

S10-L5

(Bonaldi Cristian)



Progetto - Malware Analysis

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? Fare anche una descrizione
2. Quali sono le sezioni di cui si compone il file eseguibile del malware? Fare anche una descrizione

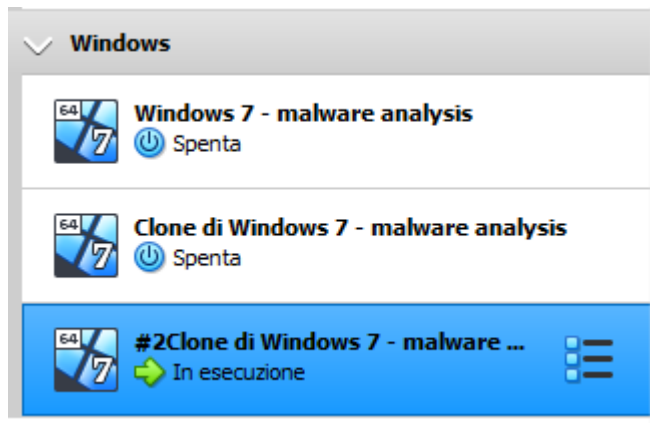
Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti noti)
 4. Ipotesizzare il comportamento della funzionalità implementata
 5. Fare una tabella per spiegare il significato delle singole righe di codice
-

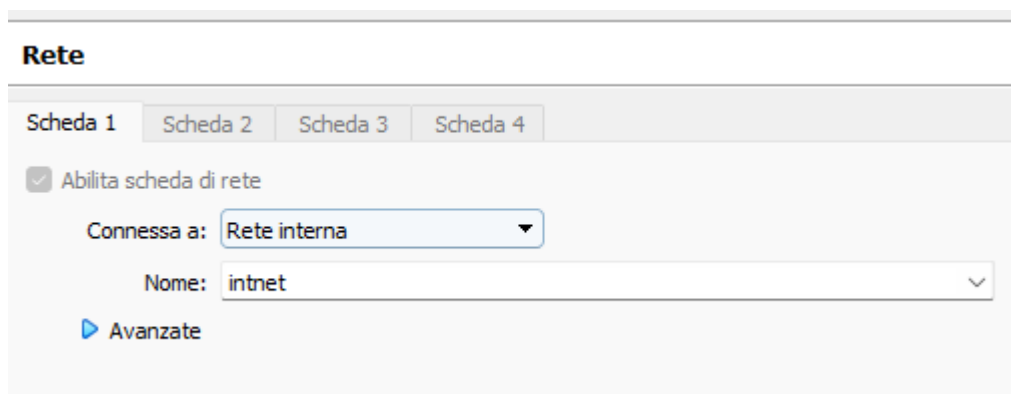
Esercizio e sviluppo:

La traccia richiede di analizzare a fondo un file eseguibile sospetto, ovvero capire se si tratta di un malware o meno, e di conseguenza esaminare il suo comportamento.

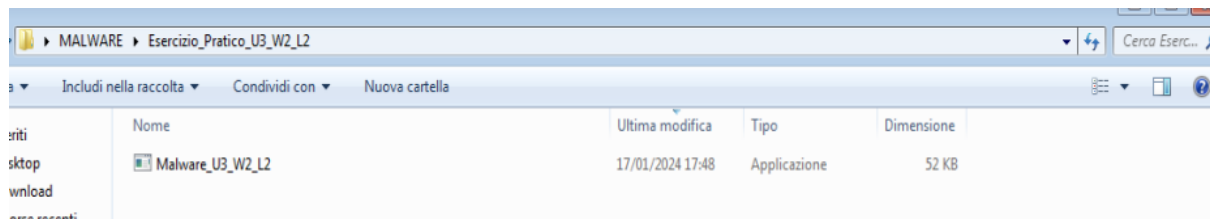
Ovviamente abbiamo bisogno di lavorare all'interno di un ambiente isolato per l'analisi onde evitare ripercussioni dannose. Quindi procediamo alla clonazione della macchina Windows 7 come in figura:



Dopodichè isoliamo la macchina virtuale settandola in Rete Interna dalle impostazioni di rete di Virtual Box.



