PHANTOM s.r.l

IMPOSSIBLE IS OUR TARGET

EPICODE

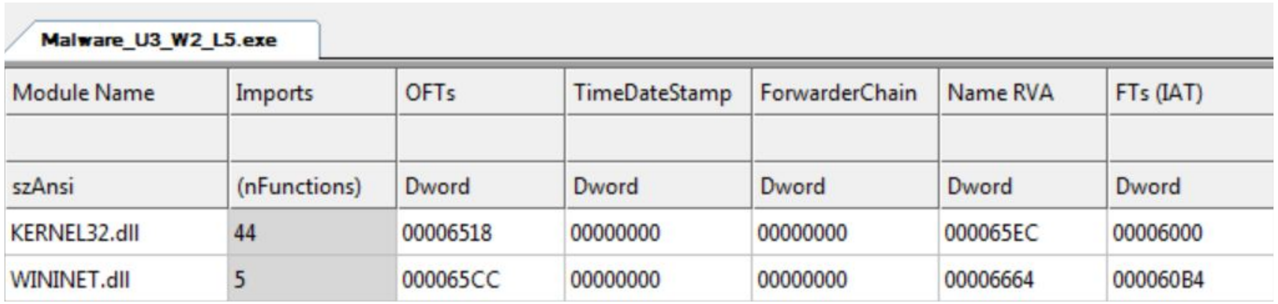
REPORT S10L5

PIO srl

Protecting is better than being safe

Malware Analysis

Imported Library from Malware



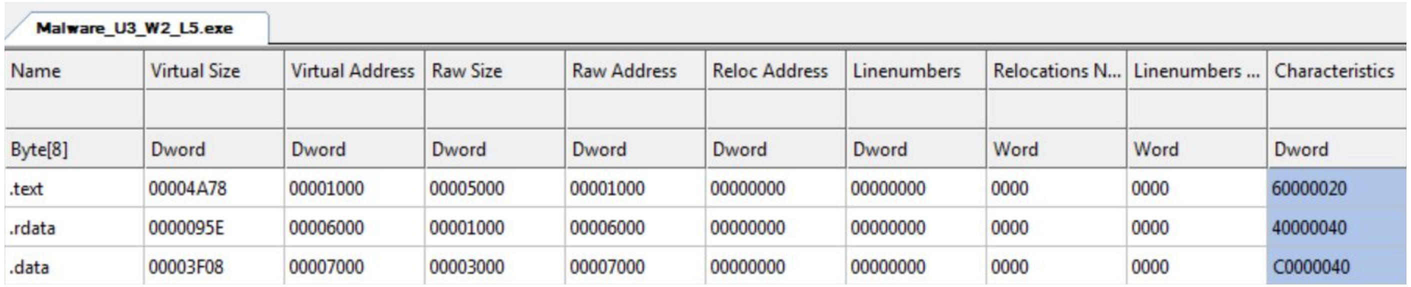
Malware_U3_W2_L5.exe

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

KERNEL32.dll
is an essential component of the Windows operating system and It is a dynamic link library file that contains various functions and resources required for the proper functioning of the Windows kernel.

WININET.dll
is a crucial component of the Windows operating system that plays a significant role in establishing and maintaining internet connections and It's responsible for handling various internet-related functions, such as HTTP, FTP, and HTTPS protocols, as well as managing cookies and caching.

Sections of Malware



Malware_U3_W2_L5.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

.text
is the file that
contain all
instructions to send
at CPU when the
malware starts

.rdata
is the file that
contain the libraries
imported and
exported from the
malware to other

