

1. Quanti parametri sono passati alla funzione Main()? Quante variabili sono dichiarate all'interno della funzione Main()?

2. Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno due di quelle identificate

3. Quali librerie importa il Malware? Per ognuna delle librerie importate, fate delle ipotesi

sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.

SVOLGIMENTO

1. Quanti parametri sono passati alla funzione Main()? Quante variabili sono dichiarate all'interno della funzione Main()?

Per avviare il processo di analisi, eseguiamo il software IDA Pro in modalità amministratore. Questa modalità garantisce che il programma abbia tutte le autorizzazioni necessarie per accedere e manipolare i file di sistema e le risorse di rete durante l'analisi.


















Una volta avviato il software, utilizziamo la funzione "Apri" per importare il malware in questione. IDA Pro offre una potente interfaccia di disassemblaggio e decompilazione che ci consente di esaminare in dettaglio il codice del malware. Quando carichiamo il file, IDA Pro inizia a disassemblare automaticamente il codice eseguibile, traducendo il linguaggio macchina in un formato assembly leggibile.

La prima schermata significativa che IDA Pro ci presenterà è tipicamente la funzione `main` del codice, o una funzione equivalente che rappresenta il punto di ingresso del programma. Questa funzione è il cuore dell'esecuzione del malware e spesso rivela la struttura e il flusso di controllo del programma.

Analizzando la funzione ``main``, possiamo individuare i parametri della funzione all'interno delle parentesi, che rappresentano i dati passati alla funzione stessa. IDA Pro ci permette di visualizzare anche gli offset, che sono essenziali per comprendere l'accesso alla memoria del programma.

Gli offset sono indicati in relazione al puntatore dello stack (stack pointer). In genere, gli offset delle variabili locali sono preceduti da un segno negativo, indicando che queste variabili si trovano in posizioni di memoria relative a indirizzi inferiori rispetto al puntatore dello stack. Ad esempio, un offset di ``-4`` può indicare una variabile locale che si trova 4 byte sotto l'indirizzo del puntatore dello stack corrente.

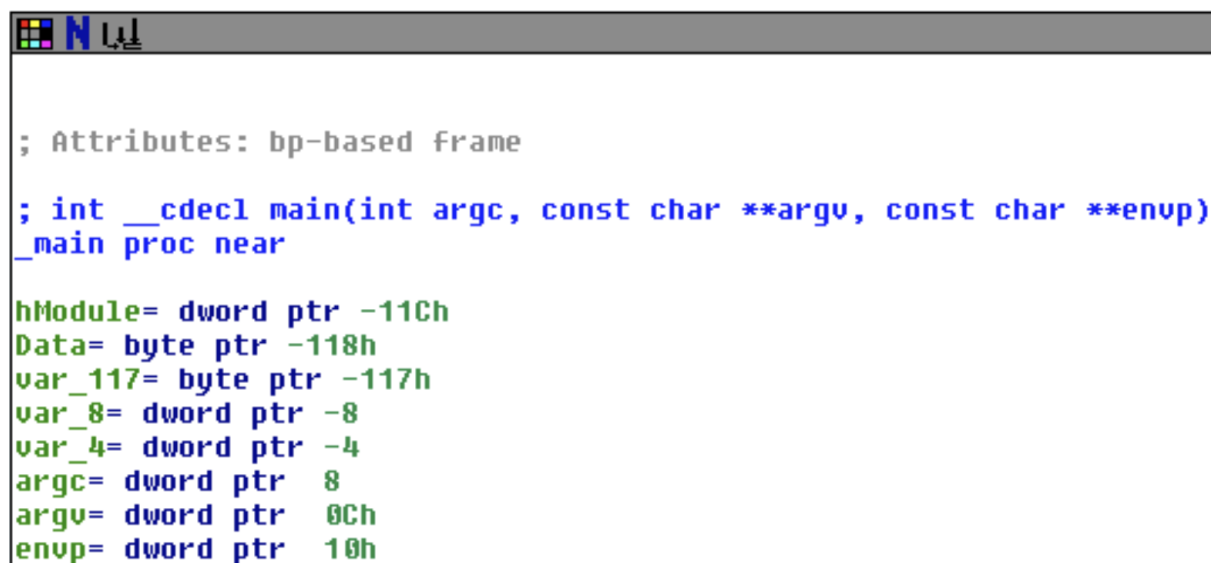
Al contrario, gli offset dei parametri della funzione sono solitamente preceduti da un segno positivo, indicando che si trovano in posizioni di memoria relative a indirizzi superiori rispetto al puntatore dello stack. Questi parametri vengono passati alla funzione tramite lo stack e quindi risiedono a indirizzi più alti rispetto al puntatore dello stack al momento della chiamata della funzione. Un esempio comune potrebbe essere un offset di ``+8``, che rappresenta un parametro che si trova 8 byte sopra l'indirizzo del puntatore dello stack.