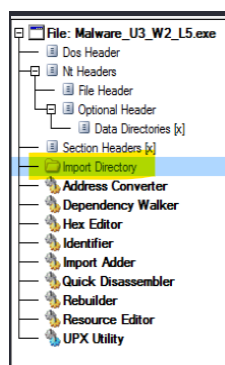
Siamo pronti ora ad avviare la macchina e andare ad analizzare il file eseguibile:



La traccia a questo punto richiede di capire quali librerie vengono importate dall'eseguibile e la loro descrizione.

Andiamo quindi a controllare le **importazioni delle DLL** (Dynamic Link Library), ovvero tutte quelle librerie che contengono funzioni e procedure che il programma può usare per diversi scopi, ovvero accedere a Internet, manipolare file o gestire la memoria. Per fare questo utilizziamo il tool visto nelle precedenti lezioni chiamato **CFF Explorer** per fare un'analisi statica del file eseguibile. Con questo strumento, riusciamo a vedere quali DLL vengono importate dal programma.

Ci rechiamo quindi nella sezione Import Directory del tool dove ci vengono mostrate le dll importate.



All'interno di questa sezione, come possiamo notare, vengono identificate rispettivamente:

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**: è una libreria fondamentale per Windows, che contiene funzioni base del sistema operativo, come la gestione della memoria, dei processi e dei file. In pratica, è una delle librerie più importanti che Windows usa per far funzionare le sue operazioni di base.
- **WININET.dll**: questa invece è una libreria specifica per le operazioni di rete. Contiene funzioni che permettono al programma di accedere a Internet, gestire le connessioni di rete, e fare operazioni come il download o l'upload di file tramite HTTP o FTP.

Come si evince in figura, kernel32.dll importa 44 funzioni, tra cui alcune molto importanti che ci danno un'idea più chiara di cosa fa il malware, ovvero:

Sleep: sospende l'esecuzione per qualche millisecondo o secondo. Potrebbe essere usata per ritardare l'esecuzione di certe operazioni, magari per evitare di essere rilevato.

VirtualFree e **VirtualAlloc:** sono funzioni complementari per gestire la memoria del programma. In sostanza **VirtualAlloc** serve per riservare memoria, mentre **VirtualFree** per liberarla quando non serve più. Questo è molto importante per capire che il malware potrebbe allocare memoria dinamicamente per caricare o gestire dati in modo flessibile.

GetProcAddress: fa sì che il malware ottenga l'indirizzo di una funzione esportata da una DLL e che quindi sia in grado di trovare e usare altre funzioni di sistema in modo dinamico, anche durante l'esecuzione.

LoadLibraryA: carica altre DLL in memoria, permettendo al malware di usare altre librerie di funzioni durante l'esecuzione.

WriteFile: questa funzione scrive dati all'interno di un file. Il malware quindi, mediante l'utilizzo di questa funzione, può creare o modificare file sul sistema infettato.

CloseHandle: chiude gli handle, ovvero gli identificatori delle risorse di sistema (ad esempio file o processi). È importante per evitare perdite di memoria o di risorse.

Procediamo quindi ad identificare le **sezioni** che compongono il malware, ovvero le parti di cui si compone l'eseguibile ognuna contenente diversi tipi di dati e codice necessari per far funzionare il programma.

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

.text: Questa è la sezione dove c'è tutto il codice macchina che viene eseguito dal processore, dove si trovano le istruzioni che fanno funzionare il programma. È la parte più importante dell'eseguibile, perché contiene tutte le operazioni che il malware fa.

.rdata: qui ci sono dati di sola lettura, cioè informazioni che il programma può solo leggere ma non modificare (es. stringhe, indirizzi di funzioni o tabelle di puntatori che vengono lette ma non modificate).