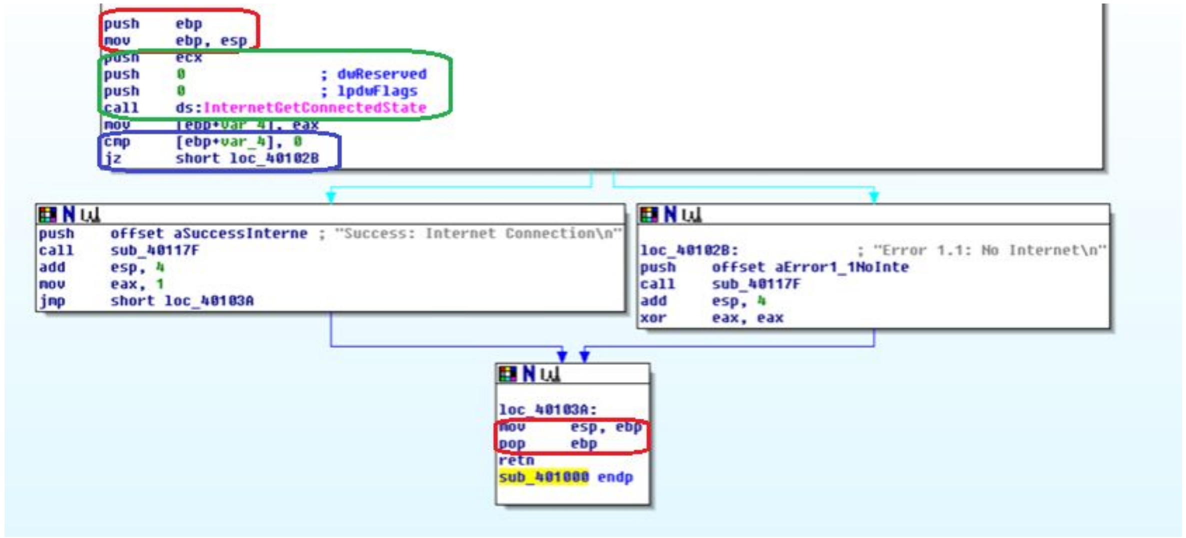
.data
is the file that
include all variables
used in the malware

EPICODE

PHANTOM s.r.l
IMPOSSIBLE IS OUR TARGET

PIO srl
Protecting is better than being safe

Structure in Assembly of Malware



```
push    ebp
mov     ebp, esp
push    ecx
push    0                   ; dwReserved
push    0                   ; lpdwFlags
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

```
loc_40102B:             ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```

```
loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp
```

The malware contain 3 main structures:

- Create e Close the stack
- Call an external function that is used to verify the internet connection state
- IF structure that used to check if connection is successful or not

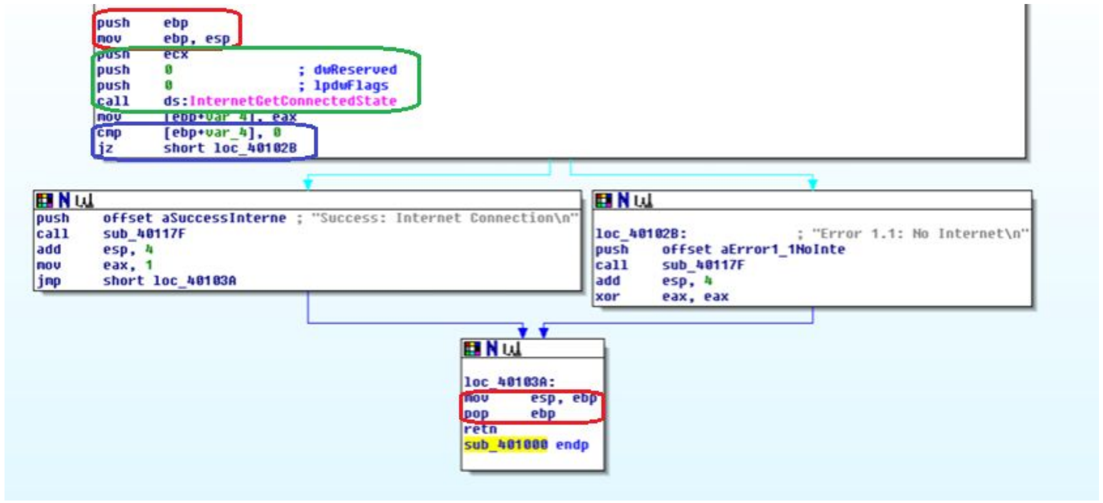
EPICODE

PHANTOM s.r.l
IMPOSSIBLE IS OUR TARGET

PIO srl
Protecting is better than being safe

Try to Know the logic of Malware

The code in question is the result after disassembling a malware. The result is code in x86 assembly language.
The code initially creates a stack and then calls an external function to check if it is connected to Internet.
It then uses an IF-ELSE construct to compare the result of the function called before and if compare is 1(True) follow the instructions in the rectangle on left side of the picture; After the compare(cmp) is 0(False) the program jump to ELSE that is contained in the rectangle at right side of the picture with command jz short loc_40102B.
Then in the last rectangle on the bottom of the picture the malware close the stack and return(retn) values on the main function and with subroutine (sub_401000 endp) take the control to the main part of code



```
push    ebp
mov     ebp, esp
push    ecx
push    0               ; dwReserved
push    0               ; lpdwFlags
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

```
loc_40102B:             ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```

