W24D4 PROGETTO

Il nostro obiettivo sarà analizzare adeguatamente il malware contenuto nella Build_Week_Unit_3. La traccia porta con se i seguenti punti.

Analisi statica

Con riferimento al file eseguibile Malware_Build_Week_U3, rispondere ai seguenti quesiti utilizzando i tool e le tecniche apprese nelle lezioni teoriche:

- Quanti parametri sono passati alla funzione Main()?

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

parametri passati

3 parametri in tutto vengono passati: l'intero argc, e le due caratteri costanti **argv e **envp.

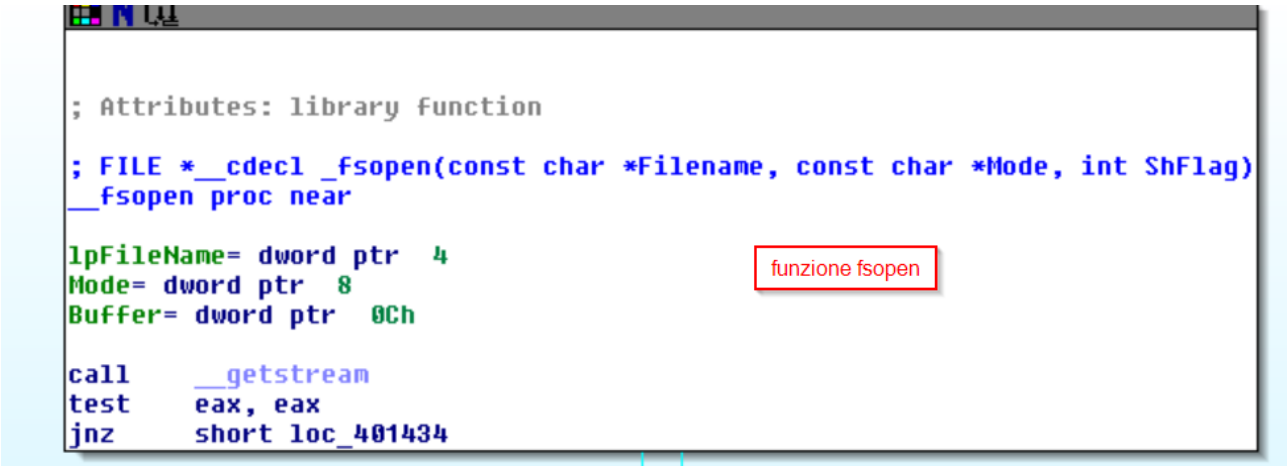
- Quante variabili sono dichiarate all'interno della funzione Main()?

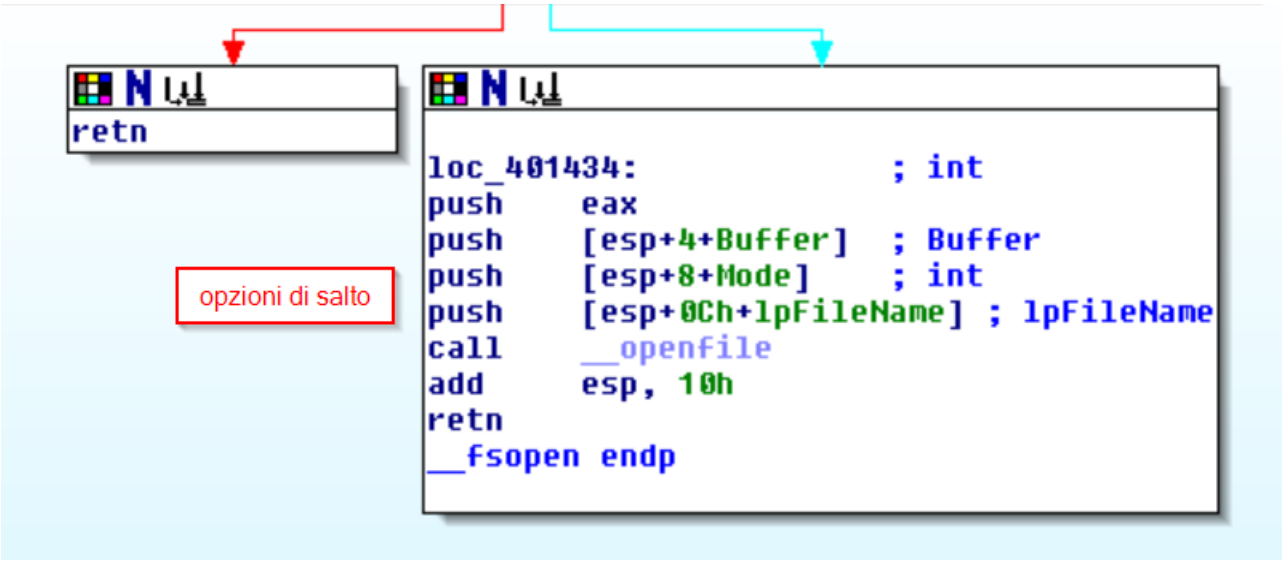
```
hModule= dword ptr -11Ch
Data= byte ptr -118h
var_117= byte ptr -117h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h
```

