



S10/L5

Malware Analyses



Prepared by

Noemi de Martino

Table of Content



<u>Traccia</u>	1
<u>Quesito 1</u>	2
<u>Quesito 2</u>	5
<u>Quesito 3</u>	6
<u>Quesito 4</u>	8
<u>Quesito 5</u>	9
<u>Bonus</u>	12

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:

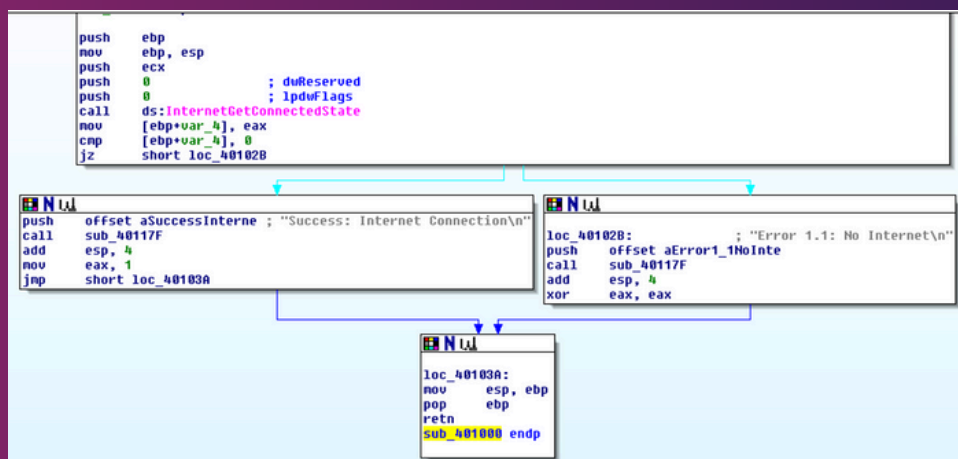
QUESITO 1:

Quali librerie vengono importate dal file eseguibile? Descriverle.

QUESITO 2:

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

Con riferimento alla figura, rispondere ai seguenti quesiti:



QUESITO 3:

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

QUESITO 4:

Ipotizzare il comportamento della funzionalità implementata

QUESITO 5:

Spiegare il significato delle singole righe di codice

↳ Quesito 1

Librerie importate

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

Attraverso CFF Explorer, è possibile analizzare la composizione di un malware. Nella sezione Directory si trovano librerie cruciali come **KERNEL32.dll** e **WININET.dll**.

KERNEL32.dll

KERNEL32.dll è una delle librerie di sistema più fondamentali per il sistema operativo Windows. Le sue funzioni principali sono:

- Gestione della memoria
- Gestione dei processi e dei thread
- Operazioni di input/output
- Gestione delle risorse di sistema
- Sincronizzazione

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

Notiamo anche come vengano importate 44 funzioni da questa libreria.

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





KERNEL32.dll

Analizziamo alcune le 44 funzioni riportate:

- **Sleep:**

000065E4	000065E4	0296	Sleep
----------	----------	------	-------

Sospende l'esecuzione del thread corrente per un intervallo di tempo specificato. Questa funzione è utilizzata per il controllo temporale e la gestione dei ritardi nelle operazioni. Inoltre, può essere sfruttata per eludere tecniche di rilevamento dinamico che si aspettano un'attività immediata.

- **SetStdHandle:**

00006940	00006940	027C	SetStdHandle
----------	----------	------	--------------

Imposta o modifica i permessi o le caratteristiche di un handle associato agli input/output standard (stdin, stdout, stderr) di un processo. Questa funzione permette di reindirizzare l'input/output standard.

- **GetStringTypeW:**

0000692E	0000692E	0156	GetStringTypeW
----------	----------	------	----------------

Determina le caratteristiche di un carattere Unicode, come se esso rappresenti numeri, lettere o altri tipi di caratteri. Questa funzione è utilizzata per la manipolazione e la gestione delle stringhe Unicode.

