

S10-L5

# Report Malware Analyses

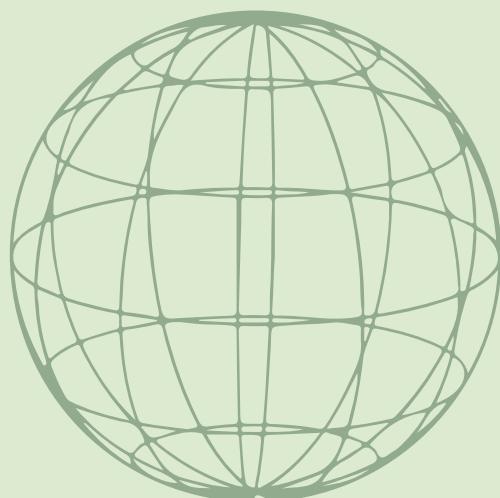
SCRITTO DA

Andrea D.B,  
Lorenzo F.  
Mario R.  
Samuele A.,



# Indice

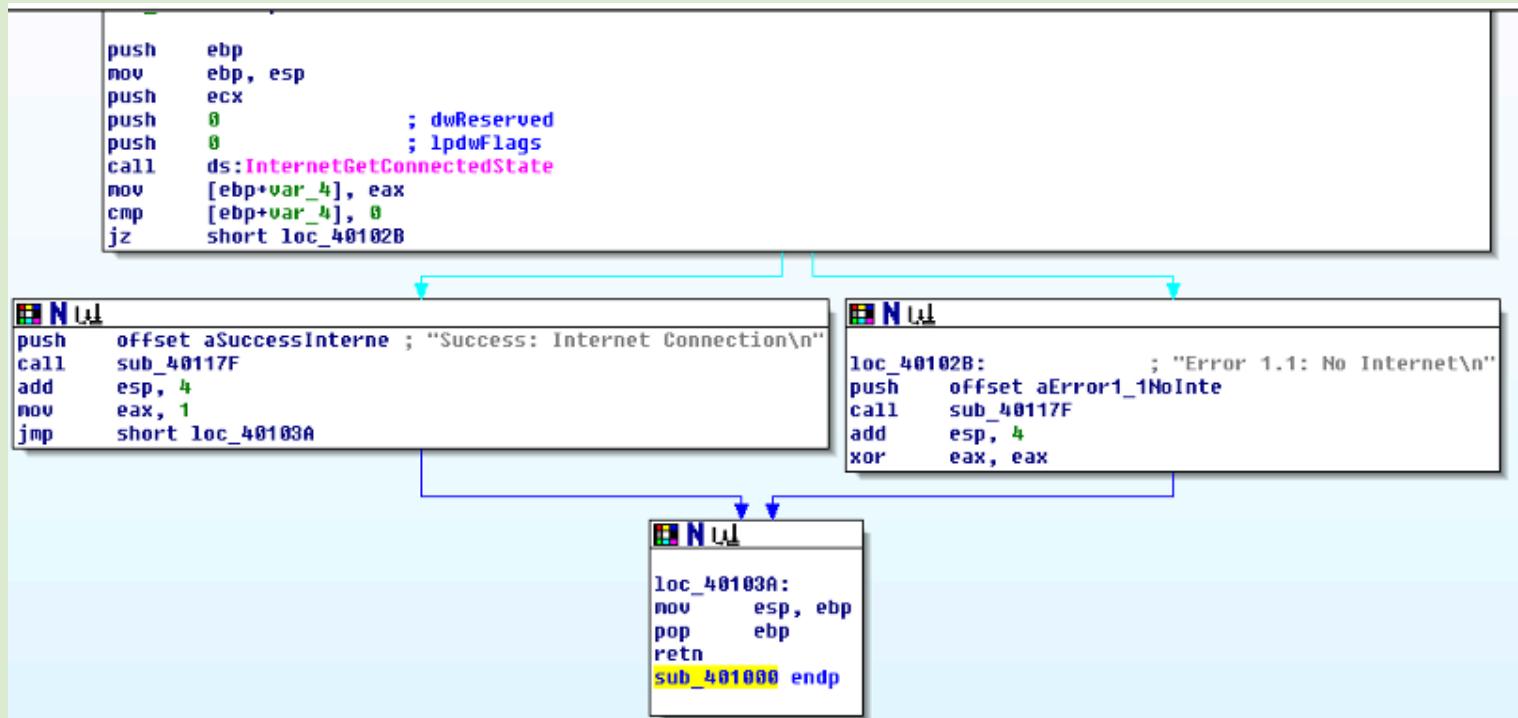
Traccia .....	3
Librerie e loro composizioni .....	4
Identificazione Costrutti Noti .....	7
Ipotizzare Comportamento Funzionalità .....	9
Analisi codice Riga per Riga .....	10
Conclusioni .....	11



# 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,
4. Ipotizzare il comportamento della funzionalità
5. **BONUS** fare tabella con significato delle singole righe di codice assembly



Risposta traccia 1-2

# Librerie e loro composizione

## Analisi Kernel & Wininet

### Librerie

Grazie a CFF Explorer XIII, potremmo effettuare l'analisi del malware. Come primo passo, controlleremo quali directory sono presenti nel malware.