Variabili dichiarate

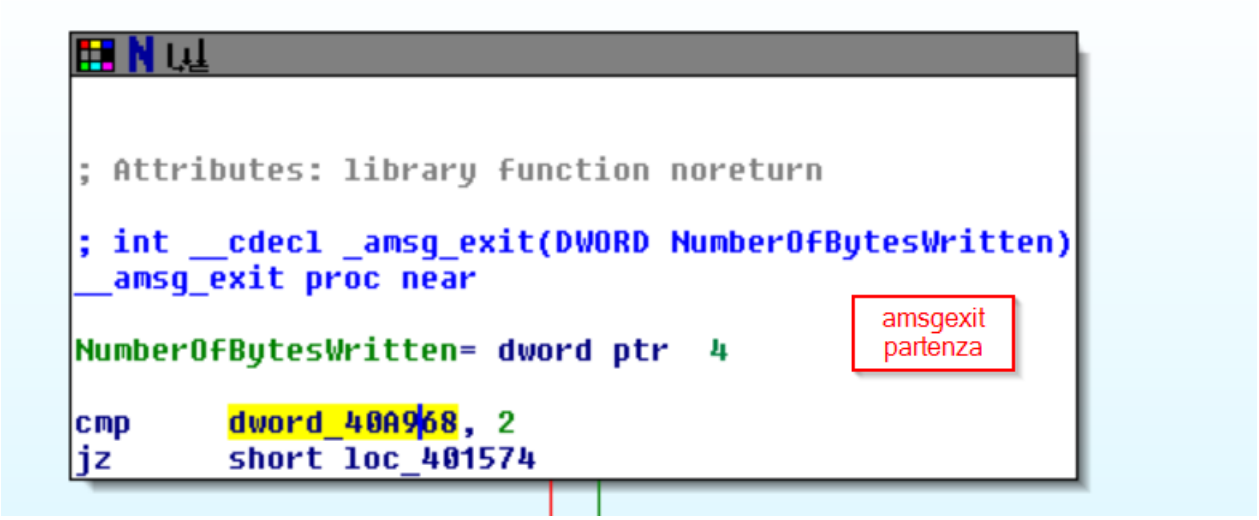
Le variabili sottostanti l'inizio della funzione main riportano, oltre ai parametri passati, hModule, Data, var_117, var_8 e var_4

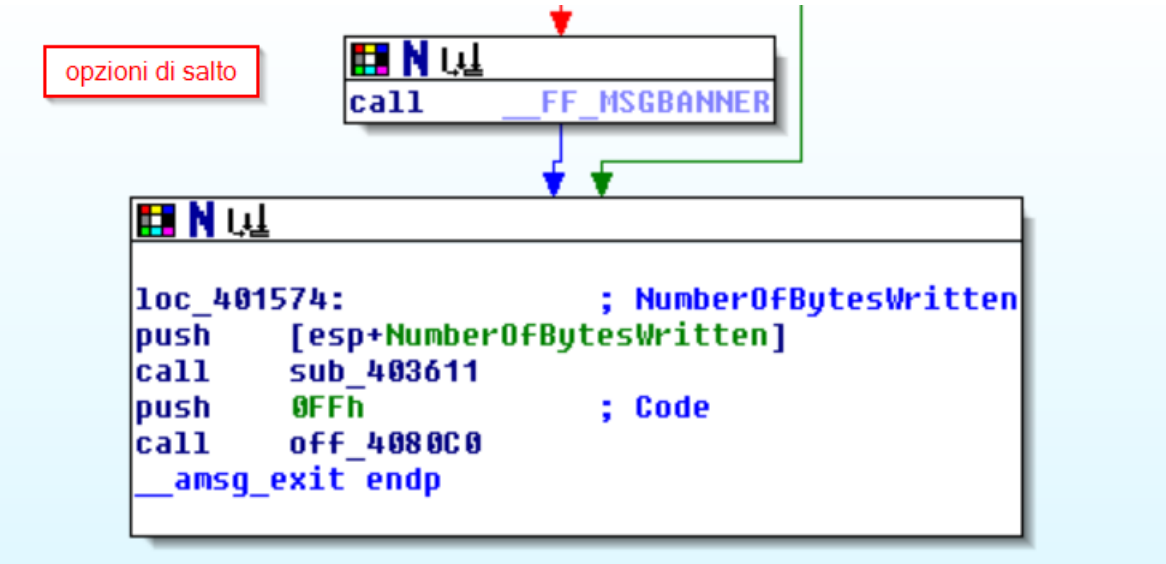
- Quali sezioni sono presenti all'interno del file eseguibile? descrivete brevemente almeno 2 di quelle identificate.





La funzione fopen sembra essere una wrapper function per aprire un file in stream (lettura o scrittura). LpFileName è un puntatore a una stringa che rappresenta il nome del file. Mode specifica la modalità di apertura, mentre Buffer serve a memorizzare il file aperto. La funzione getstream verifica il valore restiuito in eax. Se eax è diverso da zero, verrà effettuato un salto alla locazione 401434, altrimenti viene eseguito un ritorno.