- **GetStringTypeA:**

0000691C	0000691C	0153	GetStringTypeA
----------	----------	------	----------------

Determina le caratteristiche di un carattere ANSI, come se esso rappresenti numeri, lettere o altri tipi di caratteri. Questa funzione è utilizzata per la manipolazione e la gestione delle stringhe ANSI.

- **MultiByteToWideChar:**

000068E6	000068E6	01E4	MultiByteToWideChar
----------	----------	------	---------------------

Converte una stringa multibyte in una stringa Unicode. Questa funzione è utilizzata per la conversione di dati tra differenti formati di codifica delle stringhe.

- **GetCommandLineA:**

00006670	00006670	00CA	GetCommandLineA
----------	----------	------	-----------------

Recupera il comando di esecuzione utilizzato per avviare l'applicazione in formato ANSI. Fornisce accesso ai parametri di avvio del programma, permettendo di analizzare come è stato invocato il processo.

- **ExitProcess:**

00006690	00006690	007D	ExitProcess
----------	----------	------	-------------

Termina il processo corrente e chiude tutti i thread all'interno del processo. Questa funzione è spesso utilizzata dai malware per terminare il processo dopo aver completato un'attività dannosa.

- **TerminateProcess:**

0000669E	0000669E	029E	TerminateProcess
----------	----------	------	------------------

Termina un processo specificato forzatamente, inclusi tutti i thread all'interno del processo. È utilizzata per chiudere software di sicurezza o altri processi che potrebbero ostacolare le operazioni di un malware.



WININET.dll

WININET.dll è una libreria che fornisce un'interfaccia per applicazioni client che necessitano di accedere a risorse Internet. Questa libreria è ampiamente utilizzata per la gestione delle comunicazioni HTTP e FTP. Le funzionalità principali offerte da WININET.dll includono:

Le funzioni principali sono:

- Gestione delle connessioni Internet
- Download e upload di file
- Gestione delle sessioni di navigazione
- Gestione dei cookie, autenticazione e gestione di sessioni web

Il malware può utilizzare queste funzionalità per comunicare con il server di comando e controllo, scaricare ulteriori componenti dannosi o estrapolare dati sensibili.

In questo caso, vengono importate 5 funzioni da questa libreria.

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

Analizziamo le 5 funzioni riportate:

- **InternetOpenUrlA:** 00006640 00006640 0071 InternetOpenUrlA

Apri un URL specificato utilizzando un handle di sessione Internet. Questa funzione permette al malware di accedere a risorse su Internet, facilitando il collegamento a server remoti per varie operazioni malevoli.

- **InternetCloseHandle:** 0000662A 0000662A 0056 InternetCloseHandle

Chiude un handle utilizzato in una sessione Internet. Questa funzione è cruciale per la gestione delle risorse e per prevenire perdite di memoria, assicurando una chiusura ordinata delle connessioni.

- **InternetReadFile:** 00006616 00006616 0077 InternetReadFile

Legge dati da un handle di un file Internet. Viene utilizzata per scaricare dati da Internet, che possono includere aggiornamenti del malware o nuovi payload da eseguire sul sistema infetto.

- **InternetGetConnectedState:** 000065FA 000065FA 0066 InternetGetConnectedState

Verifica lo stato della connessione Internet del sistema. Il malware può utilizzare questa funzione per determinare quando attivare o eseguire determinate azioni in base alla disponibilità della connessione Internet.

- **InternetOpenA:** 00006654 00006654 006F InternetOpenA

Inizia una sessione Internet che può essere utilizzata per accedere a risorse Internet. È il primo passo per stabilire una connessione Internet da parte del programma, preparando l'ambiente per ulteriori operazioni di rete.

Quesito 2

Sezioni del Malware

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

1 .text

- La sezione **.text** contiene il codice eseguibile del programma. Questa è la parte del file dove risiede il codice macchina che viene eseguito dal processore.
- In un malware, la sezione **.text** include il codice dannoso che viene eseguito una volta che il file è lanciato. Può contenere routine per l'infezione del sistema, la comunicazione con il server di comando e controllo, il furto di dati, e altre attività malevoli.

