

ANALISI STATICA BASICA

Gian Marco Pellegrino

Progetto S10-L5

Il progetto odierno è diviso in due parti: un'Analisi statica basica e un'Analisi di un codice in Assembly di un Malware. In questa sezione verrà affrontata la prima parte.

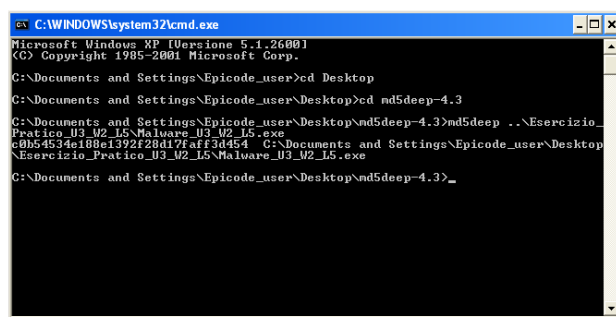
TRACCIA:

1. Quali librerie vengono importate dal file eseguibile?
2. Quali sono le sezioni di cui si compone il file eseguibile del malware?

Prima di rispondere a i quesiti posti nella traccia però, è necessario rispondere ad un'ulteriore domanda: che cos'è un'analisi statica basica?

Un'analisi statica basica è il primo tassello che compone un mosaico di tecniche e fasi per l'analisi di un Malware. Infatti, si potrebbe considerare come un'analisi semplice da eseguire, ma proprio per questo funzionale solo superficialmente, la quale raggiunge presto i suoi limiti quando si ha a che fare con Malware più intricati.

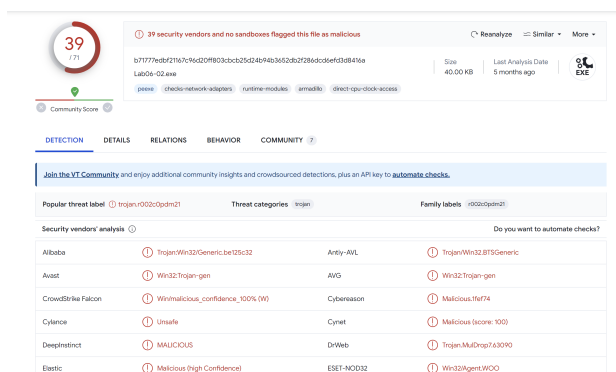
Nell'analisi statica basica si va ad analizzare un file senza eseguirlo, attraverso l'utilizzo di diversi tool: md5deep per esempio, un tool da prompt di comando, utilizzato per ricavare l'hash del file, oppure CFF Explorer, molto utile per analisi di file eseguibili come PE o DLL.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versione 5.1.2600.1
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Epicode_user>cd Desktop
C:\Documents and Settings\Epicode_user\Desktop>cd md5deep-4.3
C:\Documents and Settings\Epicode_user\Desktop\md5deep-4.3>md5deep ..\Esercizio_
Pratico_U3_U2_L5\Malware_U3_U2_L5.exe
e0b54534e188e1392f28d17fa93d454 C:\Documents and Settings\Epicode_user\Desktop
\Esercizio_Pratico_U3_U2_L5\Malware_U3_U2_L5.exe