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

### KERNEL32.dll

KERNEL32.dll è una delle librerie più centrali in Windows. Fornisce applicazioni con accesso a funzionalità di basso livello del sistema operativo, tra cui:

- Gestione della memoria
- Operazioni di input/output di file
- Creazione e gestione di processi e thread
- Sincronizzazione
- Comunicazione tra processi

Questo modulo contiene funzioni che sono usate praticamente da ogni programma Windows. Data la sua importanza, è spesso un target per malware che cerca di manipolare il comportamento standard di Windows a proprio vantaggio.

### WININET.dll

WININET.dll offre funzionalità legate alla rete Internet, permettendo agli sviluppatori di Windows di accedere facilmente a protocolli internet come HTTP, FTP e HTTPS. Le funzioni in questa libreria possono essere utilizzate per:

- Scaricare e caricare file da e verso server internet
- Gestire cookie, autenticazione e gestione di sessioni web
- Operare con protocolli web standard per inviare e ricevere dati via Internet

I malware possono utilizzare queste funzionalità per comunicare con server di comando e controllo, scaricare ulteriori componenti dannosi o esfiltrare dati sensibili.

In entrambi i casi, la presenza di queste librerie in un'analisi di malware non è sorprendente dato che forniscono accesso a funzionalità di sistema critico che può essere sfruttato per attività dannose. L'importanza e la versatilità di queste librerie le rendono componenti comuni in molte analisi di software malevolo.

Risposta traccia 1-2

# Librerie e loro composizione

## KERNEL32.dll Funzioni

Elencaremo alcune delle 44 funzioni presenti in "Kernel32.dll"

Malware_U3_W2_L5.exe						
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
000065EC	N/A	000064DC	000064E0	000064E4	000064E8	000064EC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMapStringW
000068FC	000068FC	01BF	LCMapStringA
000068E6	000068E6	01E4	MultiByteToWideChar
00006670	00006670	00CA	GetCommandLineA
00006682	00006682	0174	GetVersion

**Sleep:** Usata per mettere in pausa l'esecuzione del programma per un certo periodo. Questo può essere utilizzato in malware per eludere tecniche di rilevamento dinamico che si aspettano un'attività immediata.

**SetHandle:** Imposta o modifica i permessi o le caratteristiche di un identificatore (handle) di un oggetto sistema, come file, processi o thread.

**GetStringTypeW:** Ottiene informazioni sui caratteri di una stringa, ad esempio se sono lettere, numeri, ecc., utile per la validazione o l'elaborazione delle stringhe.

**GetStringTypeA:** Versione ANSI di GetStringTypeW, per la gestione di stringhe in codifica ANSI.

**MultiByteToWideChar:** Converte una stringa da una codifica multibyte (tipicamente usata per supportare più lingue) in una stringa di caratteri wide (Unicode), importante per l'internazionalizzazione delle applicazioni.

**GetCommandLineA:** Recupera la linea di comando usata per eseguire il programma corrente. Questo può essere usato per analizzare gli input esterni o modificare comportamenti basati su parametri di avvio.

**ExitProcess:** Termina l'esecuzione del processo corrente. Molto usato in malware per chiudere pulitamente dopo aver completato un'attività dannosa.

**TerminateProcess:** Utilizzato per forzare la terminazione di un altro processo, potenzialmente utilizzato per chiudere software di sicurezza o altri processi che potrebbero interferire con l'attività del malware.

Risposta traccia 1-2

# Librerie e loro composizione

## KERNEL32.dll Funzioni

Elencaremo le 5 funzioni presenti in "WININET.dll"

WININET.dll		5	000065CC	00000000	00000000	00006664	000060B4
OFTs	FTs (IAT)	Hint	Name				
Dword	Dword	Word	szAnsi				
00006640	00006640	0071	InternetOpenUrlA				
0000662A	0000662A	0056	InternetCloseHandle				
00006616	00006616	0077	InternetReadFile				
000065FA	000065FA	0066	InternetGetConnectedState				
00006654	00006654	006F	InternetOpenA				



**InternetOpenUrlA:** Aprire un URL specificato usando un handle di sessione Internet. Questo permette al malware di accedere a risorse su Internet.

