

TEAM 5 BUILD-WEEK 3



TAVOLA DEI CONTENUTI

02
TAVOLA DEI
CONTENUTI

22
GIORNO 3

03
GIORNO 1

32
GIORNO 4

14
GIORNO 2

37
GIORNO 5

46
OUR TEAM

GIORNO 1

TRACCIA:

Il Malware da analizzare è nella cartella Build_Week_Unit_3 presente sul desktop della macchina virtuale dedicata.

Giorno 1: Con riferimento al file eseguibile Malware_Build_U3, rispondere ai seguenti quesiti utilizzando le tecniche apprese nelle lezioni teoriche:

- Quanti parametri sono passati alla funzione Main()?
- Quante variabili sono dichiarate all'interno della funzione?
- Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate.
- Quali librerie importa il Malware?

Per ognuna delle librerie importate, fate delle 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.

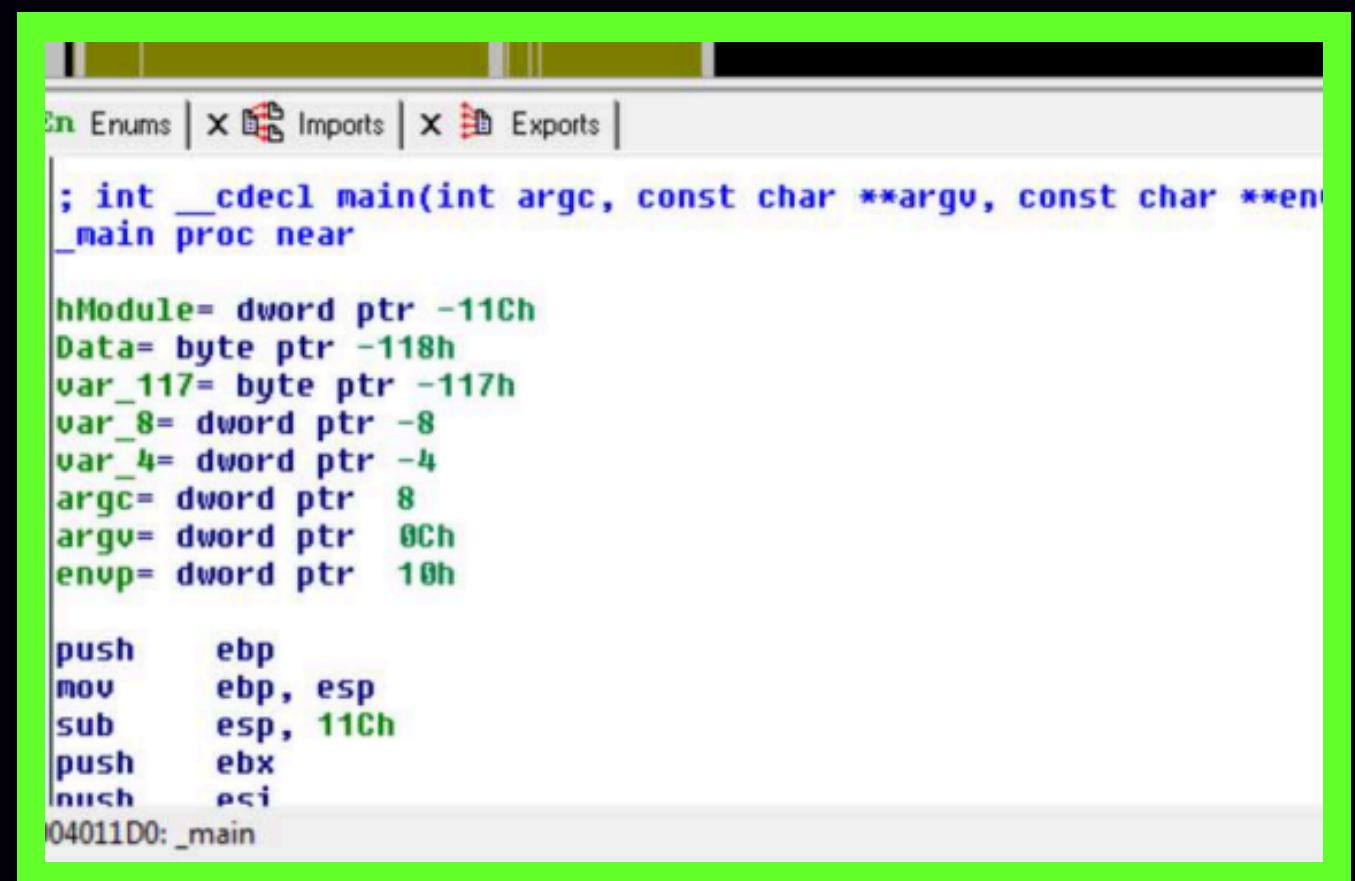


GIORNO 1

IDA (Interactive Disassembler) è un software ampiamente utilizzato per il disassemblaggio e il debugging.

Le **principali caratteristiche** di IDA includono:

- **Disassemblaggio:** IDA può disassemblare eseguibili binari in codice assembly, fornendo una rappresentazione leggibile dall'uomo del codice macchina. Rappresentazione Grafica: Offre un'interfaccia grafica per navigare attraverso il codice disassemblato, che aiuta a comprendere le strutture di programma complesse.
- **Analisi:** IDA esegue un'analisi statica del binario, identificando funzioni, variabili e strutture di flusso di controllo. Questo aiuta a comprendere la funzionalità del software.
- **Debugging:** Fornisce funzionalità di debugging, consentendo agli utenti di passare attraverso il codice disassemblato, impostare punti di interruzione, ispezionare la memoria e analizzare il comportamento in fase di esecuzione.
- **Supporto per lo Scripting:** IDA supporta lo scripting utilizzando vari linguaggi di programmazione come IDC (IDA C), Python e IDC Script. Ciò consente di automatizzare compiti e personalizzare il processo di analisi.
- **Ecosistema dei Plugin:** IDA può essere esteso attraverso plugin, che forniscono funzionalità aggiuntive adattate a specifici casi d'uso o compiti di analisi.