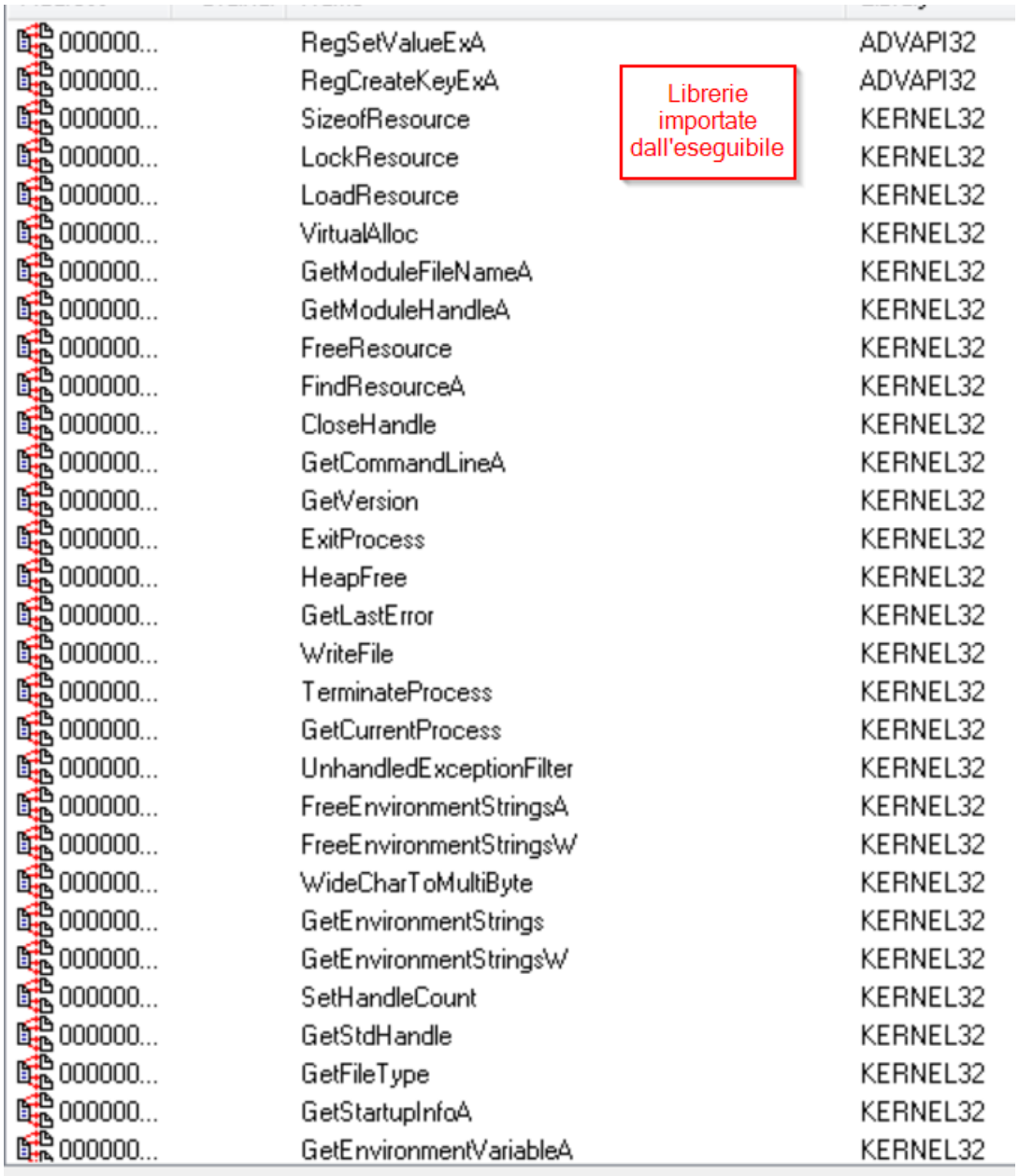




_amsg_exit è una funzione della libreria di runtime C/C++. Viene chiamata quando si verifica un errore di runtime all'interno di un programma. Il suo compito principale è generare un messaggio di errore e terminare l'applicazione.

La funzione parte con la verifica di di word_40A968, una variabile globale. Se dword_40A968 è uguale a 2, salta a loc_401574, altrimenti, chiama FF_MSGBANNER.

- 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.



Le funzioni nella lista “imports” sono state importate dalle librerie kernel32 e advapi32.

La prima libreria include funzioni dedite all’interazione con i servizi e i registri. RegSetValueExA, ad esempio, imposta dati e tipo di un valore all’interno di una chiave di registro del sistema, mentre RegCreateKeyExA si occupa di creare la chiave specificata, a patto che quest’ultima non esista di già, in quel caso viene aperta. Per mezzo di queste funzioni, il malware potrebbe compiere diverse azioni dannose:

- Il malware potrebbe utilizzare RegSetValueExA per creare o modificare valori all'interno delle chiavi di registro di avvio automatico, come HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run. In questo modo otterrebbe la persistenza.
- Potrebbe modificare il comportamento del sistema e altre applicazioni, come ad esempio il browser predefinito, esso potrebbe indirizzare l'utente verso siti web malevoli o pubblicitari.
- Potrebbe utilizzare i valori modificati/creati per occultare la propria presenza o negare il corretto funzionamento degli strumenti di sicurezza.