2 .rdata

- La sezione **.rdata (read-only data)** contiene dati di sola lettura, come stringhe costanti e tabelle di importazione. Questi dati non vengono modificati durante l'esecuzione del programma.
- In un malware, la sezione **.rdata** può contenere URL, chiavi di crittografia, messaggi di errore, e altre stringhe costanti utilizzate dal codice malevolo. Può anche includere informazioni sulle librerie e funzioni di sistema che il malware importa.

3 .data

- La sezione **.data** contiene dati variabili e globali utilizzati dal programma. Questa sezione è leggibile e scrivibile, quindi i dati in essa contenuti possono essere modificati durante l'esecuzione.
- In un malware, la sezione **.data** può includere variabili di configurazione, buffer di dati temporanei, e altre informazioni che cambiano durante l'esecuzione del malware. Può anche contenere dati raccolti dal sistema infetto, che il malware utilizza per prendere decisioni o che invia a server remoti.

Quesito 3

Identificazione Costrutti

```
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    ebp
mov     ebp, esp
```

Il segmento di codice fornito costituisce il prologo tipico di una funzione, utilizzato per salvare il contesto di esecuzione corrente e preparare un nuovo contesto per l'esecuzione della funzione attuale. Questo processo comporta la creazione di uno stack di dimensione non specificata.

Le istruzioni presenti nel codice sono destinate a posizionare i valori nello stack, che saranno successivamente utilizzati come parametri per la chiamata alla funzione

InternetGetConnectedState.

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

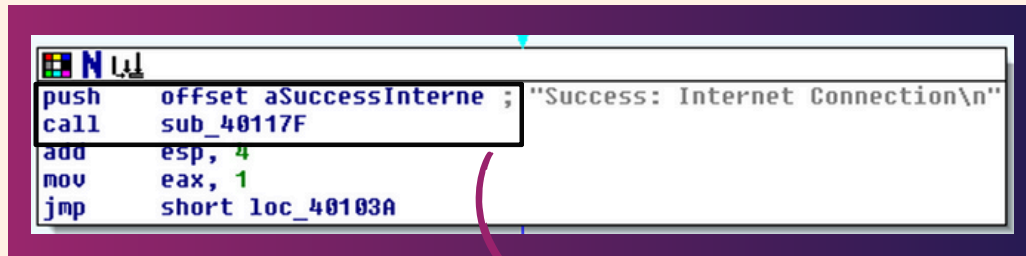
```
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

Inoltre, il codice include una struttura condizionale, sotto forma di un'istruzione **if**, che verifica se il risultato della chiamata a **InternetGetConnectedState** è zero (indicando l'assenza di connessione).

In caso affermativo, il flusso di esecuzione viene deviato verso un blocco di codice dedicato alla gestione di questa situazione.

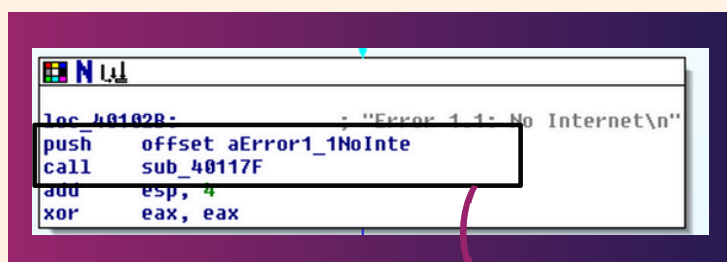
Quesito 3

Identificazione Costrutti



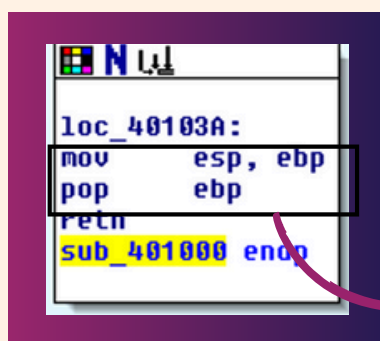
```
push offset aSuccessInterne ; "Success: Internet Connection\n"
call sub_40117F
add esp, 4
mov eax, 1
jmp short loc_40103A
```

Il codice in questione gestisce una chiamata di funzione con il relativo passaggio di parametri. Nella fase di gestione della memoria, l'istruzione **push offset aSuccessInternetC** posiziona l'indirizzo di una stringa costante sullo stack, la quale viene poi utilizzata come parametro per la subroutine di stampa.