.data: qui si trovano i dati veri e propri che il programma può sia leggere che scrivere durante la sua esecuzione (variabili globali, strutture dati o altre informazioni che il malware gestisce in esecuzione).

Per un'analisi più approfondita del file eseguibile e per un'ulteriore sicurezza in merito, ci rechiamo su **VirusTotal**, una piattaforma online che permette di scansionare file sospetti utilizzando decine di motori antivirus diversi. All'interno di VirusTotal si può caricare un file e in pochi secondi si ottiene un report dettagliato su cosa ne pensano tutti gli antivirus più famosi. Questo ci è molto di aiuto per capire se un file è potenzialmente pericoloso e per vedere se è già stato segnalato come malware.

Quindi andiamo a caricare il file su VirusTotal per avere un'idea più chiara, andando a copiare l'hash in MD5 dell'eseguibile che troviamo nella sezione delle proprietà in CFF Explorer, e quindi:

Malware_U3_W2_L5.exe	
Property	Value
File Name	C:\Users\user\Desktop\MALWARE\Esercizio_Pratico_U3_W2_L5\Malw...
File Type	Portable Executable 32
File Info	Microsoft Visual C++ 6.0
File Size	40.00 KB (40960 bytes)
PE Size	40.00 KB (40960 bytes)
Created	Wednesday 02 February 2011, 17.29.06
Modified	Wednesday 17 January 2024, 18.48.15
Accessed	Wednesday 02 February 2011, 17.29.06
MD5	C0B54534E188E1392F28D17FAFF3D454
SHA-1	BB6F01B1FEF74A9CFC83EC2303D14F492A671F3C

Questo è quanto apparirà tramite l'analisi con VirusTotal:

40
/74

Community Score

40/74 security vendors flagged this file as malicious

Reanalyze

Similar

More

b71777edbf21167c96d20ff803cbcb25d24b94b3652db2f286dcd6efd3d8416a

Size
40.00 KB

Last Analysis Date
1 month ago

EXE

Lab06-02.exe

peexe

direct-cpu-clock-access

runtime-modules

armadillo

checks-network-adapters

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY 7

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label trojan.r002c0pdm21/ymacco