  La seconda libreria porta con sé funzioni per modificare i file e gestire la memoria. Tra le varie funzioni importate, prendiamo per esempio la tripletta SizeOfResource, LockOfResource e LoadOfResource:

- SizeOfResource: Restituisce la dimensione di una risorsa specificata. È possibile utilizzarla per determinare quanto spazio di memoria è necessario per archiviare una risorsa.

- LockOfResource: Restituisce un puntatore al primo byte dei dati associati a una risorsa. Dopo aver caricato una risorsa con LoadResource, è possibile utilizzare LockResource per accedere ai dati della risorsa.

- LoadResource: La funzione LoadResource recupera un handle che può essere utilizzato per ottenere un puntatore al primo byte della risorsa specificata in memoria.

  È possibile utilizzarla per caricare una risorsa da un modulo (file eseguibile) e ottenere un handle per i dati associati alla risorsa.

LoadResource restituisce un handle di tipo HGLOBAL, ma non deve essere passato alle funzioni GlobalLock o GlobalFree. Per ottenere un puntatore ai dati della risorsa, è necessario chiamare LockResource. Per ottenere le dimensioni della risorsa, è possibile chiamare SizeofResource.

Ipotizzando dei possibili scenari, un malware, solo utilizzando queste funzioni potrebbe:

- Utilizzare LoadResource per accedere a risorse all'interno di un file eseguibile o di una libreria dinamica. Ad esempio, potrebbe cercare di estrarre informazioni sensibili come chiavi di crittografia, password o dati di configurazione.
- Utilizzando LockResource, un malware potrebbe ottenere un puntatore ai dati di una risorsa. Successivamente, potrebbe sovrascrivere i dati con codice malevolo e iniettarlo nel processo in esecuzione. Questo potrebbe portare a comportamenti imprevisti o dannosi nel programma.
- Un malware potrebbe utilizzare SizeOfResource per determinare la dimensione di una risorsa. Successivamente, potrebbe allocare una grande quantità di memoria per risorse, esaurire le risorse del sistema e causare un DoS.

Con riferimento al Malware in analisi, spiegare:

- Lo scopo della funzione chiamata alla locazione di memoria 00401021:

La funzione RegCreateKeyExA crea tutte le chiavi mancanti nel percorso specificato, oppure le apre nel caso esistano già. Come specificato prima.

- Come vengono passati i parametri alla funzione alla locazione 00401021:

- push ebp: Questa istruzione mette il valore corrente del registro ebp nello stack.

- mov ebp, esp: Questa istruzione copia il valore dello stack pointer (esp) nel registro base del frame (ebp). In altre parole, ebp ora punta alla posizione corrente dello stack.

- push ecx: Mette il valore del registro ecx nello stack. Questo potrebbe essere utilizzato per salvare temporaneamente il valore di ecx prima di modificarlo all'interno della funzione.

- push 0: Mette il valore zero nello stack, corrispondente al parametro lpdwDisposition.

- lea eax, [ebp+hObject]: Calcola l'indirizzo della variabile hObject all'interno del frame dello stack e lo mette nel registro eax.

- push eax: Mette il valore di eax (che è l'indirizzo di hObject) nello stack, corrispondente al parametro phkResult.

- push 0: Mette il valore zero nello stack, corrispondente al parametro lpSecurityAttributes.

- push 0F003Fh: Mette il valore esadecimale 0F003Fh nello stack, corrispondente al parametro samDesired.

- push 0: Mette il valore zero nello stack, corrispondente al parametro dwOptions.

- push 0: Ancora un altro valore zero nello stack, corrispondente al parametro lpClass.

- push offset SubKey: Mette l'indirizzo della stringa "SOFTWARE\Microsoft\Windows NT\CurrentVersion" nello stack.

- push 80000002h: Mette il valore esadecimale 80000002h nello stack, corrispondente al parametro hkey.

- call ds:RegCreateKeyExA: Chiama la funzione RegCreateKeyExA. parametri passati alla funzione sono quelli che abbiamo messo nello stack in precedenza.

- Che oggetto rappresenta il parametro alla locazione 00401017, Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.

il parametro alla locazione 00401017, ovvero "SOFTWARE\Microsoft\Windows NT\CurrentVersion" si riferisce al registro di sistema di Windows. Questo percorso contiene informazioni sulla versione attuale di Windows NT installata sul sistema, tra cui informa zioni di configurazione e altre impostazioni di sistema rilevanti.

Le istruzioni tra gli indirizzi 00401027 e 00401029 sono le seguenti:

test eax,eax, questa istruzione effettua l'AND logico tra il registro eax e se stesso. Se il risultato è zero, imposta lo flag di zero a 1 (ZF), altrimenti lo azzererà.

jz short loc_401032, questa istruzione effettua un salto alla locazione 401032 se lo zero flag corrisponde ad una, altrimenti si procede con le istruzione successiva.