**InternetCloseHandle:** Chiudere un handle utilizzato in una sessione Internet. Questo è importante per la gestione delle risorse e per evitare perdite di memoria.

**InternetReadFile:** Leggere dati da un handle di un file Internet. È utilizzato per scaricare dati da Internet, che potrebbero includere aggiornamenti del malware o nuovi payload.

**InternetGetConnectedState:** Verificare la connessione a Internet del sistema. Il malware può utilizzare questa funzione per decidere quando attivare o eseguire determinate azioni a seconda della disponibilità della connessione Internet.

**InternetOpenA:** Inizia una sessione Internet che può essere utilizzata per accedere a risorse Internet. È il primo passo per stabilire una connessione Internet dal programma. Ognuna di queste funzioni fornisce capacità fondamentali per un malware che vuole interagire con la rete, eseguire download/upload di dati, o semplicemente monitorare lo stato della connessione Internet del dispositivo infetto. Queste API sono strumenti comuni utilizzati nei software dannosi per realizzare varie attività malevole tramite la rete.

# Identificazione costrutti noti

Chi scrive malware non bada alla singola riga di codice, ma piuttosto struttura il codice in base alla funzionalità che vuole implementare e per farlo fa uso dei **costrutti**, come possono essere i ben noti cicli if, cicli for, gli statement switch, ed i loop come il for ed il while e così via.  
Riconoscere i costrutti in Assembly è una competenza fondamentale, spesso chiamata «**reverse engineering**» ovvero ingegneria inversa. Si riferisce al processo di ricostruire ad alto livello le funzionalità di un malware a partire dall'analisi del codice assembly.

```
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  
mov

Costituiscono il prologo tipico di una funzione, usato per salvare il frame corrente e preparare un nuovo frame per la funzione attuale. Qui si crea uno stack di dimensione non specificata.

Sono istruzioni per posizionare valori sullo stack che saranno usati come parametri per la chiamata a funzione InternetGetConnectedState.

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

cmp  
jz

Formano una struttura condizionale (ciclo if) che controlla se il risultato di InternetGetConnectedState è zero (nessuna connessione) e salta a un blocco di codice per gestire questa situazione.

# Identificazione costrutti noti

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

```
push    offset aSuccessInternet ; "Success: Internet Connection\n"
call    sub_40117F
```

Chiamata di funzione con relativo passaggio di parametri. Qui vi è la gestione della memoria: **Push offset aSuccessInternetC** posiziona l'indirizzo di stringhe costanti sullo stack, che vengono poi passate alla subroutine di stampa.

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