Threat categories trojan

Family labels r002c0pdm21 ymacco

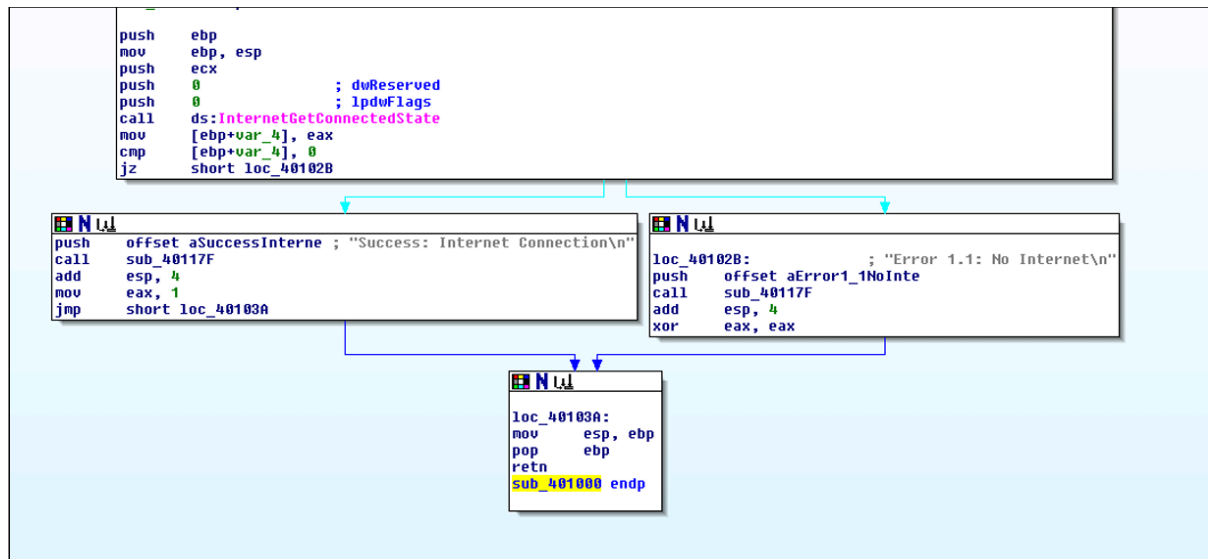
Security vendors' analysis

Do you want to automate checks?

Alibaba	Trojan:Win32/Generic.2cc376c1	AliCloud	Trojan:Win/Ymacco.AMH1
Antiy-AVL	Trojan/Win32.BTSGeneric	Avast	Win32:PUP-gen [PUP]
AVG	Win32:PUP-gen [PUP]	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cylance	Unsafe	DeepInstinct	MALICIOUS
DrWeb	Trojan.MulDrop7.63090	Elastic	Malicious (moderate Confidence)
ESET-NOD32	Win32/Agent.WOO	Fortinet	W32/Agent.WOO!tr
GData	Win32.Trojan.Agent.DZ3C1W	Google	Detected
Gridinsoft (no cloud)	Ransom.Win32.Wacatac.oa!s1	Ikarus	Trojan.Win32.Agent
Kingsoft	Win32.Troj.Undef.a	Lionic	Trojan.Win32.Generic.4!c

Assembly e costrutti noti:

Ora, con riferimento alla figura in slide 3 riportata nella traccia:



Viene richiesto di identificare i costrutti noti visti in precedenza, sulle indicazioni delle istruzioni in **Assembly x86** riportate sopra.

Flusso delle istruzioni:

Creazione dello Stack

- **push ebp**: Salva il valore di EBP (Base Pointer) nello stack.
- **mov ebp, esp**: Inizia un nuovo frame dello stack, dicendo al programma di usare il valore attuale di ESP (che indica la cima dello stack) come nuovo punto di riferimento per le operazioni future. E' quindi come un nuovo punto di partenza per tutto quello che verrà fatto dopo nella funzione.

Chiamata di Funzione:

- **call ds:** chiama una funzione esterna per verificare lo stato della connessione a Internet (InternetGetConnectedState). I parametri vengono pushati nello stack tramite l'istruzione **push 0**:

```

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

```

If (Condizione):


- **cmp [ebp+var_4], 0**: Confronta il valore salvato in [ebp+var_4] con 0.
- **jz short loc_40102B**: jumpa all'etichetta loc_40102B se il confronto (cmp) precedente risulta uguale (jz = jump if zero), e quindi se ZF (Zero Flag) = 1.

```

mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B

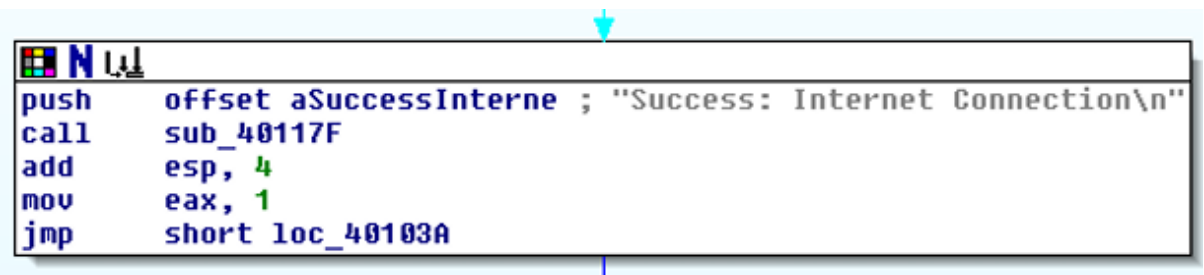
```

Una volta effettuato il salto a loc_40102B quindi, spinge l'indirizzo della stringa sullo stack con un messaggio di errore "*Error 1.1: No Internet\n*" e ripulirà lo stack.