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

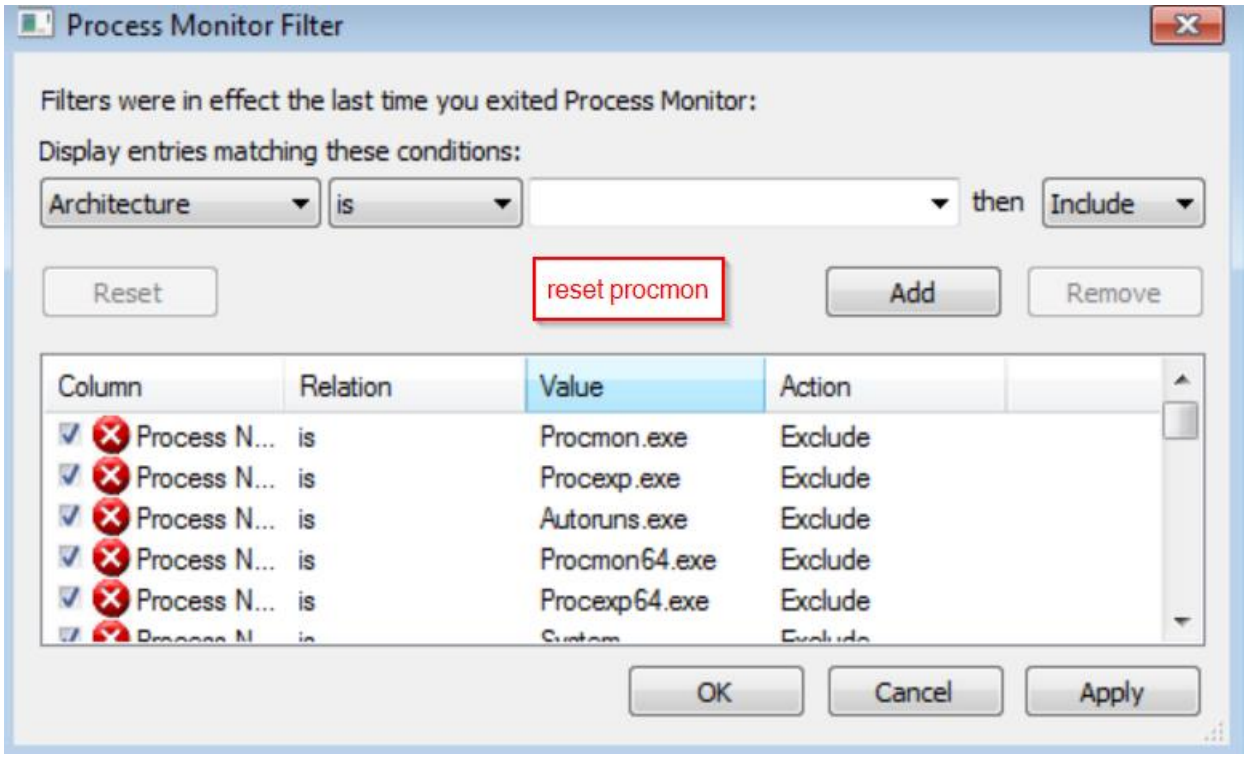
- Traduzione in C

```
if (eax == 0) {
goto loc_401032; // permette di saltare ad un etichetta specifica all'interno della funzione
}
```

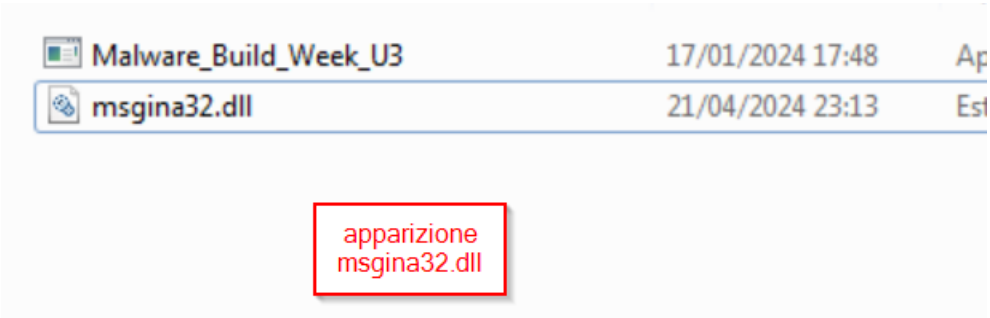
- Il valore del parametro ValueName, che verrà passato alla funzione RegSetValueExA è "GinaDLL".

Analisi dinamica

Preparate l'ambiente ed i tool per l'esecuzione del Malware (suggerimento: avviate principalmente Process Monitor ed assicurate di eliminare ogni filtro cliccando sul tasto «reset» quando richiesto in fase di avvio). Eseguite il Malware, facendo doppio click sull'icona dell'eseguibile.