Function name	Segment	Start	Length	R	F	L	S	B	T	=
 sub_401000	.text	00401000	0000007F	R	.	.	.	B	T	.
 sub_401080	.text	00401080	00000145	R	.	.	.	B	T	.
 _main	.text	004011D0	000000C9	R	.	.	.	B	T	.
 sub_401299	.text	00401299	00000031	R
 _fclose	.text	004012CA	00000056	R	.	L	.	.	T	.
 _fwrite	.text	00401320	0000010A	R	.	L	.	B	T	.
 __fsopen	.text	0040142A	00000020	R	.	L	.	.	T	.
 _fopen	.text	0040144A	00000013	R	.	L	.	.	T	.
 _strchr	.text	00401460	00000027	R	.	L	.	B	T	.
 start	.text	00401487	000000D4	R	.	L	.	B	.	.
 __amsg_exit	.text	00401566	00000025	R	.	L	.	.	T	.
 _fast_error_exit	.text	0040158B	00000024	R	.	L	S	.	T	.
 __stbuf	.text	004015AF	0000008D	R	.	L
 __ftbuf	.text	0040163C	0000003D	R	.	L
 sub_401679	.text	00401679	0000077E	R	.	.	.	B	T	.
 _write_char	.text	00401E17	00000035	R	.	L	S	B	T	.
 write_multi_char	.text	00401F4C	00000031	R	.	L	S	.	T	.

Alla funzione main vengono passati **3 parametri**: Un **int** e due **char** e **5 variabili**: hModule, Data, var_117, var_8 e var_4.

Le variabili si distinguono dai parametri in quanto sono ad un offset negativo rispetto al registro EBP mentre, argc, argv e envp, essendo parametri, si trovano ad un offset positivo rispetto a EBP.

Di seguito lo screen da IDA:



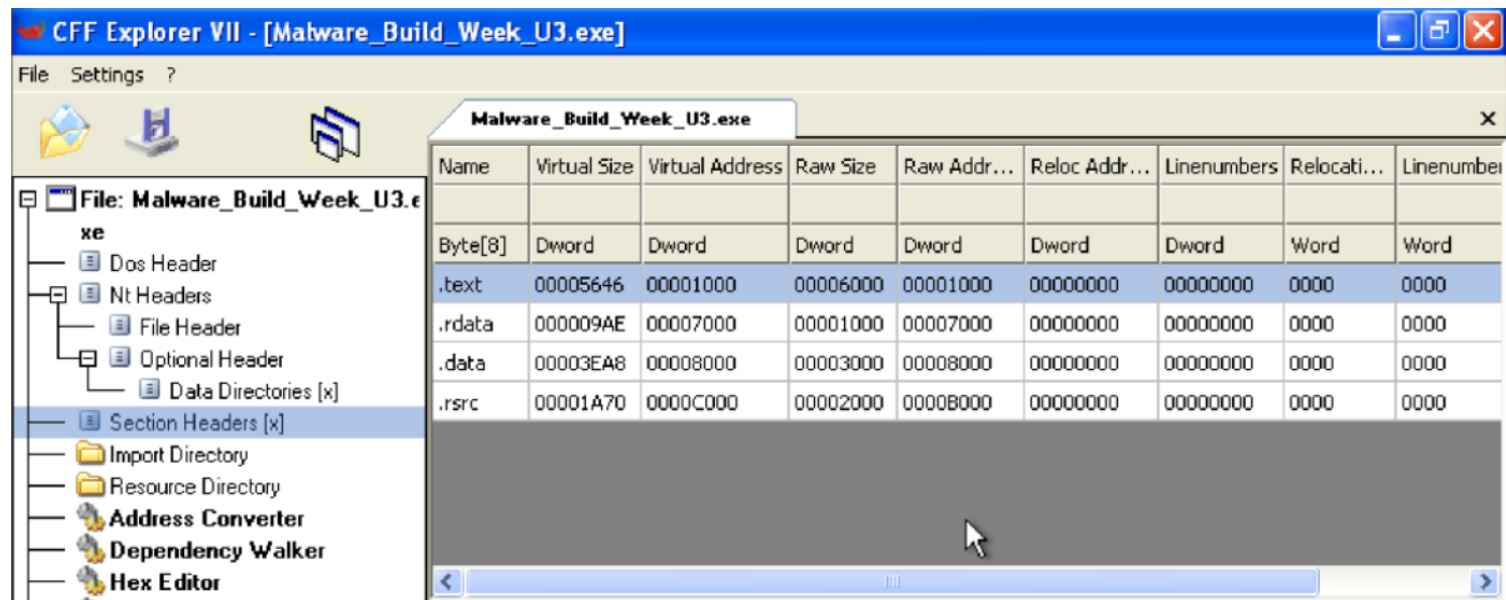
```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

hModule= dword ptr -11Ch
Data= byte ptr -118h
var_117= byte ptr -117h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
```

2. Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno due di quelle identificate

Per rispondere a questa domanda andremo a utilizzare CFF Explorer



Come si può osservare dalle immagini, questo file è suddiviso in diverse sezioni, ognuna con uno scopo specifico:

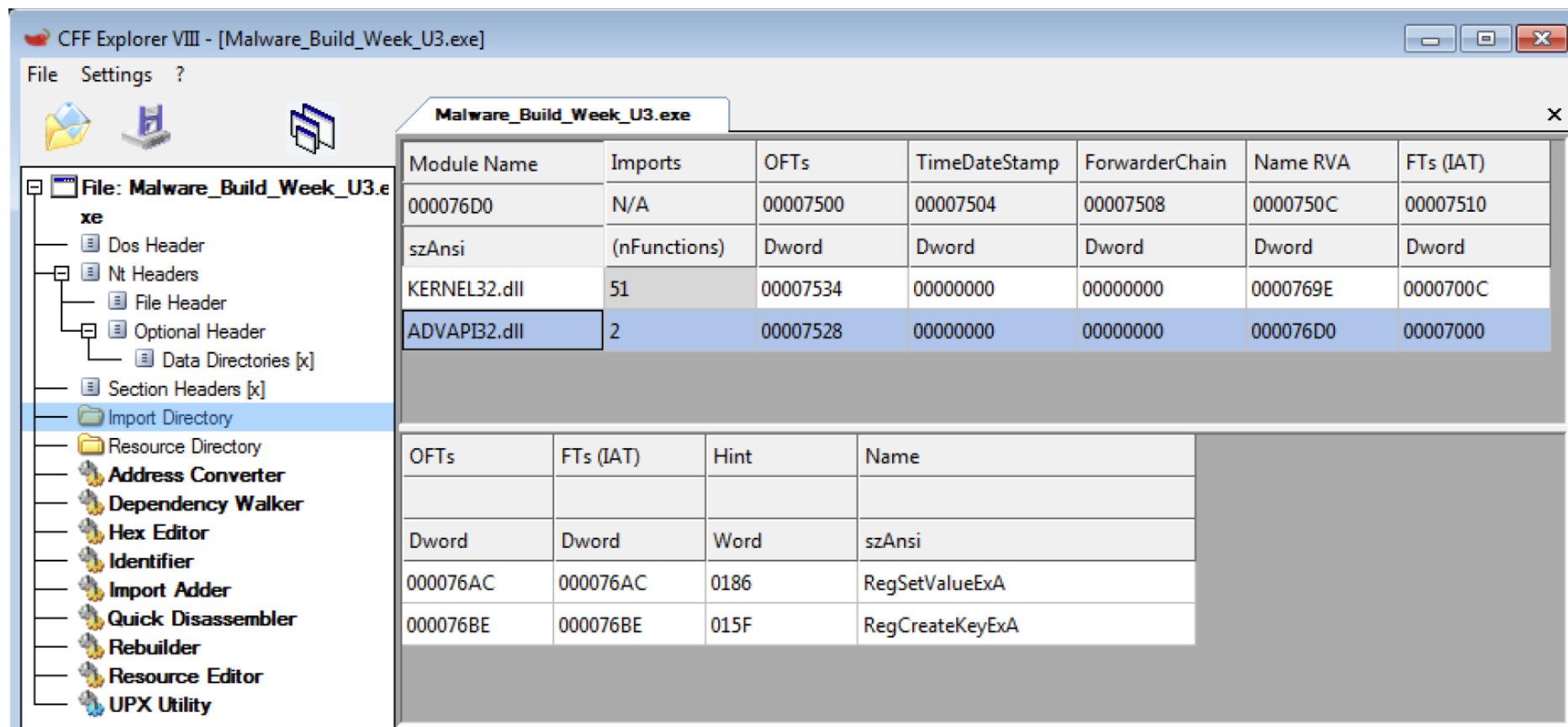
.text: Questa sezione contiene il codice eseguibile, ovvero le istruzioni che la CPU eseguirà una volta avviato il software.