The screenshot shows the IDA Pro interface with a green border around the assembly window. The assembly code for the `_main` function is displayed:

```
; 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

push    ebp
mov     ebp, esp
sub    esp, 11Ch
push    ebx
push    esi
I04011D0: _main
```

GIORNO 1



Parametri e Variabili

argc: Indica il numero di entrate nell'array argv

argv: Un array di puntatori ad una stringa che contengono gli argomenti del programma

envp: Un array di puntatori ad una stringa che definisce l'ambiente del programma

La funzione main è fornita dall'user ed è dove l'esecuzione del programma ha inizio. La linea di comando al programma è suddivisa in sequenze di tokens o separate da blanks; e sono passate al main come un array di puntatori a stringhe/strings in argv.

Il numero di argomenti trovati passa al parametro argc. L'argomento argv è un puntatore ad un carattere string contenente il nome del programma. L'ultimo elemento dell'array puntato ad argv e' NULL. Gli argomenti contenenti blanks possono essere passati al main racchiudendoli in una citazione (che viene rimossa da quel elemento nel vettore argv).

L'argomento envp punta ad un array di puntatori a caratteri strings che sono l'ambiente delle stringhe per il corrente processo. Questo valore è identico alla variabile environ definita nel file header.

GIORNO 1

hModule	Questa è una variabile usata nel SO Windows per rappresentare un identificatore (handle) associato ad un modulo
Data	Si tratta di una variabile flessibile adatta per memorizzare dati di qualsiasi tipo senza specificare un formato particolare
var_117	Questa potrebbe rappresentare un'area di memoria usata per memorizzare dati temporanei o risultati non definitivi durante l'esecuzione di un programma
var_8	Questa è una variabile che può contenere diversi tipi di dati
var_4	Questa è una variabile che può contenere diversi tipi di dati

push ebx
push esi
push edi

hModule= dword ptr -11Ch
Data= byte ptr -118h
var_117= byte ptr -117h
var_8= dword ptr -8
var_4= dword ptr -4

GIORNO 1

Nella descrizione, "**ptr**" sta per puntatore, un concetto fondamentale in molti linguaggi di programmazione che consente di riferirsi direttamente all'indirizzo di memoria di una variabile o di un dato.

"**Dword**" e "**byte**" sono tipi di dato che rappresentano rispettivamente un double word, ovvero una parola di 32 bit, e un byte, che è una sequenza di 8 bit. Un "**dword**" può quindi contenere valori più grandi rispetto a un "**byte**" e viene spesso utilizzato per rappresentare interi o indirizzi di memoria più estesi.

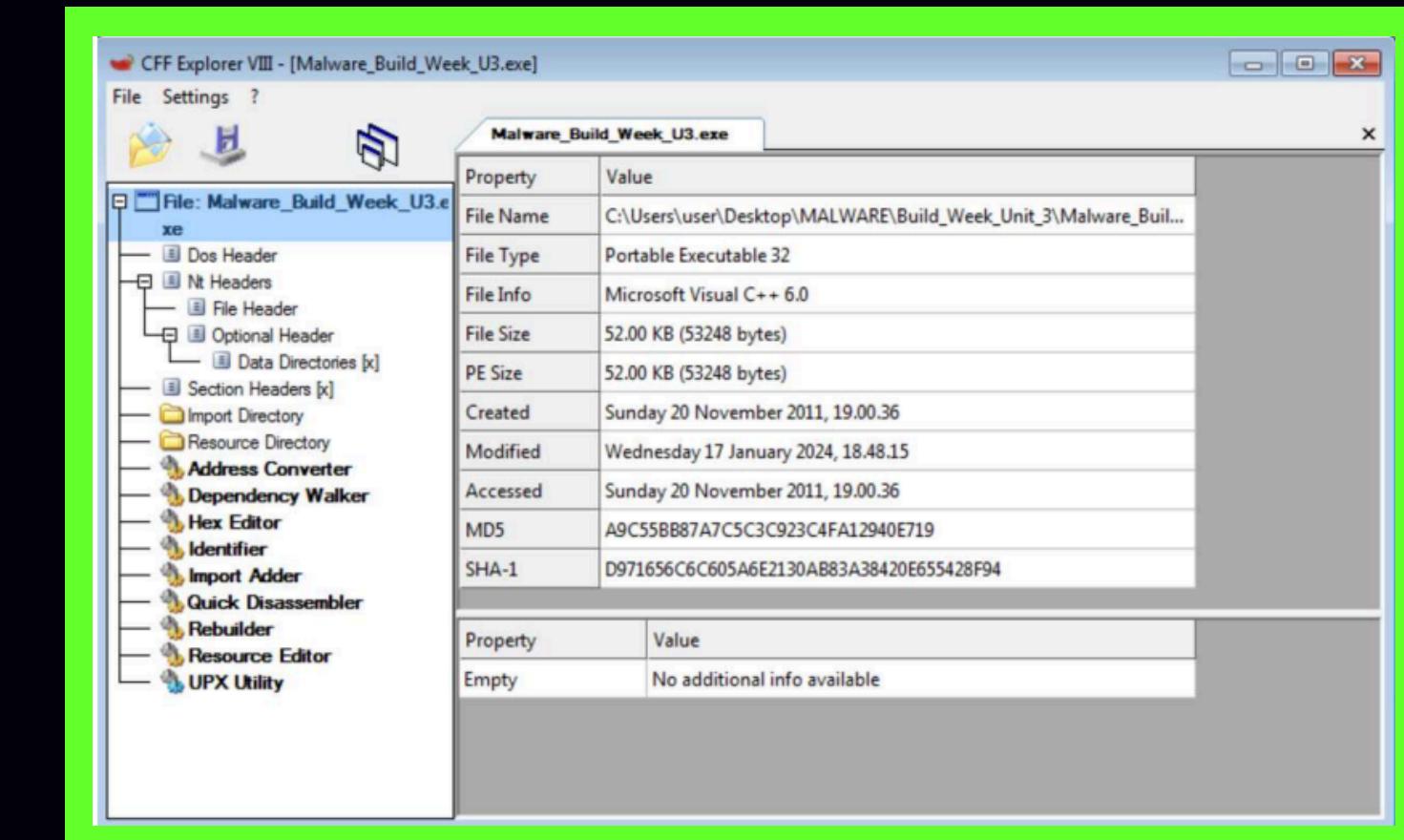
Gli offset negativi sono utilizzati per indicare che le variabili sono posizionate in memoria a una distanza negativa rispetto a un punto di riferimento, spesso il puntatore del frame di una funzione o un registro specifico. Questo tipo di rappresentazione è comune nei linguaggi di programmazione a basso livello, come l'assembly, dove è cruciale avere un controllo preciso sulla gestione della memoria.