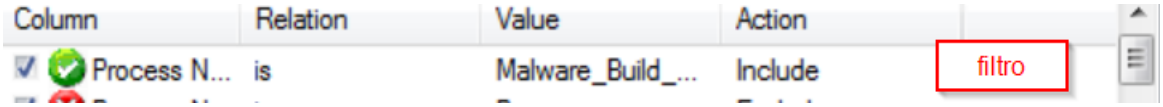
- Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware? Spiegate cosa è avvenuto, unendo le evidenze che avete raccolto finora per rispondere alla domanda

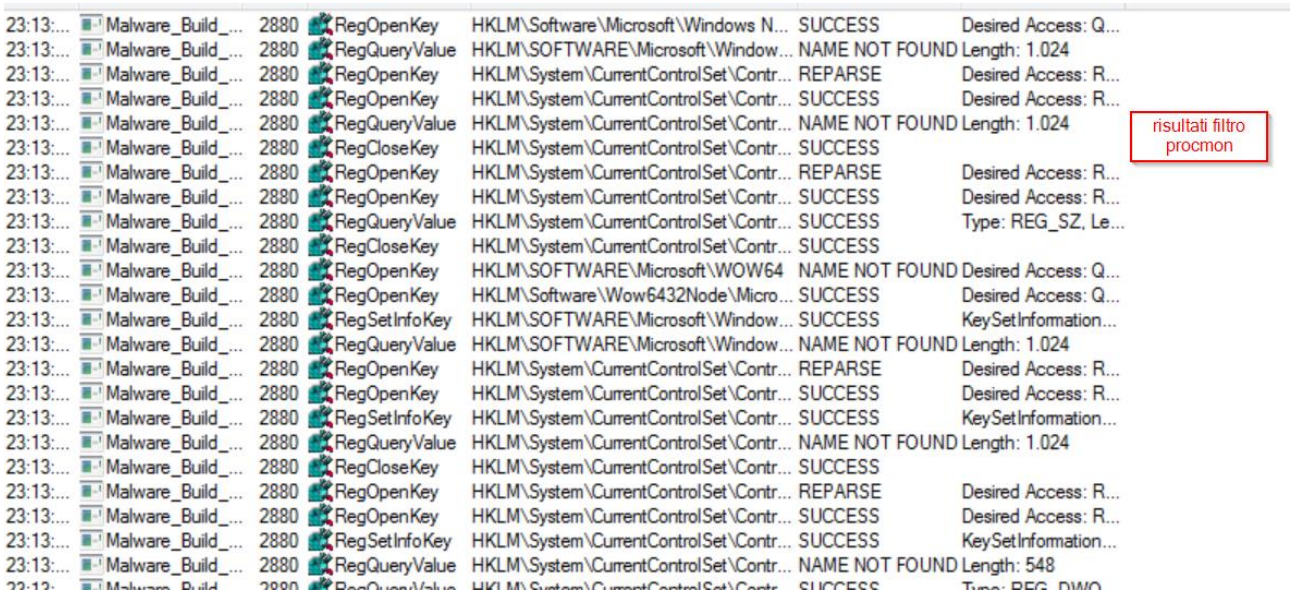


Nella cartella dove è situato l'eseguibile del malware dopo l'esecuzione, notiamo la presenza di un file chiamato "msgina32.dll". In base all'analisi statica precedente, possiamo dedurre che il malware ha utilizzato la funzione RegSetValueExA per scrivere nel registro di sistema il valore "GinaDLL", e successivamente ha creato un file DLL chiamato "msgina32.dll".

Questo suggerisce fortemente che il malware stia cercando di sostituire o aggiungere un componente di autenticazione GINA (Graphical Identification and Authentication) nel sistema.

- Analizzate ora i risultati di Process Monitor Filtrate includendo solamente l'attività sul registro di Windows.