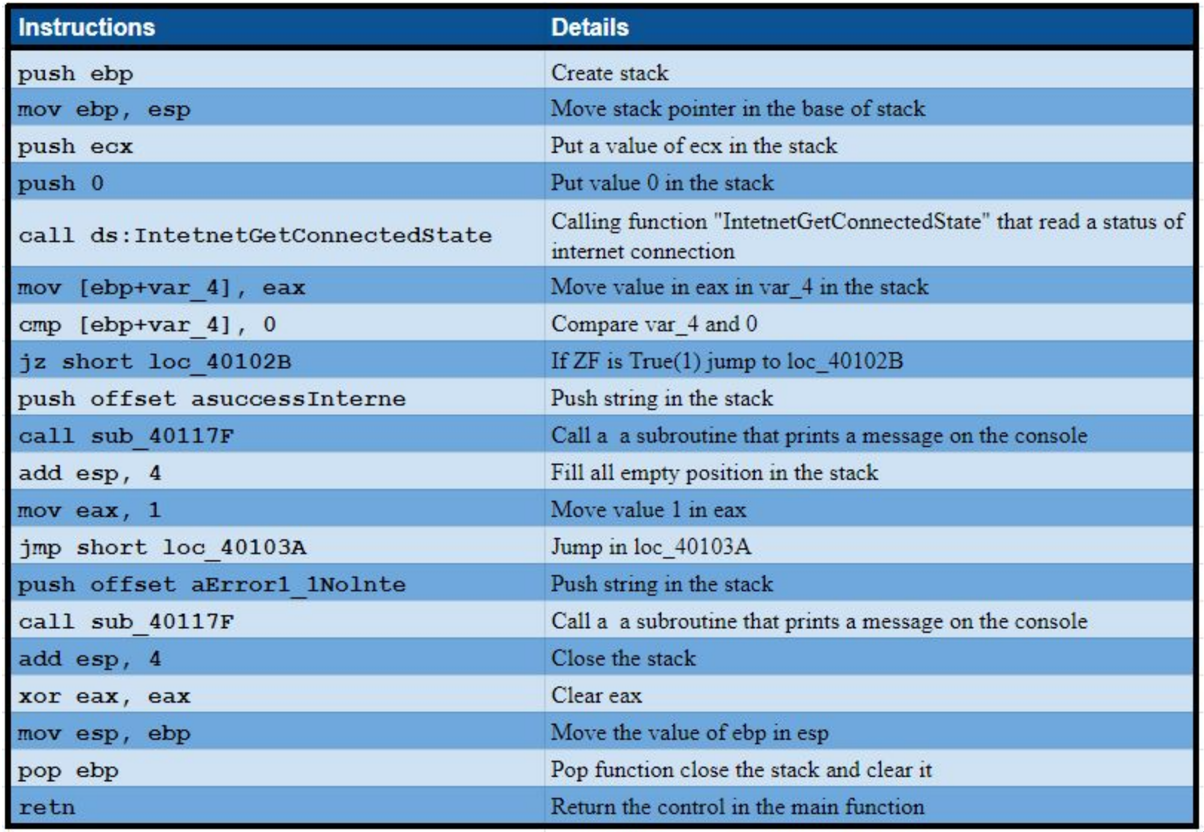
```
loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp
```

EPICODE

PHANTOM s.r.l
IMPOSSIBLE IS OUR TARGET

PIO srl
Protecting is better than being safe

Assembly code Explained
- BONUS Part -



Instructions	Details
push ebp	Create stack
mov ebp, esp	Move stack pointer in the base of stack
push ecx	Put a value of ecx in the stack
push 0	Put value 0 in the stack
call ds:IntetnetGetConnectedState	Calling function "IntetnetGetConnectedState" that read a status of internet connection
mov [ebp+var_4], eax	Move value in eax in var_4 in the stack
cmp [ebp+var_4], 0	Compare var_4 and 0
jz short loc_40102B	If ZF is True(1) jump to loc_40102B
push offset asuccessInterne	Push string in the stack
call sub_40117F	Call a a subroutine that prints a message on the console
add esp, 4	Fill all empty position in the stack
mov eax, 1	Move value 1 in eax
jmp short loc_40103A	Jump in loc_40103A
push offset aError1_1Nolnte	Push string in the stack
call sub_40117F	Call a a subroutine that prints a message on the console
add esp, 4	Close the stack
xor eax, eax	Clear eax
mov esp, ebp	Move the value of ebp in esp
pop ebp	Pop function close the stack and clear it
retn	Return the control in the main function

EPICODE

PHANTOM s.r.l
IMPOSSIBLE IS OUR TARGET

PIO srl
Protecting is better than being safe