In questo contesto, la gestione degli offset negativi è essenziale per l'organizzazione della stack frame di una funzione, dove le variabili locali e i parametri della funzione sono memorizzati a distanze specifiche dal puntatore del frame. Questa struttura consente un accesso rapido ed efficiente ai dati durante l'esecuzione di un programma, migliorando le prestazioni e la gestione delle risorse.

La comprensione di questi concetti è fondamentale per chi lavora con l'assembly o con linguaggi di programmazione a basso livello, poiché permette di scrivere codice più efficiente e ottimizzato, e di effettuare un debug approfondito dei programmi, identificando e risolvendo problemi legati alla gestione della memoria e alla manipolazione diretta degli indirizzi di memoria.

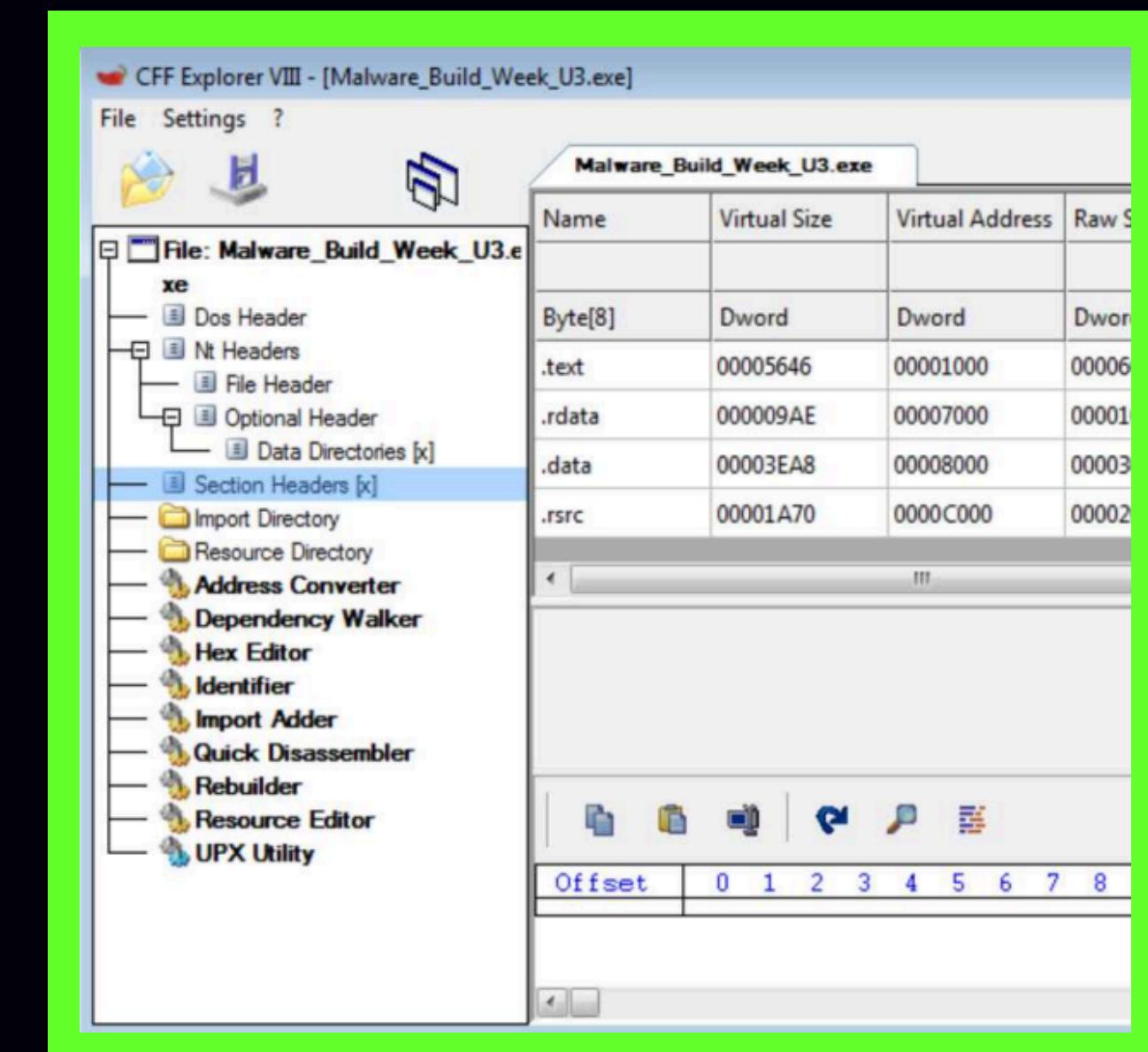
GIORNO 1

Utilizziamo **CFF Explorer** per l'analisi dei file eseguibili su Windows; possiamo trovare al suo interno varie funzionalità che ci permettono di analizzare le funzioni, le risorse, le librerie etc.