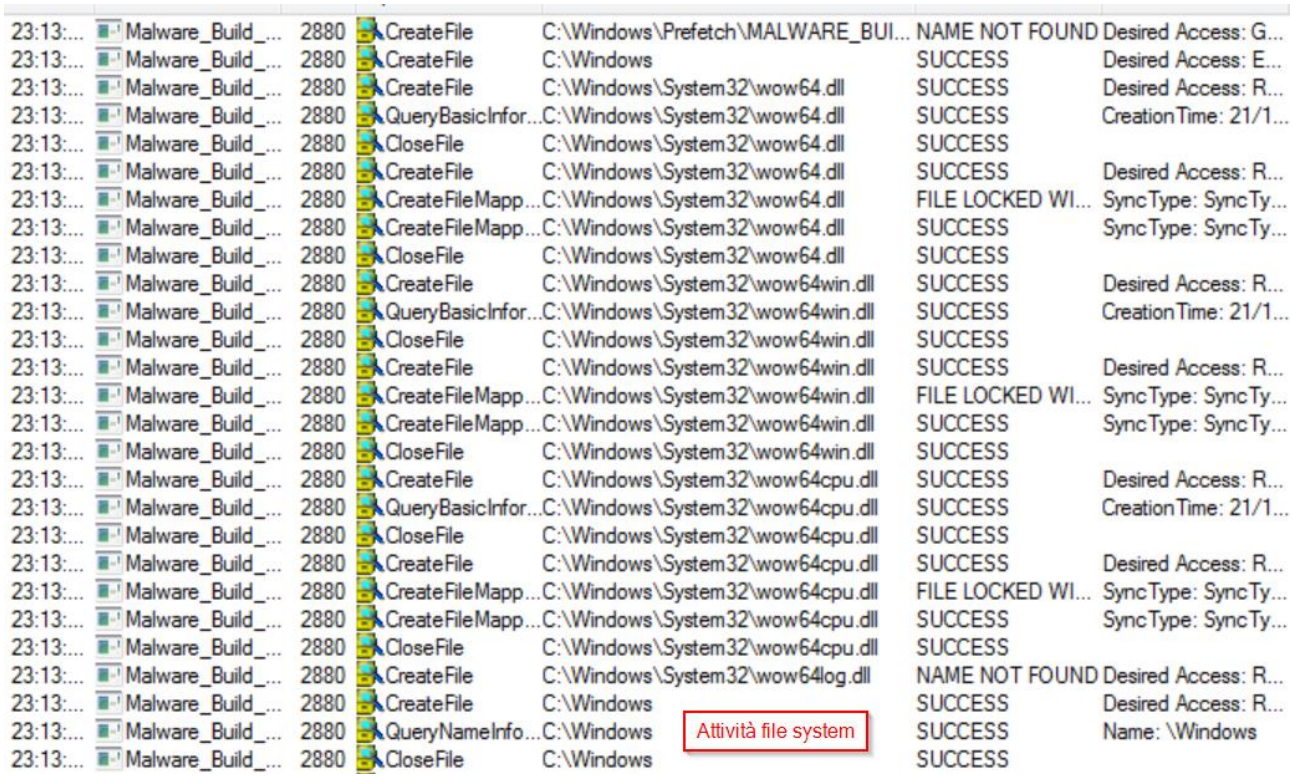
- Quale chiave di registro viene creata?

  msgina32.dll

- Quale valore viene associato alla chiave di registro creata?

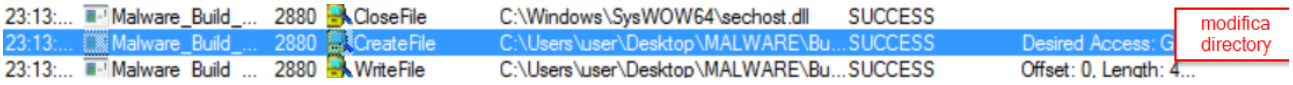
  GinaDLL

- Passate ora alla visualizzazione dell'attività sul file system.



23:13:...	Malware_Build_...	2880	CreateFile	C:\Windows\Prefetch\MALWARE_BUI...	NAME NOT FOUND	Desired Access: G...
23:13:...	Malware_Build_...	2880	CreateFile	C:\Windows	SUCCESS	Desired Access: E...
23:13:...	Malware_Build_...	2880	CreateFile	C:\Windows\System32\wow64.dll	SUCCESS	Desired Access: R...
23:13:...	Malware_Build_...	2880	QueryBasicInfor...	C:\Windows\System32\wow64.dll	SUCCESS	CreationTime: 21/1...
23:13:...	Malware_Build_...	2880	CloseFile	C:\Windows\System32\wow64.dll	SUCCESS	
23:13:...	Malware_Build_...	2880	CreateFile	C:\Windows\System32\wow64.dll	SUCCESS	Desired Access: R...
23:13:...	Malware_Build_...	2880	CreateFileMapp...	C:\Windows\System32\wow64.dll	FILE LOCKED WI...	SyncType: SyncTy...
23:13:...	Malware_Build_...	2880	CreateFileMapp...	C:\Windows\System32\wow64.dll	SUCCESS	SyncType: SyncTy...
23:13:...	Malware_Build_...	2880	CloseFile	C:\Windows\System32\wow64.dll	SUCCESS	
23:13:...	Malware_Build_...	2880	CreateFile	C:\Windows\System32\wow64win.dll	SUCCESS	Desired Access: R...
23:13:...	Malware_Build_...	2880	QueryBasicInfor...	C:\Windows\System32\wow64win.dll	SUCCESS	CreationTime: 21/1...
23:13:...	Malware_Build_...	2880	CloseFile	C:\Windows\System32\wow64win.dll	SUCCESS	
23:13:...	Malware_Build_...	2880	CreateFile	C:\Windows\System32\wow64win.dll	SUCCESS	Desired Access: R...
23:13:...	Malware_Build_...	2880	CreateFileMapp...	C:\Windows\System32\wow64win.dll	FILE LOCKED WI...	SyncType: SyncTy...
23:13:...	Malware_Build_...	2880	CreateFileMapp...	C:\Windows\System32\wow64win.dll	SUCCESS	SyncType: SyncTy...
23:13:...	Malware_Build_...	2880	CloseFile	C:\Windows\System32\wow64win.dll	SUCCESS	
23:13:...	Malware_Build_...	2880	CreateFile	C:\Windows\System32\wow64cpu.dll	SUCCESS	Desired Access: R...
23:13:...	Malware_Build_...	2880	QueryBasicInfor...	C:\Windows\System32\wow64cpu.dll	SUCCESS	CreationTime: 21/1...
23:13:...	Malware_Build_...	2880	CloseFile	C:\Windows\System32\wow64cpu.dll	SUCCESS	
23:13:...	Malware_Build_...	2880	CreateFile	C:\Windows\System32\wow64cpu.dll	SUCCESS	Desired Access: R...
23:13:...	Malware_Build_...	2880	CreateFileMapp...	C:\Windows\System32\wow64cpu.dll	FILE LOCKED WI...	SyncType: SyncTy...
23:13:...	Malware_Build_...	2880	CreateFileMapp...	C:\Windows\System32\wow64cpu.dll	SUCCESS	SyncType: SyncTy...
23:13:...	Malware_Build_...	2880	CloseFile	C:\Windows\System32\wow64cpu.dll	SUCCESS	
23:13:...	Malware_Build_...	2880	CreateFile	C:\Windows\System32\wow64log.dll	NAME NOT FOUND	Desired Access: R...
23:13:...	Malware_Build_...	2880	CreateFile	C:\Windows	SUCCESS	Desired Access: R...
23:13:...	Malware_Build_...	2880	QueryNameInfo...	C:\Windows	SUCCESS	Name: \Windows
23:13:...	Malware_Build_...	2880	CloseFile	C:\Windows	SUCCESS	