.rdata: Include informazioni sulle librerie e le funzioni che sono importate ed esportate dall'eseguibile.

.data: Questa sezione memorizza i dati e le variabili globali del programma, che devono essere accessibili in qualsiasi punto del programma.

.rsrc: Contiene le risorse utilizzate dall'eseguibile, come icone, immagini e stringhe di testo che non fanno parte del codice eseguibile stesso.

3. Quali librerie importa il Malware? Per ognuna delle librerie importate, fate delle ipotesi



Il malware importa due librerie cruciali per il funzionamento del sistema operativo: **KERNEL32.dll** e **ADVAPI32.dll**. Facendo riferimento al materiale fornito da Epicode, possiamo spiegare le principali funzioni di queste librerie:

- **Kernel32.dll:** Questa libreria è una delle più comuni e contiene un'ampia gamma di funzioni essenziali per l'interazione con il sistema operativo. Tra le principali funzioni offerte troviamo:

- **Manipolazione dei file:** Funzioni per la creazione, apertura, lettura, scrittura e chiusura dei file.
- **Gestione della memoria:** Funzioni per allocare, deallocare e gestire la memoria.
- **Gestione dei processi e dei thread:** Funzioni per la creazione e la gestione di processi e thread.
- **Gestione delle operazioni di I/O:** Funzioni per gestire input e output di dati.
- **Advapi32.dll:** Questa libreria contiene funzioni specifiche per interagire con i servizi e i registri del sistema operativo Microsoft. Alcune delle funzioni chiave includono:
 - **Gestione dei servizi:** Funzioni per creare, configurare e gestire i servizi di Windows.
 - **Accesso al registro di sistema:** Funzioni per creare, aprire, leggere, scrivere e cancellare chiavi e valori del registro di sistema.
 - **Sicurezza e controllo degli accessi:** Funzioni per gestire i permessi e le policy di sicurezza.

In particolare, nel contesto di questo malware, la libreria **Advapi32.dll** richiama due funzioni specifiche del registro di sistema:

- **RegCreateKeyExA:** Questa funzione viene utilizzata per creare una nuova chiave del registro di sistema. Se la chiave specificata esiste già, la funzione la apre invece di crearne una nuova. Questa operazione è fondamentale per molte applicazioni che necessitano di salvare configurazioni e impostazioni nel registro.
- **RegSetValueExA:** Questa funzione imposta i dati e il tipo di un valore specificato all'interno di una chiave del registro di sistema. È utilizzata per modificare i valori esistenti o per crearne di nuovi, aggiornando così le informazioni necessarie per l'esecuzione corretta dell'applicazione.