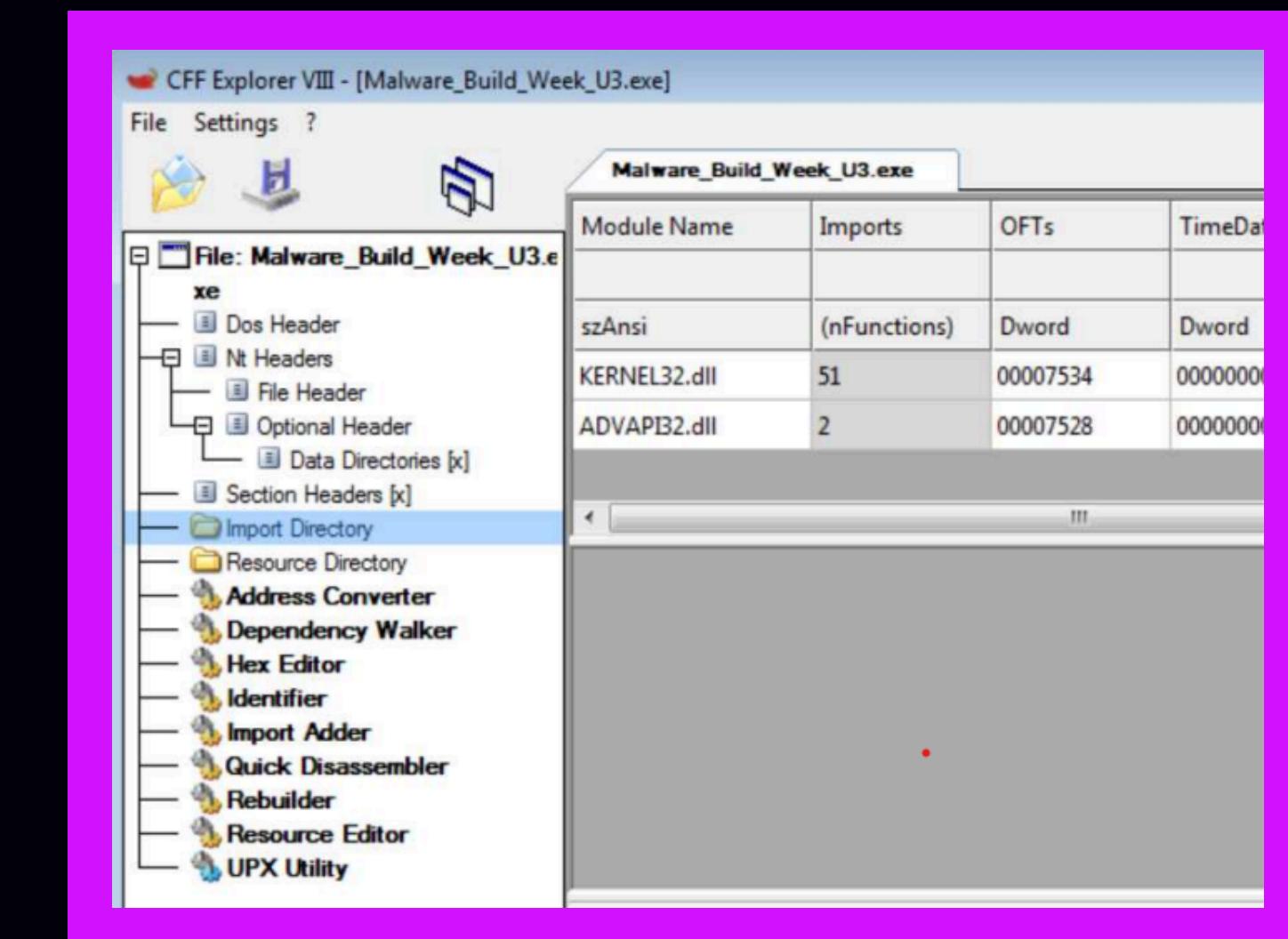
GIORNO 1

.text	Contiene le istruzioni che il processore eseguirà una volta che il software verrà avviato. Questa è, generalmente, l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto
.rdata	Include le informazioni circa le librerie e le funzioni importate ed esportate dal file eseguibile
.data	Contiene le variabili globali del programma eseguibile; si tratta di variabili non definite all'interno di un contesto di una funzione, ma bensì globalmente dichiarate e di conseguenza accessibili da qualsiasi funzione all'interno dell'eseguibile
.rsrc	Include le risorse utilizzate dall'eseguibile come ad esempio icone, immagini, menu e stringhe che non sono parte dell'eseguibile stesso



GIORNO 1

KERNEL32.dll	Contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione dei file, la gestione della memoria
ADVAPI32.dll	Contiene le funzioni per interagire con i servizi ed i registri del sistema operativo



GIORNO 1

Tra le funzioni importate dal malware possiamo vederne alcune del Kernel32.dll:

Nel contesto del frammento di codice fornito, la funzione LockResource() viene utilizzata dopo che una risorsa è stata caricata in memoria con LoadResource(). Questo comportamento suggerisce che il malware stia cercando di ottenere accesso alla risorsa per eseguire operazioni specifiche su di essa.

La funzione **SizeofResource()** è un'**API di Windows** utilizzata per ottenere la dimensione, espressa in byte, di una risorsa specifica.

Questa funzione prende come argomenti un handle della risorsa (ottenuto solitamente tramite la funzione **LoadResource()**) e restituisce la dimensione della risorsa.

Nel contesto del frammento di codice fornito, la funzione SizeofResource() viene chiamata dopo che una risorsa è stata bloccata in memoria con la funzione LockResource(). Questo suggerisce che il malware sta cercando di ottenere le dimensioni della risorsa bloccata in memoria per eseguire operazioni specifiche su di essa.

OPTS	FTS (DAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource

GIORNO 1

RegSetValueExA: permette di aggiungere un nuovo valore all'interno del registro e di settare i rispettivi dati.

Accetta come parametri la chiave, la sottochiave e il dato da inserire. Ciò implica che il malware tenta di modificare o aggiungere un valore all'interno del Registro di sistema per poter persistere nel sistema o modifi carne le configurazioni.

RegCreateKeyExA: viene usata per creare una nuova chiave o aprire una chiave esistente nel Registro di sistema.

Potrebbe voler dire che il malware tenta di stabilire una persistenza all'interno del sistema o tenta di modificare le impostazioni di sistema.

OFTs	FTs (IAT)	Hint	Name
			szAnsi
Dword	Dword	Word	RegSetValueExA
000076AC	000076AC	0186	RegCreateKeyExA
000076BE	000076BE	015F	

GIORNO 1

Attraverso queste funzioni possiamo ipotizzare che il comportamento del malware riguardi il caricamento e l'uso di risorse all'interno del sistema. Questo suggerisce operazioni come il caricamento di dati, asset e la manipolazione delle risorse di sistema per eseguire codice malevolo o installare una backdoor.

Le funzioni esaminate indicano che il malware interagisce con il registro di sistema e utilizza funzioni di sistema per acquisire informazioni o modificare configurazioni di sistema. L'analisi effettuata porta a ipotizzare un comportamento tipico di un malware che tenta di ottenere accesso e controllo del sistema attraverso la manipolazione del registro di sistema e l'uso delle risorse di sistema.

Il **registro di Windows** è utilizzato per salvare informazioni sul sistema operativo, come le configurazioni del sistema stesso o delle applicazioni che girano sul sistema. I malware spesso sfruttano il registro di sistema per ottenere persistenza, assicurandosi che il codice malevolo venga eseguito automaticamente ad ogni avvio del sistema. Questa manipolazione del registro può includere l'aggiunta di chiavi di avvio automatico, la modifica di configurazioni critiche per eludere i controlli di sicurezza, o la registrazione di DLL malevole che vengono caricate da applicazioni legittime.