C:\Documents and Settings\Epicode_user\Desktop\md5deep-4.3>
```



39 / 71

39 security vendors and no sandboxes flagged this file as malicious

Reanalyze Similar More

b71777ecdbf2167e764d0f803bcb25024b4b36b3b2f28dc0efcd3d8476a

Size: 40.00 KB | Last Analysis Date: 5 months ago

Details: checks-network-adapters runtime-modules (arm64) direct-gpu-clock-access

Community Score

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

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

Popular threat label: trojan:Q320pdm21 Threat categories: trojan Family labels: Q320pdm21

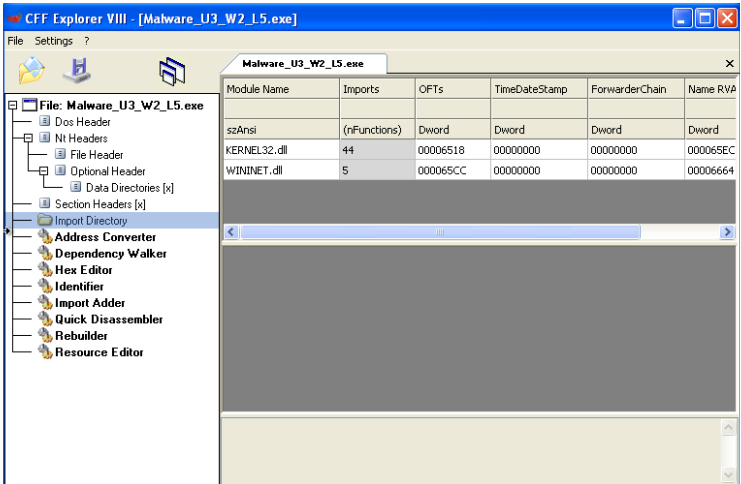
Security vendors' analysis

Security vendor	Detection	Confidence
Alibaba	Trojan:Win32/Generic.bet05c32	Antiy-AVL
Avast	Win32:Trojan-gen	AVG
CrowdStrike Falcon	WinMalicious,confidence_100%(0)	Cybereason
Cylance	Unsafe	Cynet
DeepInsight	MAUCIOUS	Dr-Web
Elastic	Malicious (High Confidence)	ESET-NOD32

Do you want to automate checks?

L'utilizzo di md5deep e VirusTotal non erano richiesti nella traccia ma ho voluto aggiungerli solamente per fornire altri esempi di approccio ad una analisi statica basica. Dopo aver trovato l'hash (codice univoco di un file) è possibile caricarlo su VirusTotal, il quale fornirà molti dettagli sul Malware. In questo caso, il Malware dell'esercizio è segnalato come un Trojan.

Per quanto riguarda CFF Explorer invece, basterà caricare il file al suo interno:



In Import Directory sarà possibile osservare le librerie utilizzate dal file. Le librerie sono delle “raccolte” di funzioni, quest’ultime verranno utilizzate dal file quando ne avrà bisogno. Le librerie possono essere richiamate in tre diversi modi: staticamente, a runtime e dinamicamente.

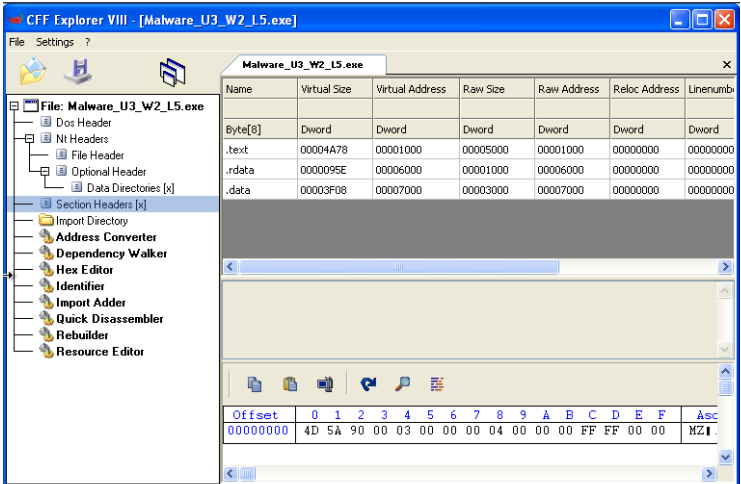
Le librerie seguenti, trovate nell’analisi del Malware, sono dei classici esempi di librerie importate dinamicamente.

Librerie

TERM	DEFINITION	EXAMPLE
KERNEL32.DLL	Contiene le funzioni principali per interagire con il sistema operativo	Manipolazione dei file; La gestione della memoria
WININET.DLL	Al suo interno sono presenti delle funzioni per l'implementazione di alcuni protocolli di rete	HTTP; FTP; NTP

Il secondo punto dell’esercizio chiedeva di quali sono le sezioni del Malware analizzato.

Le sezioni si riferiscono a delle parti distinte all’interno di un file eseguibile, ogni sezione svolge un ruolo specifico nell’ organizzazione di del codice e dei dati all’interno dell’eseguibile.



Queste sono le sezioni riscontrate all’interno del Malware.

Sezioni

TERM	DEFINITION
.text	Questa sezione contiene il codice eseguibile, scritto in Assembly o compilato da un linguaggio di alto livello. Di solito, questa sezione è l’unica che viene eseguita dalla CPU.
.rdata	Questa sezione contiene dati di sola lettura, informazioni riguardo librerie e funzioni. Queste non devono essere modificate durante l’esecuzione.
.data	Contiene solitamente i dati globali del programma eseguibile, per esempio delle variabili globali che devono poter essere richiamate da qualsiasi parte all’interno del programma.

ANALISI COD. ASSEMBLY

Gian Marco Pellegrino

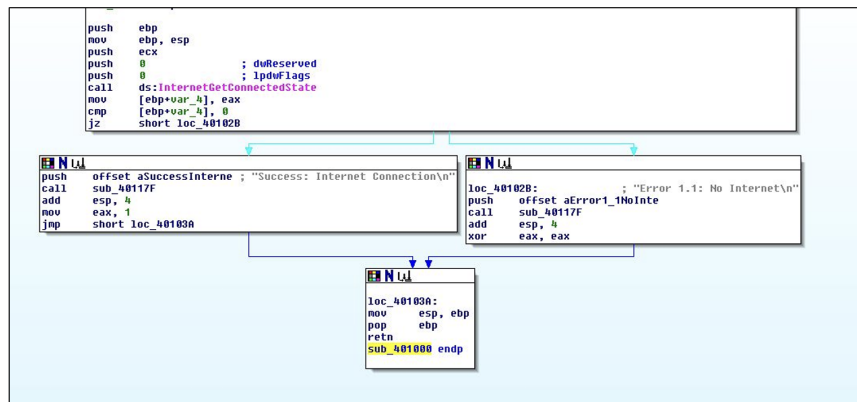
Progetto S10-L5

TRACCIA:

1. Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)
2. Ipotesizzare il comportamento della funzionalità implementata
3. BONUS fare tabella con significato delle singole righe di codice Assembly

Qui sotto ho messo il codice fornito per l'esercizio:

Figura 1



Ho riscritto il codice per avere un'idea più chiara di quello su cui si andrà a lavorare. I "blocchi" dell'immagine precedenti sono stati messi in ordine in questo modo: Blocco superiore, blocco a sinistra, blocco a destra e blocco inferiore.