L'uso di queste funzioni suggerisce che il malware potrebbe tentare di scrivere o modificare il registro di Windows. Una delle possibilità più comuni per cui un malware modifica il registro di sistema è per garantirsi la persistenza. Ad esempio, può aggiungere valori che permettono l'avvio automatico del malware all'accensione del computer, assicurandosi così che venga eseguito ogni volta che il sistema si avvia.

Questo tipo di operazioni è indicativo di un malware che cerca di stabilire una presenza duratura nel sistema, rendendo più difficile la sua rimozione e aumentando le possibilità di eseguire attività malevole nel tempo. La modifica del registro per ottenere la persistenza è

una tecnica comune tra i malware, poiché il registro di sistema di Windows è un punto centrale per la configurazione e il comportamento del sistema operativo.

Malware Analysis

Con riferimento al Malware in analisi, spiegare:

Lo scopo della funzione chiamata alla locazione di memoria 00401021

Come vengono passati i parametri alla funzione alla locazione 00401021;

Che oggetto rappresenta il parametro alla locazione 00401017

Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.

Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito C. Valutate ora la chiamata alla locazione 00401047, qual è il valore del parametro «ValueName»?

.Lo scopo della funzione chiamata alla locazione di memoria 00401021

Come vengono passati i parametri alla funzione alla locazione 00401021;


```

push    0                ; lpClass
push    0                ; Reserved
push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
push    80000002h        ; hKey
call     ds:RegCreateKeyExA
test     eax, eax
jz       short loc_401032

```

```

.text:00401015      push    0                ; Reserved
.text:00401017      push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
.text:0040101C      push    80000002h        ; hKey
.text:00401021      call     ds:RegCreateKeyExA
.text:00401027      test     eax, eax
.text:00401029      jz       short loc_401032
.text:0040102B      mov      eax, 1
.text:00401030      jmp      short loc_40107B
.text:00401032      ; -----
.text:00401032      loc_401032:
.text:00401032      mov      ecx, [ebp+cbData] ; CODE XREF: sub_401000+29↑j
.text:00401035      push     ecx              ; cbData
.text:00401036      mov      edx, [ebp+lpData]
.text:00401039      push     edx              ; lpData
.text:0040103A      push     1                ; dwType

```

```

; LSTATUS __stdcall RegCreateKeyExA(HKEY hKey, LPCSTR lpSubKey, DWORD Reserved, LPSTR lpClass, DWORD dwOptions, REGSAM samDesired, const LPSECURITY_ATTRIBUTES lpSecurityAttributes, HKEY* pNewKey)
extrn RegCreateKeyExA:dword ; CODE XREF: sub_401000+21↑p

```

Nella locazione di memoria 00401021 si trova un'istruzione call, utilizzata per chiamare una funzione. In questo caso, la funzione chiamata è RegCreateKeyExA, che permette di creare o aprire una chiave di registro esistente. I parametri necessari per questa funzione vengono passati tramite l'istruzione push sullo stack. Se la sottochiave (subkey) specificata esiste già, la funzione la apre; altrimenti, ne crea una nuova.

.Che oggetto rappresenta il parametro alla locazione 00401017

```
push    0                ; lpClass
push    0                ; Reserved
push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
push    80000002h        ; hKey
call    ds:RegCreateKeyExA
```

rappresenta l'offset della stringa ""SubKey". La subkey è un valore/sottocartella contenuto in una chiave di registro.

.Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.

• .text:0040101C	push	80000002h ; hKey
• .text:00401021	call	ds:RegCreateKeyExA
• .text:00401027	test	eax, eax
• .text:00401029	jz	short loc_401032
• .text:0040102B	mov	eax, 1
• .text:00401030	jmp	short loc_40107B
• .text:00401032 ;		

Le istruzioni sono una test ed un jz. La test, è molto simile a un'istruzione AND ma rispetto ad essa non modifica il valore contenuto negli operandi. Nel nostro caso la test è sullo stesso registro ... test EAX, EAX. In questo possiamo affermare che la test viene utilizzata per controllare se un valore è zero. In caso di riscontro lo ZF (Zero Flag) viene settato ad 1 e quindi viene poi eseguita l'istruzione successiva ... jz ShortLoc... Questo salto condizionale viene effettuato se lo Zero Flag è a 1 (Jump Zero).

Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito C. Valutate ora la chiamata alla locazione 00401047, qual è il valore del parametro «ValueName»?

Traduzione delle istruzioni alla domanda precedente in linguaggio C

Test eax, eax ➡ If (var == 0)

Jz short loc_401032 ➡ esegui codice contenuto tra le parentesi graffe

If(var==0){ Istruzioni

}



Valutate ora la chiamata alla locazione 00401047, qual è il valore del parametro «ValueName»?

• .text:0040103C	push	0	; Reserved
• .text:0040103E	push	offset ValueName	; "GinaDLL"
• .text:00401043	mov	eax, [ebp+hObject]	
• .text:00401046	push	eax	; hKey
• .text:00401047	call	ds:RegSetValueExA	

Il valore di ValueName è la stringa "GinaDLL" che viene quindi passata come parametro alla funzione RegSetValueExA.

Analisi Dinamica

Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware? Spiegate cosa è avvenuto, unendo le evidenze che avete raccolto finora per rispondere alla domanda

Nome	Ultima modifica	Tipo	Dimensione
 Malware_Build_Week_U3	17/01/2024 17:48	Applicazione	52 KB
 msgina32.dll	21/04/2024 16:03	Estensione dell'ap...	7 KB

viene creata la libreria **msgina32.dll**. "GinaDLL" viene caricato dal sistema in HKLM per poi essere passata come parametro alla funzione RegSetValueExA per modificarne il valore o sostituirlo.

-Quale chiave di registro viene creata?

-Quale valore viene associato alla chiave di registro creata?

00:12:...	Malware_Build_...	2728	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
00:12:...	Malware_Build_...	2728	RegQueryValue	HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\Image File Execution Options\DisableUserModeCallbackFilter
00:12:...	Malware_Build_...	2728	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager
00:12:...	Malware_Build_...	2728	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager
00:12:...	Malware_Build_...	2728	RegQueryValue	HKLM\System\CurrentControlSet\Control\SESSION MANAGER\CWDIllegalInDLLSearch
00:12:...	Malware_Build_...	2728	RegCloseKey	HKLM\System\CurrentControlSet\Control\SESSION MANAGER
00:12:...	Malware_Build_...	2728	RegOpenKey	HKLM\System\CurrentControlSet\Control\hivelist
00:12:...	Malware_Build_...	2728	RegOpenKey	HKLM\System\CurrentControlSet\Control\hivelist
00:12:...	Malware_Build_...	2728	RegQueryValue	HKLM\System\CurrentControlSet\Control\hivelist\Registry\User\S-1-5-21-3771313050-58705377-3452663501-1001_Classes
00:12:...	Malware_Build_...	2728	RegCloseKey	HKLM\System\CurrentControlSet\Control\hivelist
00:12:...	Malware_Build_...	2728	RegOpenKey	HKLM\SOFTWARE\Microsoft\WOW64
00:12:...	Malware_Build_...	2728	RegOpenKey	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
00:12:...	Malware_Build_...	2728	RegSetInfoKey	HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\Image File Execution Options
00:12:...	Malware_Build_...	2728	RegQueryValue	HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\Image File Execution Options\DisableUserModeCallbackFilter
00:12:...	Malware_Build_...	2728	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager
00:12:...	Malware_Build_...	2728	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager
00:12:...	Malware_Build_...	2728	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\SESSION MANAGER
00:12:...	Malware_Build_...	2728	RegQueryValue	HKLM\System\CurrentControlSet\Control\SESSION MANAGER\CWDIllegalInDLLSearch
00:12:...	Malware_Build_...	2728	RegCloseKey	HKLM\System\CurrentControlSet\Control\SESSION MANAGER
00:12:...	Malware_Build_...	2728	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server