Inoltre, l'uso delle risorse di sistema può indicare che il malware sta cercando di eseguire attività nascoste, come la raccolta di dati sensibili, la comunicazione con server di comando e controllo, o l'esfiltrazione di informazioni. La capacità di interagire con il registro di sistema e di caricare risorse dinamicamente rende il malware particolarmente pericoloso, in quanto può adattarsi e modificare il suo comportamento per evitare la rilevazione e mantenere il controllo sul sistema infettato.

GIORNO 2



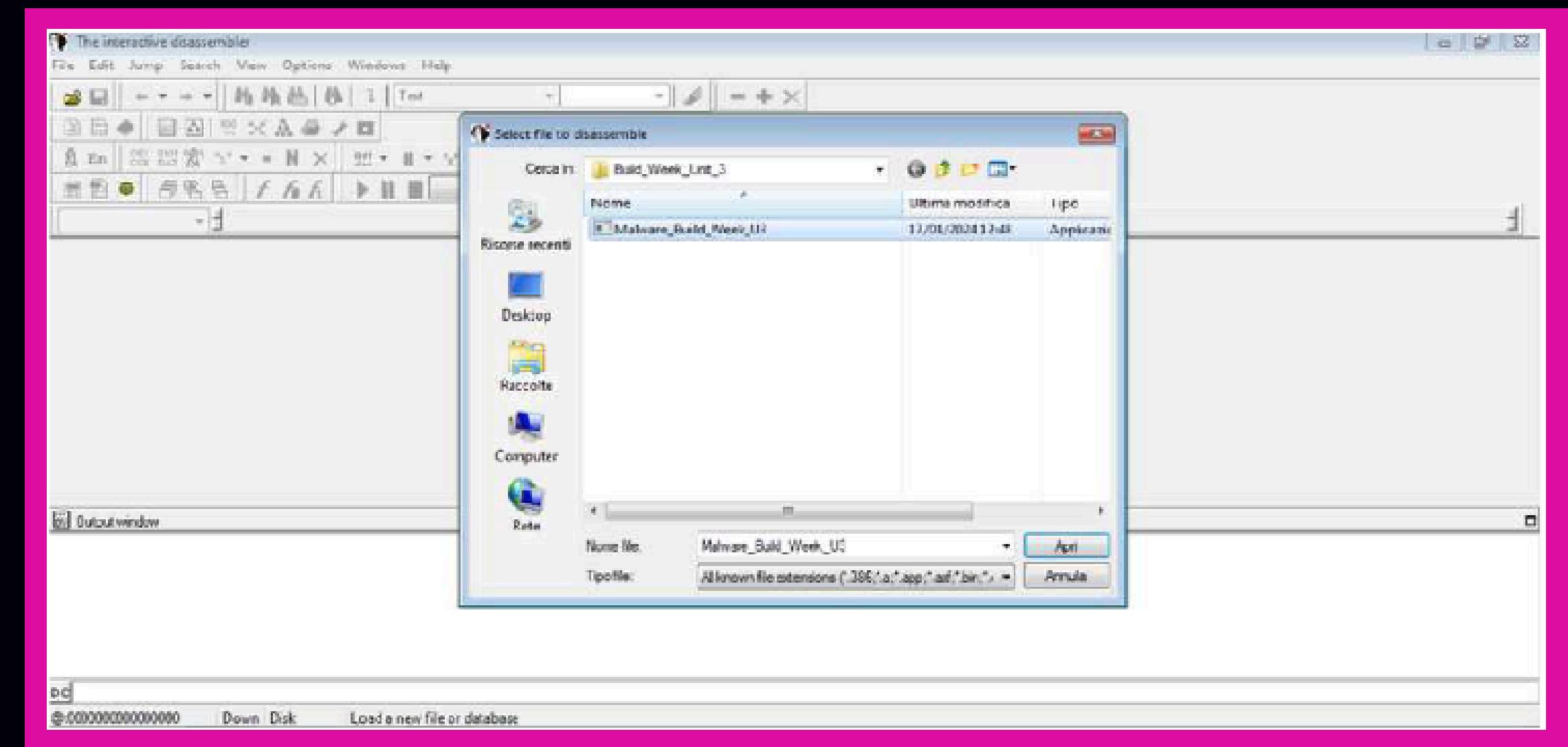
TRACCIA:

1. **Scopo della funzione alla locazione 00401021:** Descrivere l'obiettivo della funzione all'indirizzo 00401021.
2. **Passaggio dei parametri alla funzione 00401021:** Spiegare come vengono trasferiti i parametri alla funzione all'indirizzo 00401021.
3. **Oggetto del parametro alla locazione 00401017:** Identificare cosa rappresenta il parametro all'indirizzo 00401017.
4. **Significato delle istruzioni tra 00401027 e 00401029:** Analizzare e spiegare le istruzioni tra gli indirizzi 00401027 e 00401029.
5. **Traduzione in C del codice Assembly:** Convertire le istruzioni assembly tra 00401027 e 00401029 nel linguaggio C.
6. **Valore del parametro "ValueName" alla locazione 00401047:** Determinare il valore del parametro "ValueName" alla locazione 00401047.

Infine, spiegare quale funzione specifica stia implementando il Malware in questa sezione, considerando l'insieme delle due funzionalità appena descritte.

