

# CS0124

## REPORT

## PROGETTO

## S10L5

**Traccia: Con riferimento al file Malware\_U3\_W2\_L5 presente all'interno della cartella «Esercizio\_Pratico\_U3\_W2\_L5 » sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:**

**1. Quali librerie vengono importate dal file eseguibile?**

**2. Quali sono le sezioni di cui si compone il file eseguibile del malware?**

**Con riferimento alla figura in slide 3, risponde ai seguenti quesiti: 3.**

**Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti )**

**4. Ipotizzare il comportamento della funzionalità implementata**

**5. BONUS fare tabella con significato delle singole righe di codice assembly**

# Prima consegna

Per eseguire un'analisi statica ci avvaliamo del tool CFF in modo tale da visualizzare le librerie importate e le sezioni del file eseguibile.

Come possiamo osservare il file importa 2 librerie

KERNEL32.DLL: una libreria che interagisce direttamente con la memoria e gestisce il sistema operativo

WININET.dll: questa libreria racchiude i protocolli di rete come FTP, NPT e HTTP per la connessione ad internet. Da qui si intende che il malware cerca di connettersi ad internet per eventualmente scaricare altri file da remoto.

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.DLL	44	00000000	00000000	00000000	000065E4	00006000
WININET.dll	5	00000000	00000000	00000000	000065F1	000060B4

# Seconda consegna

A questo punto visualizziamo le sezione di cui e' composto il malware;  
Cosi' facendo potremmo ottenere delle informazione bonus sul funzionamento interno del  
malware e capire come scomporlo.

.text	000002DC	00001000	00001000	00001000	00000000	00000000	0000	0000	60000020
.rdata	00000372	00002000	00001000	00002000	00000000	00000000	0000	0000	40000040
.data	0000008C	00003000	00001000	00003000	00000000	00000000	0000	0000	C0000040

.text racchiude le informazioni che il  
malware va ad eseguire

.rdata comprende le informazioni delle  
librerie importate

.data comprendele informazioni delle variabili del file

Ai fini di analizzare al meglio il file abbiamo anche strings, un programma che ci consente  
di visualizzare tutte le stringhe all'interno del file e che ci consente quindi di recuperare  
maggiori informazioni come la richiesta di connessione,il tipo di browser,un possibile URL  
collegato, e delle info sul dispositivo utilizzato.



```

GetCPInfo
GetACP
GetOEMCP
VirtualAlloc
HeapReAlloc
GetProcAddress
LoadLibraryA
GetLastError
FlushFileBuffers
SetFilePointer
MultiByteToWideChar
LCMapStringA
LCMapStringW
GetStringTypeA
GetStringTypeW
SetStdHandle
CloseHandle
d5@
Error 1.1: No Internet
Success: Internet Connection
Error 2.3: Fail to get command
Error 2.2: Fail to ReadFile
Error 2.1: Fail to OpenUrl
http://www.practicalmalwareanalysis.com/cc.htm
Internet Explorer 7.5/pma
Success: Parsed command is %c

```

```

hq@
rq@
"R@
&R@
Sleep
KERNEL32.dll
InternetGetConnectedState
InternetReadFile
InternetCloseHandle
InternetOpenUrlA
InternetOpenA
WININET.dll
GetCommandLineA
GetVersion
ExitProcess
TerminateProcess
GetCurrentProcess
UnhandledExceptionFilter
GetModuleFileNameA
FreeEnvironmentStringsA
FreeEnvironmentStringsW
WideCharToMultiByte
GetEnvironmentStrings
GetEnvironmentStringsW
SetHandleCount
GetStdHandle
GetFileType
GetStartupInfoA
GetModuleHandleA
GetEnvironmentVariableA
GetVersionExA

```

## Considerazioni sul malware:

A mio avviso questo tipo di malware sembrerebbe essere un trojan che cerca di ottenere informazioni sul dispositivo e successivamente connettersi ad internet per scaricare altri file.

# Terza consegna

Una volta che ci e' stato dato il codice assembly cerchiamo di individuare i costrutti:

```
push    ebp
mov     ebp, esp
```

<----- qui viene creato lo stack

```
push    0                ; dwReserved
push    0                ; lpdwFlags
call    ds:InternetGetConnectedState
```

Qui viene richiamata la funzione InternetGetConnectedState

```
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

Qui viene utilizzato il costrutto IF else ovvero dopo la comparazione se il risultato ottenuto e' 0 salta alla riga 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

```

qui verifichiamo se la funzione ha avuto come risultato 0 e in quel caso non risulta la connessione ad internet,richiamando una subroutine

Al contrario se il risultato ottenuto sara' diverso da 0 abbiamo ottenuto la connessione ad internet, richiamando di nuovo una subroutine

```

mov esp, ebp
pop ebp

```

qui viene copiato il valore del base point all'interno dello stack point e successivamente abbiamo un pop,quindi viene chiuso lo stack

```

push    ebp    creiamo lo stack
mov     ebp, esp    copiamo il valore di esp in ebp
push    ecx    salva il valore corrente su ecx
push    0      ; dwReserved mette il valore 0 alla funzione
push    0      ; lpdwFlags mette il valore 0 all funzione
call    ds:InternetGetConnectedState    richiama questa funzione
mov     [ebp+var_4], eax    sposta il valore di eax in ebp+var_4
cmp     [ebp+var_4], 0    compara la variabile con 0
jz      short loc_40102B    se la variabile e' 0 vai alla riga indicata

```

```

N  L  aggiunge il messaggio di success internet
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F    richiama una subroutine
add     esp, 4    si chiude lo stack point dedicato alla funzione
mov     eax, 1    copia il valore in eax
jmp     short loc_40103A    vai direttamente alla riga indicata

```

```

N  L  loc_40102B: ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte    aggiunge il messaggio error
call    sub_40117F    richiama una subroutine
add     esp, 4    aggiunge valore 4 e chiude lo stack
xor     eax, eax    mette nella variabile eax il valore falso

```

```

N  L  loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp

```

copia il valore di ebp in esp  
 rimuove lo stack  
 ritorna alla memoria principale

Il codice sembra ricercare una connessione internet:  
 Nel momento in cui la connessione risulta attiva richiama una funzione di conferma  
 Se la connessione non e' attiva viene richiamata la subroutine del caso e passa alla gestione degli errori



**GRAZIE!**