PID	Operation	Path	Result	Detail
Id_...	3032	RegQueryValue	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions\Default	SUCCESS
Id_...	3032	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	REPARSE
Id_...	3032	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	Desired Access: Read
Id_...	3032	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\Terminal Server	Desired Access: Read
Id_...	3032	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\Terminal Server	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0
Id_...	3032	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	NAME NOT FOUND Length: 548
Id_...	3032	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSUserEnabled	SUCCESS
Id_...	3032	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	Type: REG_DWORD, Length: 4, Data: 0
Id_...	3032	RegOpenKey	HKLM	SUCCESS
Id_...	3032	RegOpenKey	HKLM	Desired Access: Maximum Allowed, Granted Access: All Access
Id_...	3032	RegOpenKey	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Diagnostics	Query: HandleTags, HandleTags: 0x0
Id_...	3032	RegOpenKey	HKLM	NAME NOT FOUND Desired Access: Read
Id_...	3032	RegQueryKey	HKLM	Query: HandleTags, HandleTags: 0x0
Id_...	3032	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
Id_...	3032	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	Desired Access: All Access, Disposition: REG_OPENED_EXISTING
Id_...	3032	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0
Id_...	3032	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	Query: HandleTags, HandleTags: 0x400
Id_...	3032	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
Id_...	3032	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	Type: REG_SZ, Length: 520, Data: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Id_...	3032	RegCloseKey	HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\Image File Execution Options	SUCCESS
Id_...	3032	RegCloseKey	HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\Image File Execution Options	SUCCESS
Id_...	3032	RegCloseKey	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions	SUCCESS
Id_...	3032	RegCloseKey	HKLM	SUCCESS

Query: HandleTags, HandleTags: 0x400

SUCCESS

Type: REG_SZ, Length: 520, Data: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll

SUCCESS

Da una analisi su Procmon possiamo notare che viene creata la chiave **HKEY_LOCAL_MACHINE (HKLM)**: dove sono contenuti i record e le configurazioni della macchina

Con riguardo al file creato nella cartella: msgina32.dll. Il parametro passato è GinaDLL (Graphical Identification and Authentication Dynamic Link Library) un importante componente per il processo di avvio di Windows.

La funzione API del parametro è **RegSetValue** che viene utilizzata per impostare il valore di una chiave di registro, in questo caso notiamo che il path porta alla libreria creata in precedenza Gina.dll.

-Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware?

3032	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
3032	RegOpenKey	HKLM	SUCCESS	Desired Access: Maximum Allowed, Granted Access: All Access
3032	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
3032	RegOpenKey	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Diagnostics	NAME NOT FOUND	Desired Access: Read
3032	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: Over...
3032	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS	Offset: 0, Length: 4,096, Priority: Normal
3032	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS	Offset: 4,096, Length: 2,560, Priority: Normal
3032	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS	
3032	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
3032	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All Access, Disposition: REG_OPENED_EXISTIN...
3032	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0

Con la funzione **CreateFile** possiamo notare che è stato generato un file.dll quindi una libreria all'interno della cartella del malware. In questo caso msgina32.dll.

-Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware.

Dalle analisi effettuate, possiamo dedurre che si tratta di un programma malevolo che contiene al suo interno un malware. Analizzando le librerie importate e le modifiche alle chiavi di registro, possiamo ipotizzare che si tratti di un Dropper, un tipo di malware progettato per essere salvato su disco ed eseguito in futuro (ad esempio, all'avvio del sistema operativo).

Un elemento chiave della nostra analisi è la modifica, o probabile sostituzione, della libreria msgina32.dll. Questa libreria gestisce il processo di autenticazione degli utenti ed è chiamata durante l'avvio del sistema operativo per fornire l'interfaccia utente necessaria per l'identificazione. Sostituendo il valore di msgina32.dll, il malware garantisce la propria persistenza nel sistema, poiché verrà avviato ogni volta che il sistema operativo viene riavviato. Questo metodo permette al malware di essere eseguito automaticamente durante il processo di accesso, sfruttando la funzione critica di msgina32.dll.