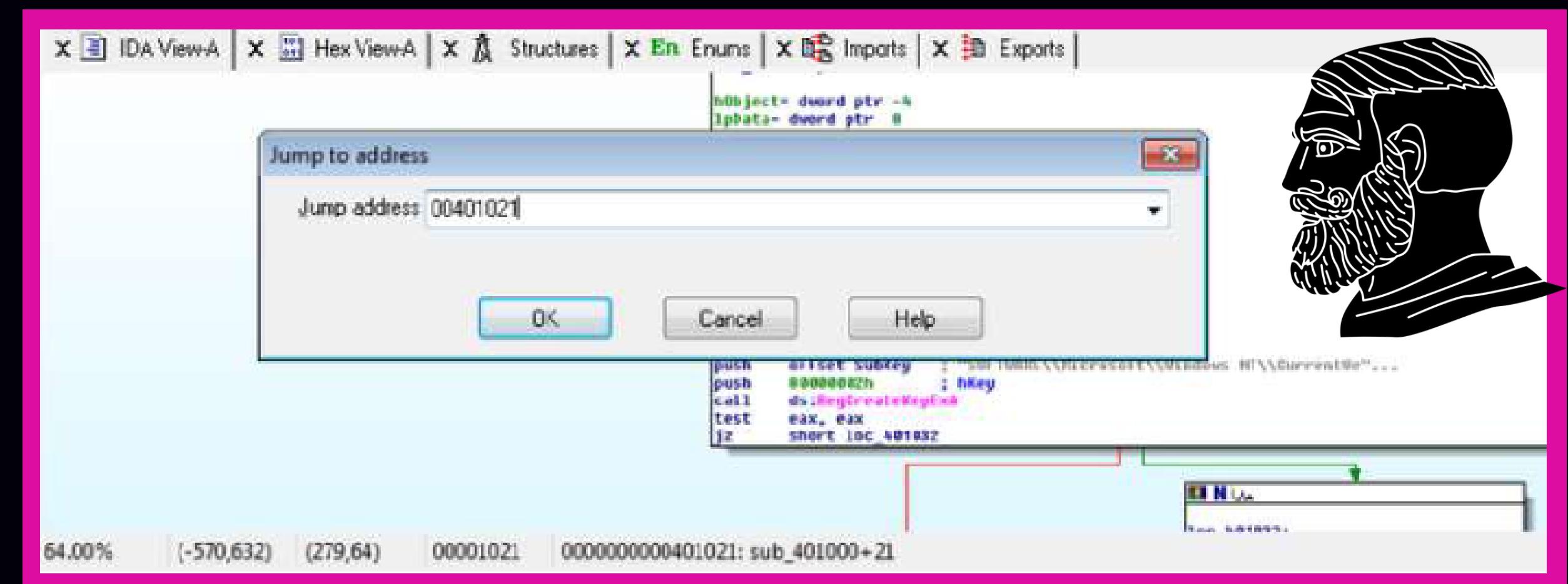
GIORNO 2

Iniziamo l'analisi del software
dannoso avviando il file con **IDA**



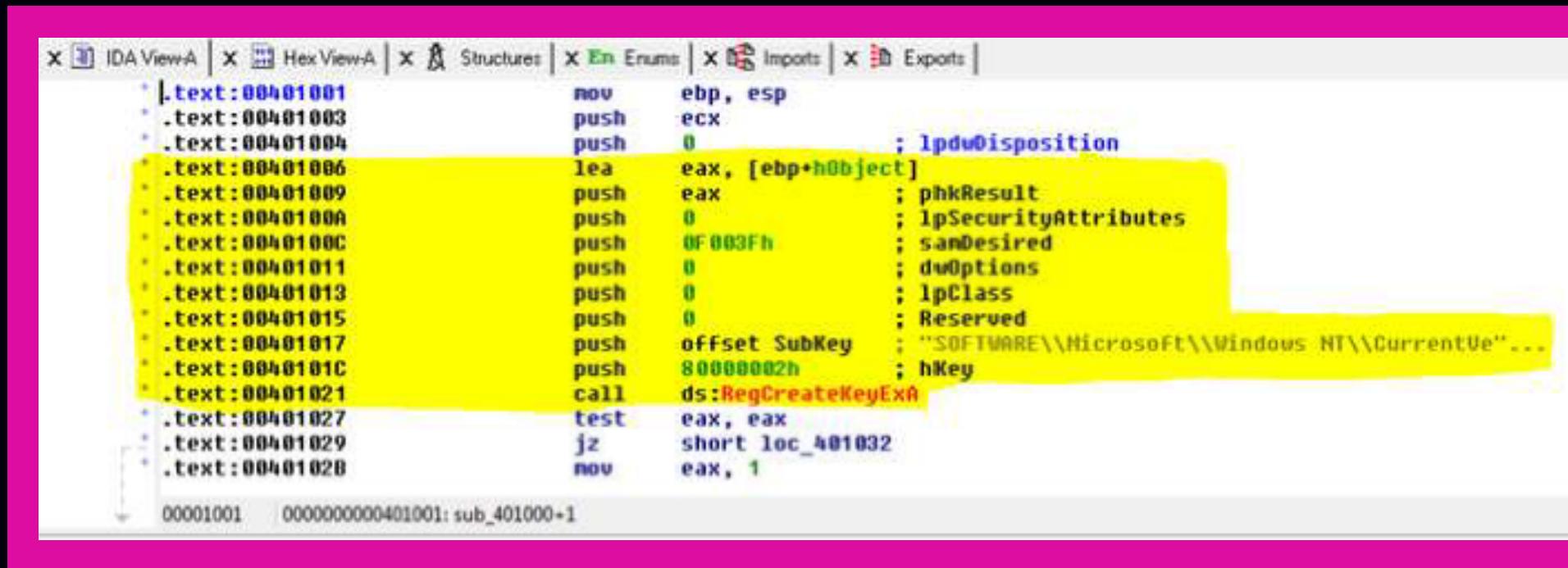
GIORNO 2

Utilizziamo la funzione “disassembler” ed otteniamo il codice assembly dal file, poi ci spostiamo all’indirizzo di memoria “00401021”



GIORNO 2

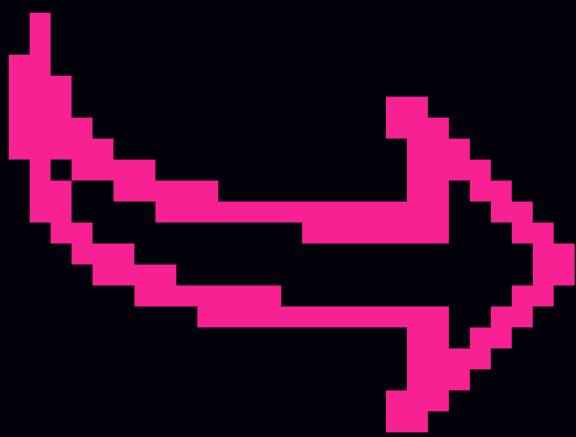
Arriviamo in questa schermata:



The screenshot shows the assembly code for a function in IDA Pro. The code is as follows:

```
.text:00401001        mov    ebp, esp
.text:00401003        push   ecx
.text:00401004        push   0          ; lpdwDisposition
.text:00401006        lea    eax, [ebp+hObject]
.text:00401009        push   eax          ; phkResult
.text:0040100A        push   0          ; lpSecurityAttributes
.text:0040100C        push   0F003Fh      ; samDesired
.text:00401011        push   0          ; dwOptions
.text:00401013        push   0          ; lpClass
.text:00401015        push   0          ; Reserved
.text:00401017        push   offset SubKey      ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
.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

00001001 0000000000401001: sub_401000+1
```



La funzione descritta dal compito serve per creare la chiave di registro specificata. Se la chiave esiste già, la funzione la apre.