```
loc_40102B:           ; error
push    offset aError1_1NoInte
call    sub_40117F
add    esp, 4
```

Qui vi è la gestione della memoria: **push offset aError1\_1NoInte** posiziona l'indirizzo di stringhe costanti sullo stack, che vengono poi passate alla subroutine di stampa **call**

Qui avviene l'eliminazione dello stack: **mov esp, ebp** e **pop ebp** fanno parte dell'epilogo della funzione, che ripristina il frame precedente alla fine della funzione.  
**retn** termina la funzione e ritorna il controllo al chiamante.

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

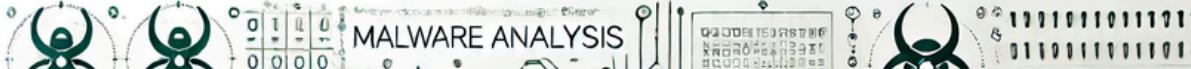


# Ipotizzare Comportamento Funzionalità

Il codice Assembly analizzato è riconducibile ad un malware. In particolare si può dedurre il seguente comportamento:

- 1. Verifica della Connettività Internet:** Il malware controlla se esiste una connessione internet attiva, presumibilmente per stabilire contatti con un server di comando e controllo o per scaricare componenti aggiuntivi.
- 2. Utilizzo delle API di Sistema:** Utilizza funzioni critiche di KERNEL32.dll e WININET.dll per manipolare processi e per operazioni di rete. Questo suggerisce che il malware può gestire processi, comunicare con l'esterno e potenzialmente inviare dati rubati.
- 3. Feedback Condizionale:** Fornisce feedback basato sul successo o fallimento delle sue operazioni, indicativo di una fase di test o debugging.
- 4. Comportamento Adattativo:** Il malware ha diversi percorsi di esecuzione basati sulle condizioni di sistema, come la presenza di connessione internet, e può operare in modo autonomo e silente per evitare rilevazioni.

In sintesi, il malware è progettato per operare in modo furtivo, adattandosi alle condizioni del sistema per eseguire attività dannose e mantenere le proprie operazioni attive.



# Analisi Codice Riga per Riga



- **push ebp** - Salva il valore corrente del base pointer sullo stack per preservare il contesto della funzione chiamante.
- **mov ebp, esp** - Imposta il base pointer per puntare alla cima dello stack corrente, stabilendo così un nuovo frame per la funzione.
- **push ecx** - Salva il contenuto del registro ecx sullo stack. Questo potrebbe essere utilizzato successivamente nella funzione.
- **push 0** - Mette sullo stack il valore 0, utilizzato qui come parametro dwFlags per la funzione InternetGetConnectedState.
- **push 0** - Mette un altro 0 sullo stack, usato come parametro lpdwReserved per InternetGetConnectedState.
- **call ds:InternetGetConnectedState** - Chiama la funzione InternetGetConnectedState per verificare la connessione a Internet. L'indirizzo della funzione è preso dal registro ds.
- **mov [ebp+var\_4], eax** - Memorizza il risultato della funzione nel registro locale var\_4.
- **cmp [ebp+var\_4], 0** - Confronta il valore in var\_4 con 0 per controllare il risultato della connessione.
- **jz short loc\_40192B** - Se il risultato è zero (indicando nessuna connessione), salta alla posizione loc\_40192B che gestisce questo caso.

## Se c'è connessione Internet:

- **push offset aSuccessInternetC** - Carica l'indirizzo della stringa "Success: Internet Connection\n" sullo stack.
- **call sub\_40117F** - Chiama una subroutine che probabilmente stampa la stringa passata.
- **add esp, 4** - Aggiusta lo stack pointer dopo la chiamata alla subroutine.
- **mov eax, 1** - Imposta eax a 1, probabilmente indicando un successo.
- **jmp short loc\_40193A** - Salta alla posizione di fine funzione.

## Se non c'è connessione Internet:

- **push offset aError11NoInter** - Carica l'indirizzo della stringa "Error 1.1: No Internet\n" sullo stack.
- **call sub\_40117F** - Chiama la stessa subroutine per gestire la stampa della stringa.
- **add esp, 4** - Corregge lo stack pointer dopo la chiamata.
- **xor eax, eax** - Azzera il registro eax, probabilmente indicando un errore.

## Chiusura della funzione:

- **mov esp, ebp** - Ripristina lo stack pointer al suo stato originale all'inizio della funzione.
- **pop ebp** - Ripristina il base pointer al suo valore precedente.
- **ret** - Ritorna dal chiamante, finendo l'esecuzione della subroutine.
- **sub\_401900 endp** - Marca la fine della subroutine.

# Conclusioni

- **Librerie Importate:** Il malware importa librerie cruciali come KERNEL32.dll e WININET.dll. Queste librerie forniscono al malware l'accesso a funzioni essenziali per la gestione dei processi, la manipolazione delle stringhe, e per operazioni di rete, che sono fondamentali per le sue attività dannose.
- **Sezioni del File Eseguibile:** Anche se non abbiamo dettagli specifici sulle sezioni del file eseguibile dall' input, tipicamente, file eseguibili malevoli includono sezioni per il testo (codice), dati, risorse e potenzialmente sezioni per il codice criptato o compresso per nascondere le proprie operazioni.
- **Analisi dei Costrutti Noti:** L'analisi del codice Assembly ha rivelato l'uso di costrutti per la gestione dello stack, condizioni (tramite istruzioni di salto condizionale), e cicli. Questi costrutti sono utilizzati per controllare il flusso del programma, gestire l'input/output e implementare logiche condizionali in base allo stato del sistema, come la disponibilità della connessione Internet.
- **Comportamento Ipotizzato:** Il malware verifica la presenza di una connessione internet attiva e reagisce di conseguenza: se c'è connessione, prosegue con le sue operazioni; se non c'è, esegue un set di istruzioni alternative. Questo comportamento indica un'operatività dinamica che si adatta all'ambiente in cui il malware opera, potenzialmente per ottimizzare l'esfiltrazione di dati o il download di ulteriori payload dannosi.
- **Tabella delle Istruzioni:** Nella nostra analisi, abbiamo scomposto le istruzioni Assembly per fornire una visione chiara di ogni operazione eseguita dal malware, dal controllo della connettività internet alla gestione di messaggi di feedback, evidenziando la sofisticata gestione del flusso di controllo e la manipolazione dei dati.

Questo malware dimostra capacità avanzate, utilizzando librerie core di Windows per eseguire operazioni critiche e sfruttare la connettività di rete. La sua capacità di adattarsi basandosi sullo stato di connessione a Internet mostra un livello di sofisticazione che rende essenziale un'analisi continua e approfondita per sviluppare strategie di mitigazione efficaci. Gli operatori della sicurezza dovrebbero prestare particolare attenzione alla prevenzione, rilevazione e risposta a tali minacce, dato il loro potenziale impatto sulle infrastrutture IT.