```
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"
call sub_40117F
add esp, 4
mov eax, 1
jmp short loc_40103A

loc_40102B:
push offset aError1_NoInte ; "Error1.1: No Internet\n"
call sub_40117F
add esp, 4
xor eax, eax

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

ROSSO/ CREAZIONE STACK	queste prime due righe di codice hanno la funzionalità di creare lo stack. Questo è evidente per la presenza di EBP e ESP, che rispettivamente indicano la creazione della base dello stack e del suo estremo superiore.
VERDE/ CHIAMATA DI FUNZIONE	ogni codice mnemonico PUSH inserisce un valore nello stack, il quale poi verrà utilizzato come argomentazione per qualunque funzione venga chiamata.
BLU/ CICLO IF	il classico ciclo IF usato nel linguaggio C potrebbe essere tradotto in modo simile in linguaggio Assembly grazie alle istruzioni cmp e j(x). In questo caso, se questo ciclo IF risulta vero, verrà stabilita correttamente connessione ad internet, se fosse falso invece si salterebbe alla loc_40102B, con l'impossibilità di stabilire una connessione.
GIALLO/ CONN. INTERNET TRUE	questo blocco sta ad indicare la corretta connessione ad internet, nel caso in cui lo stato della connessione sia diverso da 0 viene stampato un messaggio di avvenuta connessione.
VIOLA/ CONN. INTERNET FALSE	qui al contrario, se lo stato della connessione è uguale a 0 viene stampato un messaggio di connessione fallita.
BIANCO/ CHIUSURA	questo blocco serve a pulire lo stack, dato che ha concluso la sua funzione e restituisce il controllo alla funzione chiamante.

Lo scopo di questo Malware potrebbe essere quello di verificare se la vittima ha accesso a internet oppure no, e quindi di conseguenza se, un altro codice possa sfruttare la vulnerabilità, magari attraverso un Downloader oppure potrebbe essere aggiunta un integrazione al codice analizzato per usufruire di una Backdoor.

BONUS

L'esercizio prevedeva la costruzioni una tabella con la spiegazione delle singole righe di codice.

push ebp	creato “Base Stack” pointer, push spinge i valori nello stack
mov ebp, esp	imposta il valore di ebp al valore corrente dello stack
push ecx	viene salvato il registro eax nello stack per poterlo usarle quando necessario
push 0 ;dwReserved	viene spinto il valore nello stack
push 0 ;lpdwFlags	anche qui, spinto il valore nello stack
call ds:InternetGetConnectedState	chiamata della funzione per ottenere lo stato della connessione
mov [ebp+var_4], eax	viene memorizzato il valore della chiamata nella variabile locale [ebp+var_4]
cmp [ebp+var_4], 0	compara il valore memorizzato con 0***
jz short loc_40102B	se il valore memorizzato è 0, salta alla locazione data
push offset aSucessInternet; “Success”	spinge l’offset nello stack
call sub_40117F	chiamata che gestirà il processo di stampa (TRUE)
add esp, 4	aggiunge 4 byte ad esp
mov eax, 1	copia il valore 1 in eax
jmp short loc_40103A	salta alla locazione data
push offset aError1_1NoInte; “Error1.1:No Internet\n”	spinge l’offset nello stack
call sub_40117F	chiamata che gestirà il processo di stampa (FALSE)
add esp, 4	aggiunge 4 byte ad esp
xor eax, eax	usato per impostare il registro aex a 0, quando xor confronta due parametri con lo steso valore il risultato è 0
mov esp, ebp	ripristina esp al suo stato originario prima dell’inizio della funzione
pop ebp	ripristina ebp al suo valore originale, pop serve per togliere “i piatti dalla pila”

retn	restituisce il controllo alla funzione chiamante
sub_401000 endp	simboleggia la fine della procedura

***cmp lavora in modo simile a sub, come una sottrazione. Nel momento in cui due valori siano uguali ES: $5-5=0$ allora lo Zero Flag verrà impostato ad 1, in tutti gli altri casi lo Zero Flag sarà uguale a 0.