- **lea eax, [ebp+hObject]**: Carica l'indirizzo effettivo di [ebp+hObject] nel registro eax. Questo probabilmente configura un puntatore dove la funzione memorizzerà un handle alla chiave di registro appena creata o aperta.
- **push eax**: Inserisce il valore di eax nello stack, probabilmente passando il puntatore a phkResult.
- **push 0**: Inserisce un valore nullo nello stack.
- **push 0F003Fh**: Inserisce il valore 0F003Fh nello stack. Questo rappresenta il parametro samDesired, che specifica i diritti di accesso desiderati per la chiave. 0F003Fh di solito concede l'accesso completo.
- **push offset SubKey**: Inserisce l'offset della stringa "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" nello stack. Questo è il nome della chiave di registro da creare o aprire.
- **push 80000002h**: Inserisce il valore 80000002h nello stack. Questo rappresenta il parametro hKey, che specifica l'handle di una chiave aperta attualmente o uno dei valori di handle predefiniti riservati. 80000002h rappresenta HKEY_LOCAL_MACHINE.
- **call ds:RegCreateKeyExA**: Chiama la funzione RegCreateKeyExA. Questa funzione tenterà di creare o aprire la chiave di registro specificata con i parametri forniti.

GIORNO 2

Attraverso le istruzioni push, i parametri vengono inseriti nello **stack** di memoria, che la funzione successivamente utilizzerà.



The screenshot shows assembly code in the left pane and a stack dump in the right pane. The assembly code includes several push instructions:

Assembly Instruction	Description
push eax	Parameter phkResult
push 0	Parameter lpSecurityAttributes
push 0F003Fh	Parameter samDesired
push 0	Parameter dwOptions
push 0	Parameter lpClass
push 0	Parameter Reserved
push offset SubKey	Parameter hKey (string offset)
push 0000002h	Parameter "SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinLogon"

L'elemento all'indirizzo di memoria **00401017** rappresenta la chiave di registro utilizzata dal malware per mantenere la sua persistenza sul computer della vittima. Questa chiave, con il percorso "Software\Microsoft\Windows NT\CurrentVersion\WinLogon", viene creata utilizzando la funzione "RegCreateKeyExA".

Nel codice assembly fornito, il parametro all'indirizzo 00401017 è l'inizio della stringa "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon". Questa stringa rappresenta il percorso della chiave di registro che l'applicazione sta cercando di creare o aprire.

Quando si passa un indirizzo di una stringa come parametro a una funzione in assembly, si fa riferimento all'offset dell'inizio della stringa rispetto all'inizio del segmento di dati o di testo del programma. Questo offset viene calcolato durante la fase di compilazione.

Pertanto, offset SubKey indica l'offset dell'inizio della stringa "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" rispetto all'inizio del segmento di dati o di testo dell'applicazione.

GIORNO 2

```
* .text:00401027          test    eax, eax
*: .text:00401029          jz     short loc_401032
*: .text:0040102B          mov    eax, 1
*: .text:00401030          jmp    short loc_40107B
```

Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029

Questa parte del codice assembly esegue un controllo condizionale seguito da alcune istruzioni di salto.

- **test eax, eax:** Questa istruzione esegue un'operazione logica AND tra il registro eax e se stesso. Viene utilizzata per impostare i flag del processore in base al valore di eax, verificando se eax è zero.
- **jz short loc_401032:** Questa istruzione di salto condizionato controlla il flag zero (ZF). Se ZF è impostato a seguito dell'istruzione test precedente, l'esecuzione salta all'indirizzo loc_401032. Se ZF non è impostato, l'esecuzione continua con l'istruzione successiva.
- **mov eax, 1:** Se il test precedente ha mostrato che eax non è zero, questa istruzione imposta il valore di eax a 1.
- **jmp short loc_40107B:** Questa è un'istruzione di salto incondizionato che porta l'esecuzione all'indirizzo loc_40107B indipendentemente dai risultati precedenti.

In sintesi, questo segmento di codice verifica se eax è zero, e se non lo è, imposta eax a 1. Indipendentemente dal risultato del test, l'esecuzione salta poi all'indirizzo loc_40107B. Se eax è zero, l'esecuzione salta all'indirizzo loc_401032 e ignora le istruzioni seguenti.

GIORNO 2

Salto visualizzato dal diagramma di flusso:

```
loc_401032:
mov    ecx, [ebp+cbData]
push   ecx          ; cbData
mov    edx, [ebp+lpData]
push   edx          ; lpData
push   1             ; dwType
push   0             ; Reserved
push   offset ValueName ; "GinaADLL"
mov    eax, [ebp+hObject]
push   eax          ; hKey
call   ds:RegSetValueExA
test   eax, eax
jz    short loc_401062
```

```
1 #include <stdio.h>
2
3 int main()
4 {
5
6     int zf; //questa variabile simula la ZF
7     int eax = 12; //questa variabile simula il registro eax
8     int test; //questa simula l'operatore test
9     test = eax - eax; // AND logico
10    if(test == 0){ // inizio del ciclo IF
11        zf = 1;
12    }
13    if(zf == 1){ // se la ZF sarà uguale a 1 farà il salto alla locazione "401032"
14        goto loc_401032;
15    }else{
16        goto loc_401078;} // altrimenti continuerà con il codice
17
18 loc_401032: printf("Salto avvenuto");
19 loc_401078: printf("Salto non avvenuto");
20
21 return 0;
22 }
```

Nel quesito precedente, il codice Assembly è stato trasformato in una struttura C corrispondente. Abbiamo simulato l'azione dell'istruzione IF usando variabili per eseguire un confronto tra due operandi.

Dopodiché abbiamo impostato delle condizioni basate sul risultato del confronto. Se il confronto dà come risultato 0, il flag ZF (zero flag) viene impostato a 1 e avviene il salto condizionale; altrimenti, il flusso del codice prosegue.