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



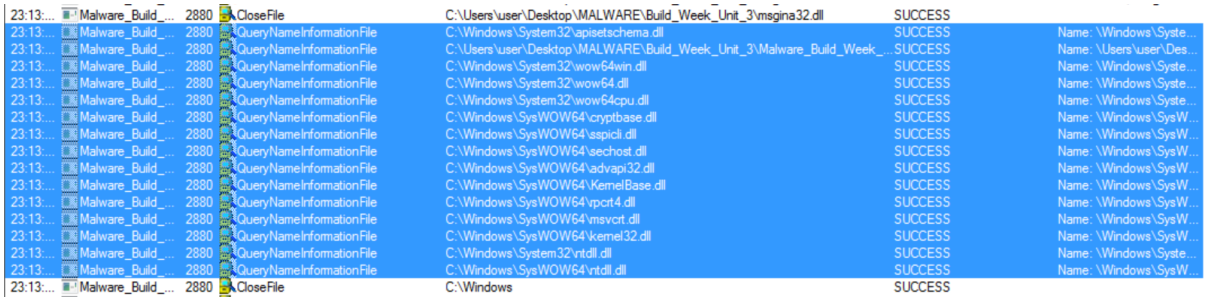
23:13:...	Malware_Build_...	2880	CloseFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	
23:13:...	Malware_Build_...	2880	CreateFile	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	Desired Access: G
23:13:...	Malware_Build_...	2880	WriteFile	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	Offset: 0, Length: 4...

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

Come abbiamo detto prima, il malware ha utilizzato la funzione RegSetValueExA per scrivere nel registro di sistema il valore "GinaDLL", e successivamente ha creato un file DLL chiamato "msgina32.dll".

Dopo di ciò, osservando l'analisi dinamica del malware, si notano diverse operazioni di tipo QueryNameInformationFile, che hanno raccolto informazioni riguardo numerose librerie del sistema operativo, ad

esempio kernel32.dll, usato per la manipolazione dei file e la gestione della memoria, advapi32.dll, utile per interagire con i servizi e i registri oppure apisetschema.dll, utilizzato per la gestione dei file di definizione delle api. Oltre a questo, il fatto che il malware abbia modificato e aperto le chiavi di registro del sistema operativo, basta e avanza per garantirgli la persistenza sul sistema.



23:13:...	Malware_Build_...	2880	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS	
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\System32\apisetschema.dll	SUCCESS	Name: \Windows\Syste...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\Malware_Build_Week_...	SUCCESS	Name: \Users\user\Des...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\System32\wow64win.dll	SUCCESS	Name: \Windows\Syste...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\System32\wow64.dll	SUCCESS	Name: \Windows\Syste...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\System32\wow64cpu.dll	SUCCESS	Name: \Windows\Syste...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\SysWOW64\cryptbase.dll	SUCCESS	Name: \Windows\SysW...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\SysWOW64\sspicli.dll	SUCCESS	Name: \Windows\SysW...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Name: \Windows\SysW...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\SysWOW64\advapi32.dll	SUCCESS	Name: \Windows\SysW...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\SysWOW64\KernelBase.dll	SUCCESS	Name: \Windows\SysW...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\SysWOW64\rpcrt4.dll	SUCCESS	Name: \Windows\SysW...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS	Name: \Windows\SysW...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\SysWOW64\kernel32.dll	SUCCESS	Name: \Windows\SysW...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\System32\ntdll.dll	SUCCESS	Name: \Windows\Syste...
23:13:...	Malware_Build_...	2880	QueryNameInformationFile	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Name: \Windows\SysW...
23:13:...	Malware_Build_...	2880	CloseFile	C:\Windows	SUCCESS	