:[<&InternetOpenA>]	
00401028	85C0	test eax,eax	
0040102A	0F84 10010000	je stage2.401140	
00401030	6A 00	push 0	
00401032	6A 00	push 0	
00401034	6A 03	push 3	
00401036	6A 00	push 0	
00401038	6A 00	push 0	
0040103A	6A 50	push 50	
0040103C	68 A0204000	push stage2.4020A0	4020A0:"definitely-not-evil.com"
00401041	50	push eax	
00401042	FF15 84204000	call dword ptr ds:[<&InternetConnectA>]	
00401048	8BDB	mov ebx,ebx	
00401050	85DB	test ebx,ebx	
00401052	0F84 F0000000	je stage2.401148	
00401058	6A 00	push 0	
0040105A	68 00000080	push 80000000	
0040105F	6A 00	push 0	
00401061	6A 00	push 0	
00401063	6A 00	push 0	
00401065	68 B8204000	push stage2.4020B8	4020B8:"100k_wh4t_y0u_m4d3_m3_d0"
0040106A	68 D4204000	push stage2.4020D4	4020D4:"GET"
0040106F	53	push ebx	
00401070	FF15 8C204000	call dword ptr ds:[<&HttpOpenRequestA>]	
00401076	8BF0	mov esi,eax	
004010AF	50	push eax	
004010B0	8D8D FCFBFFFF	lea ecx,dword ptr ss:[ebp-404]	
004010B6	51	push ecx	
004010B7	6A 70	push 70	
004010B9	68 D8204000	push stage2.4020D8	4020D8:"Content-Type: text/html\nUser-Agent: esi:"D.d"
004010BE	56	push esi	
004010BF	FF15 80204000	call dword ptr ds:[<&HttpSendRequestA>]	
004010C6	6A 00	push 0	
004010C7	6A 00	push 0	
004010C8	68 B8204000	push stage2.4020B8	4020B8:"100k_wh4t_y0u_m4d3_m3_d0"
004010CB	68 D4204000	push stage2.4020D4	4020D4:"GET"
004010D0	53	push ebx	
004010D1	FF15 8C204000	call dword ptr ds:[<&HttpOpenRequestA>]	
004010D7	8BF0	mov esi,eax	
004010D8	85F6	test esi,esi	
004010DA	0F84 A2000000	je stage2.401122	
004010DB	68 F0300000	push 3FF	

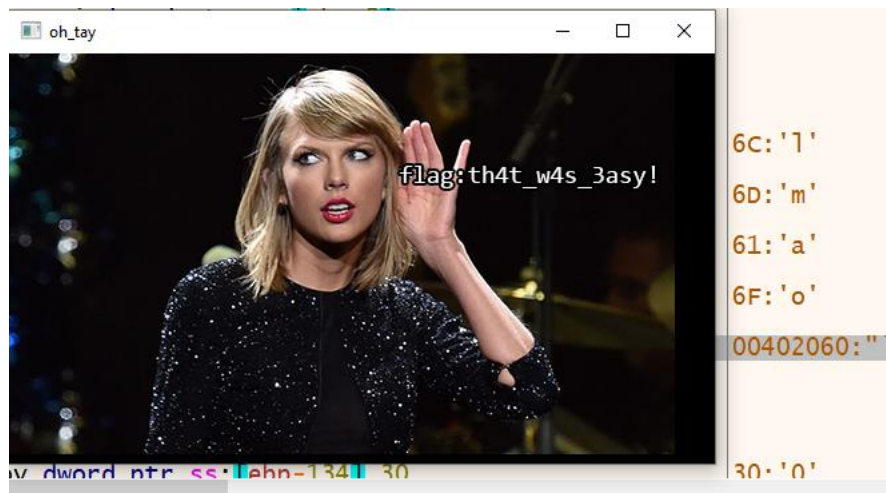
Open a connection to Definitely not evil.com then requesting
/Look_what_you_made_me_do

00401176	E8 85FEFFFF	call stage2.401000	
0040117B	80BD FCFEFFFF 6C	cmp byte ptr ss:[ebp-104],6C	6C:'l'
00401182	0F85 5E010000	jne stage2.4012E6	
00401188	80BD FDFEFFFF 6D	cmp byte ptr ss:[ebp-103],6D	6D:'m'
0040118F	0F85 51010000	jne stage2.4012E6	
00401195	80BD FEFBFFFF 61	cmp byte ptr ss:[ebp-102],61	61:'a'
0040119C	0F85 44010000	jne stage2.4012E6	
004011A2	80BD FFFBFFFF 6F	cmp byte ptr ss:[ebp-101],6F	6F:'o'
004011A9	0F85 37010000	jne stage2.4012E6	

The **HttpSendRequest** function is typically used after **HttpOpenRequest** to actually send the HTTP request to the server.