```
loc_40102B:
push offset aError1_1NoInte
call sub_40117F
add esp, 4
xor eax, eax
```

L'istruzione **push offset aError11NoInter** posiziona l'indirizzo di un'altra stringa costante sullo stack, che verrà anch'essa passata alla subroutine di stampa tramite l'istruzione call.



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

Per quanto riguarda la gestione della memoria al termine della funzione, le istruzioni **mov esp, ebp** e **pop ebp** fanno parte dell'epilogo della funzione, il quale ripristina il contesto dello stack e il frame precedente. L'istruzione **retn** conclude l'esecuzione della funzione e restituisce il controllo al chiamante.

↙ Quesito 4

Ipotesi sul comportamento delle funzionalità

L'analisi del codice assembly ha rivelato che il software in questione può essere classificato come malware. I comportamenti dedotti sono i seguenti:

- **Verifica della Connettività Internet:** Il malware esegue una verifica della connettività Internet per determinare se esiste una connessione attiva. Questa operazione è presumibilmente finalizzata a stabilire contatti con un server di comando e controllo o a scaricare ulteriori componenti.
- **Utilizzo delle API di Sistema:** Il malware impiega funzioni critiche di **KERNEL32.dll** e **WININET.dll** per manipolare processi e gestire operazioni di rete. Questo indica che il malware ha la capacità di gestire processi, comunicare con entità esterne e potenzialmente trasmettere dati rubati.
- **Feedback Condizionale:** Il malware fornisce feedback in base al successo o al fallimento delle sue operazioni, suggerendo che potrebbe essere in fase di test o debugging.
- **Comportamento Adattativo:** Il malware presenta comportamenti adattativi, scegliendo diversi percorsi di esecuzione in base alle condizioni del sistema, come la presenza di una connessione Internet. Questa adattabilità consente al malware di operare in modo autonomo e discreto, minimizzando il rischio di rilevamento.

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

Quesito 5

Analisi singole righe di codice

```
push ebp
```

Salva il valore corrente del registro **ebp (base pointer)** nello stack per preservare il contesto della funzione chiamante.

```
mov ebp, esp
```

Imposta il registro **ebp** al valore corrente di **esp (stack pointer)**, rendendo **ebp** il puntatore alla base del frame della funzione attuale.

```
push ecx
```

Salva il valore corrente del registro **ecx** nello stack, questo potrebbe essere utilizzato successivamente nella funzione.

```
push 0
```

Mette sullo stack il valore 0, utilizzato come parametro **dwFlags** per la funzione **InternetGetConnectedState**

```
push 0
```

Mette un altro sullo stack il valore 0, utilizzato come parametro **lpdwReserved** per la funzione **InternetGetConnectedState**

```
call ds:InternetGetConnectedState
```

Chiama la funzione **InternetGetConnectedState** per controllare lo stato della connessione a Internet e restituisce un risultato che indica se una connessione è attiva

```
mov [ebp+var_4], eax
```

Memorizza il risultato della funzione nel registro locale **var_4**

```
cmp [ebp+var_4], 0
```

Confronta il valore memorizzato a **[ebp+var_4]** con zero, per determinare se la connessione a internet è **attiva o meno**.

```
jz short loc_40102B
```

Se il risultato è zero è quindi non c'è connessione a internet, salta direttamente a **loc_4019**

Quesito 5

Analisi singole righe di codice

Se c'è connessione ad internet:

```
push offset aSuccessInterne ; "Success: Internet Connection\n"
```

Carica l'indirizzo della stringa **"Success: Internet Connection\n"** sullo stack

```
call sub_40117F
```

Chiama una subroutine (presumibilmente una funzione di stampa) con il messaggio passato.

```
add esp, 4
```

Ripristina il valore di **esp** dopo la chiamata alla funzione, rimuovendo il parametro passato (che occupava 4 byte).

```
mov eax, 1
```

Imposta il registro **eax** a **1**, indicando che l'operazione è probabilmente eseguita con successo.