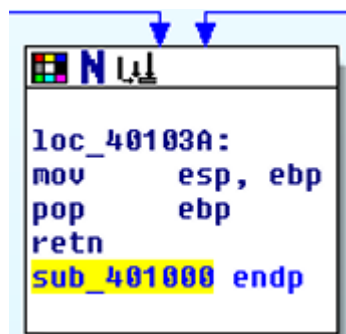


```
loc_40102B:                ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```

Altimenti, pusherà l'indirizzo della stringa con un messaggio di connessione stabilita "Success: Internet Connection\n". Al termine di questa funzione, **jmp short loc_40103A** esegue un salto incondizionato all'etichetta loc_40103A per la pulizia dello stack.



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



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

Nell'ultimo blocco di codice quindi vengono eseguite le seguenti istruzioni:

mov esp, ebp: Ripristina lo stack al punto in cui era prima della funzione, usando il valore salvato in ebp.

pop ebp: Ripristina il valore originale di ebp dallo stack.

retn: Fa tornare il controllo al punto in cui la funzione era stata chiamata.

Tabella codici:

Istruzioni primo blocco:

push ebp	Salva il valore corrente di EBP (Extended Base Pointer) sullo stack
mov ebp, esp	Imposta il valore di esp allo stesso valore di ebp per la creazione di un nuovo stack frame
push ecx	Salva il registro ecx nello stack
push 0 ; dwReserved	Pusha 0 sullo stack (primo argomento della funzione InternetConnectedState)
push 0 ; lpwdFlags	Pusha 0 sullo stack (secondo argomento della funzione InternetConnectedState)
call ds:InternetConnectedState	Chiamata alla funzione InternetConnectedState
mov [ebp+var_4], eax	Salva il valore della funzione (in eax) in [ebp+var_4], ovvero in una variabile locale
cmp [ebp+var_4], 0	Confronta la variabile locale con 0
jz short loc_40102B	Salta a loc_40102B se il valore è 0 (cioè se non è connesso).

Istruzioni blocco connessione stabilita:

push offset aSuccessInterne	Inserisce l'indirizzo della stringa aSuccessInterne nello stack.
call sub_40117F	Chiama la funzione sub_40117F per la gestione del messaggio "Success: Internet Connection\n"
add esp, 4	Ripristina il puntatore dello stack aggiungendo 4 byte a esp, liberando lo spazio che era stato utilizzato per passare l'indirizzo alla funzione.
mov eax, 1	Imposta il registro eax a 1 (successo)
jmp short loc_40103A	Jumpa all'etichetta loc_40103A per continuare l'esecuzione.

Istruzioni blocco connessione rifiutata:

loc 40102B	Etichetta che segna l'inizio del blocco di codice da eseguire quando la connessione non è riuscita.
push offset aError1_1NoInte	Inserisce l'indirizzo della stringa aError1_1NoInte nello stack, che rappresenta un errore di connessione non riuscita.
call sub_40117F	Chiama la funzione sub_40117F per la gestione del messaggio "Error 1.1: No Internet\n"

add esp, 4	Ripristina il puntatore dello stack aggiungendo 4 byte a esp, liberando lo spazio che era stato utilizzato per passare l'indirizzo alla funzione.
xor eax, eax	Imposta il registro eax a 0, che indica un errore (es. connessione rifiutata)

Istruzioni blocco finale:

loc_40103A	Etichetta che segna il punto di ritorno e pulizia dopo la gestione della connessione
mov esp, ebp	Ripristina il puntatore dello stack (esp) al valore di ebp
pop ebp	Ripristina il puntatore della base (ebp) al valore salvato dallo stack
retn	Ritorna al punto di chiamata della funzione,
sub_401000 endp	Conclude la procedura definita sopra nel codice