Then it compares the result to “lmao”, and if it’s the same then the last message box is showed:





Indicators of Compromise

The full list of IOCs can be found in the Appendices.

Network Indicators

{Description of network indicators}

00401010	8945 FC	mov dword ptr ss:[ebp-4],eax	
00401013	53	push ebx	
00401014	56	push esi	
00401015	6A 00	push 0	
00401017	6A 00	push 0	
00401019	6A 00	push 0	
0040101B	6A 01	push 1	
0040101D	68 A0204000	push stage2.4020A0	4020A0:"definitely-not-evil.com"
00401022	FF15 88204000	call dword ptr ds:[<&InternetOpenA>]	
00401028	85C0	test eax,eax	
0040102A	0F84 10010000	je stage2.401140	
00401030	6A 00	push 0	
00401032	6A 00	push 0	
00401034	6A 03	push 3	
00401036	6A 00	push 0	
00401038	6A 00	push 0	
0040103A	6A 50	push 50	
0040103C	68 A0204000	push stage2.4020A0	4020A0:"definitely-not-evil.com"
00401041	50	push eax	
00401042	FF15 84204000	call dword ptr ds:[<&InternetConnectA>]	
00401048	8B08	mov ebx,eax	
00401050	85D8	test ebx,ebx	
00401052	0F84 F0000000	je stage2.401148	
00401058	6A 00	push 0	
0040105A	68 00000080	push 80000000	
0040105F	6A 00	push 0	
00401061	6A 00	push 0	
00401063	6A 00	push 0	
00401065	68 B8204000	push stage2.4020B8	4020B8:"100k_wh4t_y0u_m4d3_m3_d0"
0040106A	68 D4204000	push stage2.4020D4	4020D4:"GET"
0040106F	53	push ebx	
00401070	FF15 8C204000	call dword ptr ds:[<&HttpOpenRequestA>]	
00401076	8BF0	mov esi,eax	
004010AF	50	push eax	
004010B0	8D8D FCFBFFFF	lea ecx,dword ptr ss:[ebp-404]	
004010B6	51	push ecx	
004010B7	6A 70	push 70	
004010B9	68 D8204000	push stage2.4020D8	4020D8:"Content-Type: text/html\nuser-Agent: esi:"d.d"
004010BE	56	push esi	
004010BF	FF15 80204000	call dword ptr ds:[<&HttpSendRequestA>]	
004010C0	50	push 0	
004010C2	50	push 0	
004010C4	68 B8204000	push stage2.4020B8	4020B8:"100k_wh4t_y0u_m4d3_m3_d0"
004010C6	68 D4204000	push stage2.4020D4	4020D4:"GET"
004010C8	53	push ebx	
004010CA	FF15 8C204000	call dword ptr ds:[<&HttpOpenRequestA>]	
004010CF	8BF0	mov esi,eax	
004010D0	85F6	test esi,esi	
004010D2	0F84 A2000000	je stage2.401122	
004010D4	68 FF030000	push 3FF	
004020B8 "100k_wh4t_y0u_m4d3_m3_d0"			
text:00401065 stage2.exe:\$1065 #465			

Using InternetConnectA, HttpOpenRequestA and HttpSendRequestA to definitely-not-evil.com

Unknown_RE1012018.zip

Feb 2024

v1.0



Stage2.exe, this file is executed as second stage:

The screenshot shows a Windows File Explorer window. The address bar at the top displays the path 'C:\Users\howl\AppData\Roaming'. The left sidebar shows the navigation pane with 'Local Disk (C:)' selected. The main pane shows a list of files and folders. The file 'stage2.exe' is selected and highlighted in blue. The status bar at the bottom shows '9 items', '1 item selected', and '446 KB'. The taskbar at the very bottom shows the 'Task View' button, a search bar, and several open applications including 'PowerShell', 'File Explorer', and 'Windows Security'.

Unknown_RE1012018.zip
Feb 2024
v1.0



Appendices

A. Yara Rules

```
rule YARA_example {
  meta:
    description = "Unknown_RE1012018"
    sha256 =
"92730427321A1C4CCFC0D0580834DAEF98121EFA9BB8963DA332BFD6CF1FDA8A"

  strings:
    $string1="hflagh{i_shtay_hout_htoo_hlateh_goth_nothhingh_in_hmy_bhrainj}"
ascii
    $string2 = "Im totally malware" ascii
    $string3="Dimmaletyoufinishbut" ascii
    $string4="stage2.exe" ascii
    $URL1="definitely-not-evil.com" ascii
    $URL2="l00k_wh4t_y0u_m4d3_m3_d0" ascii
    $IS_PE_FILE="MZ"ascii

  condition:
    $IS_PE_FILE at 0 and
    ($string1 and $string2 and $string3 and $string4 ) or ($URL1 or $URL2)
}
```

B. Callback URLs

Domain	Port
definitely-not-evil.com	80