```
jmp short loc_40103A
```

Salta alla posizione di fine funzione.

Se non c'è connessione ad internet:

```
loc_40102B: ; "Error 1.1: No Internet\n"  
push offset aError1_1NoInte
```

Carica l'indirizzo della stringa **"Error 1.1: No Internet\n"** sullo stack.

```
call sub_40117F
```

Chiama una subroutine (presumibilmente una funzione di stampa) con il messaggio passato.

```
add esp, 4
```

Ripristina il valore di **esp** dopo la chiamata alla funzione, rimuovendo il parametro passato (che occupava 4 byte).

```
xor eax, eax
```

Imposta il registro **eax** a **zero**, indicando che l'operazione non è riuscita.

↳ Quesito 5

Analisi singole righe di codice

Chiusura della funzione:

```
mov esp, ebp
```

Ripristina lo stack pointer al suo stato originale dell'inizio della funzione.

```
pop ebp
```

Ripristina il valore di ebp dal valore precedentemente salvato nello stack

```
ret
```

Completa l'esecuzione della funzione e restituisce il controllo al chiamante.

```
sub_401000 endp
```

Marca la fine della subroutine.

Bonus

Un giovane dipendente appena assunto ha segnalato al reparto tecnico la presenza di un programma sospetto.

Nonostante il suo superiore abbia cercato di rassicurarlo, il dipendente non è convinto e ha richiesto il supporto del **SOC**.

Il file "sospetto" identificato è **ieexplore.exe**, ubicato nella directory

C:\Programmi\Internet Explorer

Nella tua veste di membro senior del SOC, ti è stato assegnato il compito di persuadere il dipendente che il file in questione non rappresenta una minaccia.

Per ottemperare a tale compito, è possibile ricorrere agli strumenti di analisi statica di base e/o analisi dinamica di base precedentemente illustrati durante le lezioni.

Si sottolinea che l'utilizzo di disassemblatori, strumenti di debug o simili non è consentito. Inoltre, si richiede di non basarsi esclusivamente su VirusTotal.

Bonus

Analisi Statica Basica

- Verifichiamo il percorso del file **ieexplore.exe** se si trova nella cartella **C:\Programmi\Internet Explorer**, che è il percorso predefinito per Internet Explorer su sistemi Windows, è altamente probabile che sia legittimo.
- Controlla le proprietà del file tra cui **il produttore** e **la firma digitale** del file.

Property	Value
File Name	C:\Program Files\Internet Explorer\ieexplore.exe
File Type	Portable Executable 64
File Info	Microsoft Visual C++ 8.0 (DLL)
File Size	678.77 KB (695056 bytes)
PE Size	672.00 KB (688128 bytes)
Created	Sunday 21 November 2010, 05.24.43
Modified	Sunday 21 November 2010, 05.24.43
Accessed	Sunday 21 November 2010, 05.24.43
MD5	86257731DDB311FBC283534CC0091634
SHA-1	2AA859F008FAFBAEFB578019ED0D65CD0933981C

Property	Value
CompanyName	Microsoft Corporation
FileDescription	Internet Explorer
FileVersion	8.00.7601.17514 (win7sp1_rtm.101119-1850)
InternalName	ieexplore
LegalCopyright	© Microsoft Corporation. All rights reserved.
OriginalFilename	IEXPLORE.EXE
ProductName	Windows® Internet Explorer

Bonus

Analisi Statica Basica

- Verifichiamo il percorso del file **iexplore.exe** se si trova nella cartella **C:\Programmi\Internet Explorer**, che è il percorso predefinito per Internet Explorer su sistemi Windows, è altamente probabile che sia legittimo.
- Controlliamo le proprietà del file tra cui il **produttore** e la **firma digitale** del file.

iexplore.exe	
Property	Value
File Name	C:\Program Files\Internet Explorer\iexplore.exe
File Type	Portable Executable 64
File Info	Microsoft Visual C++ 8.0 (DLL)
File Size	678.77 KB (695056 bytes)
PE Size	672.00 KB (688128 bytes)
Created	Sunday 21 November 2010, 05.24.43
Modified	Sunday 21 November 2010, 05.24.43
Accessed	Sunday 21 November 2010, 05.24.43
MD5	86257731DDB311FBC283534CC0091634
SHA-1	2AA859F008FAFBAEFB578019ED0D65CD0933981C

Property	Value
CompanyName	Microsoft Corporation
FileDescription	Internet Explorer
FileVersion	8.00.7601.17514 (win7sp1_rtm.101119-1850)
InternalName	iexplore
LegalCopyright	© Microsoft Corporation. All rights reserved.
OriginalFilename	IEXPLORE.EXE
ProductName	Windows® Internet Explorer

↙ Bonus

Analisi Statica Basica

- Ricerchiamo l'hash e possiamo confrontarlo con i database pubblici come quelli di VirusTotal.



iexplore.exe	
Property	Value
File Name	C:\Program Files\Internet Explorer\iexplore.exe
File Type	Portable Executable 64
File Info	Microsoft Visual C++ 8.0 (DLL)
File Size	678.77 KB (695056 bytes)
PE Size	672.00 KB (688128 bytes)
Created	Sunday 21 November 2010, 05.24.43
Modified	Sunday 21 November 2010, 05.24.43
Accessed	Sunday 21 November 2010, 05.24.43
MD5	86257731DDB311FBC283534CC0091634
SHA-1	2AA859F008FAFBAEFB578019ED0D65CD0933981C

<div>1 / 73</div> <div>Community Score</div>	<div>File distributed by Microsoft</div> <div>cfa888e71c65a8807cd719a19c211d1a5dccc04b36d2ebe2d94bf17971ec22690</div> <div>IEXPLORE.EXE</div> <div>peexe direct-cpu-clock-access assembly trusted overlay via-tor known-distributor runtime-modules 64bits signed</div>	<div>Size</div> <div>678.77 KB</div>	<div>Last Analysis Date</div> <div>2 months ago</div>	<div>EXE</div>
--	---	--------------------------------------	---	----------------

Risultati Analisi Statica Basica



Il file **iexplore.exe** è situato nella directory **C:\Programmi\Internet Explorer**, che rappresenta la collocazione predefinita di Internet Explorer.

Le informazioni sul file indicano che il produttore è "**Microsoft Corporation**".

La **firma digitale** del file è stata emessa da Microsoft ed è valida.

Inoltre, l'**hash** del file coincide con quello dei file legittimi di Internet Explorer archiviati nei database pubblici.

Analisi Dinamica Basica

- | | | | | | |
|---|---|---------|----------|------------------------|-----------------------|
| ☐ |  explore.exe | 8.436 K | 23.476 K | 2480 Internet Explorer | Microsoft Corporation |
| |  explore.exe | 9.900 K | 25.480 K | 2592 Internet Explorer | Microsoft Corporation |

- [illegible]

Bonus

Risultati Analisi Statica Basica

- **Process Explorer:**

Durante la scansione con **Process Explorer**, viene evidenziato un comportamento del tutto normale che si traduce nell'apertura dell'applicazione **Microsoft Internet Explorer**.

- Il report di **Regshot** evidenzia le modifiche effettuate al registro di sistema di Windows tra due istantanee:

Vi è un valore aggiunto nel registro di sistema di **Internet Explorer**, presumibilmente correlato alla funzionalità di ripristino del browser.

Sono presenti **8 valori modificati**, tra i quali uno sembra essere un file di configurazione per le impostazioni **di sicurezza anti-phishing** di Internet Explorer, uno per il **controllo del posizionamento** delle finestre di Internet Explorer e un altro che **monitora l'utilizzo di applicazioni** e funzionalità di Windows Explorer.

Le modifiche alle chiavi di registro **sono comuni nelle configurazioni e nelle impostazioni delle applicazioni di Windows**, e al momento non emergono segnali evidenti di attività dannose derivanti esclusivamente da tali modifiche.

Basandoci su queste analisi, possiamo affermare con un elevato grado di certezza che il **file iexplore.exe non è dannoso** e costituisce parte integrante del sistema legittimo di Internet Explorer di Microsoft.