GIORNO 2

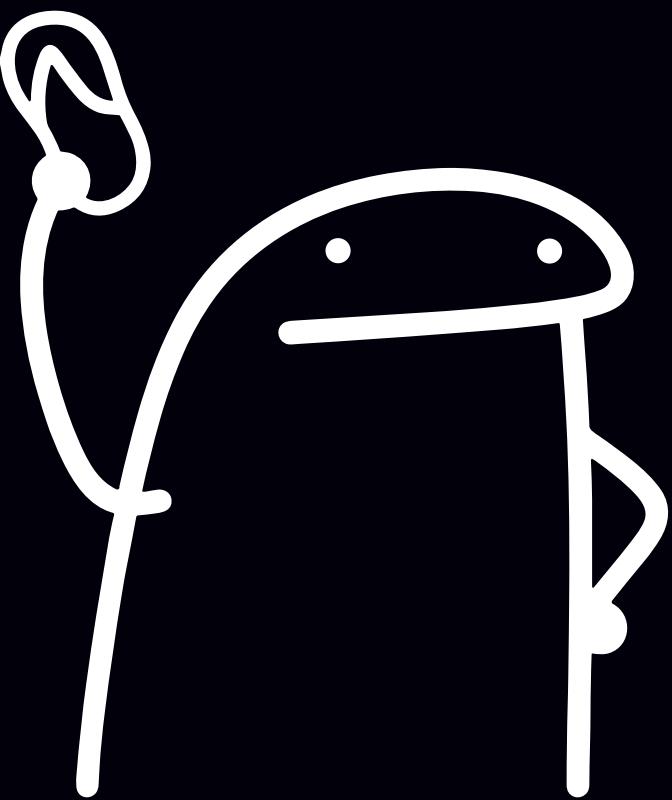
In questa parte del codice assembly, il malware sta eseguendo operazioni di manipolazione del registro di sistema di Windows.

Questa azione è probabile che faccia parte di un'attività malevola, come l'iniezione di codice dannoso o la modifica delle configurazioni di avvio del sistema.

Nel dettaglio, il malware utilizza le seguenti funzioni per manipolare il registro:

- **RegCreateKeyExA:** Questa funzione viene utilizzata per creare una nuova chiave di registro o aprire una già esistente. Nel caso specifico, il malware sta agendo sotto la chiave **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion**.
- **RegSetValueExA:** Dopo aver creato o aperto la chiave di registro, il malware utilizza questa funzione per impostare un valore di registro. In questo caso, il nome del valore è "**GinaDLL**". L'intenzione potrebbe essere quella di modificare o aggiungere un valore di configurazione nel registro di sistema.
- **Chiusura della handle della chiave di registro:** Il malware chiude la handle della chiave di registro se l'operazione di scrittura del valore ha avuto successo. Questa pratica è comune per liberare le risorse e mantenere una gestione pulita delle operazioni di registro.

Queste azioni indicano un tentativo deliberato di influenzare il comportamento del sistema operativo, potenzialmente introducendo funzionalità malevole o modificando le configurazioni di sistema per scopi dannosi.



GIORNO 3

TRACCIA:

Riprendete l'analisi del codice, analizzando le routine tra le locazioni di memoria:

- Qual è il valore del parametro «`ResourceName` » passato alla funzione `FindResourceA()`;
- Il susseguirsi delle chiamate di funzione che effettua il visto durante le lezioni teoriche. Che funzionalità sta implementando il Malware?
- È possibile identificare questa funzionalità utilizzando l'analisi statica basica? In caso di risposta affermativa, elencare le evidenze a supporto.

Entrambe le funzionalità principali del Disegnare un diagramma di flusso Malware viste finora sono richiamate all'interno della funzione (inserite all'interno dei box solo le informazioni circa le funzionalità principali) che comprenda le 3 funzioni.



GIORNO 3

Qual è il valore del parametro "ResourceName" passato alla funzione **FindResourceA()**

Il valore del parametro "ResourceName" passato alla funzione **FindResourceA()** è "**TGAD**". Questo valore è contenuto nella variabile **lpName**, come evidenziato nell'analisi del codice alla locazione di memoria **004010C9**.

La funzione **XREF** mostra che il parametro **lpName** è utilizzato nella subroutine che inizia alla locazione di memoria **00401080**

```
.text:00401088 ; CODE XREF: sub_401080+2F1j
    mov    eax, lpType
    push   eax
    mov    ecx, lpName
    push   ecx
    mov    edx, [ebp+hModule]
    push   edx
    call   ds:FindResourceA
    mov    [ebp+hResInfo], eax
    cmp    [ebp+hResInfo], 0
    jnz    short loc_4010DF
    xor    eax, eax
    jmp    loc_40110F
.loc_401088:
```

Reference Address	Value	Description
00401080	TGAD	DATA XREF: sub_401080:loc_401088T
00408034	TGAD	DATA XREF: sub_401080+3ET
00408040	BINARY	DATA XREF: .data:lpTypeTo
00408047	BINARY	DATA XREF: .data:lpNameTo
00408048	RI	DATA XREF: sub_401080:loc_401082T
0040804C	GinaDLL	DATA XREF: sub_401080+3ET
00408054	SubKey	DATA XREF: sub_401080+3ET
00408058	SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	DATA XREF: sub_401080+3ET

GIORNO 3

Che funzionalità sta implementando il Malware?

La sequenza di chiamate di funzione in questa sezione di codice sembra coinvolgere la ricerca e il caricamento di una risorsa da un modulo. La funzionalità implementata è legata al recupero e all'utilizzo di risorse da un modulo.

Questo comportamento può essere utilizzato per scopi come il caricamento di file o la manipolazione di risorse, suggerendo che il malware sta cercando di ottenere accesso a queste risorse per eseguire operazioni specifiche su di esse.

```
.text:004010DF          ; CODE XREF: sub_401080+56↑j
.text:004010DF loc_4010DF:
.text:004010DF          mov    eax, [ebp+hResInfo]
.text:004010E2          push   eax
                           ; hResInfo
.text:004010E3          mov    ecx, [ebp+hModule]
                           ; hModule
.text:004010E6          push   ecx
.text:004010E7          call   ds:LoadResource
.text:004010ED          mov    [ebp+hResData], eax
.text:004010F0          cmp    [ebp+hResData], 0
.text:004010F4          jnz   short loc_4010FB
.text:004010F6          jmp   loc_4011A5
```

GIORNO 3

È possibile identificare questa funzionalità utilizzando l'analisi statica basica?

GIORNO 3

È possibile identificare questa funzionalità utilizzando l'analisi statica basica?

Sì, è possibile identificare questa funzionalità
utilizzando l'analisi statica basica.

GIORNO 3



GIORNO 3

In caso di risposta affermativa, elencare le evidenze a supporto.

Le evidenze a supporto dell'identificazione della funzionalità tramite analisi statica basica includono:

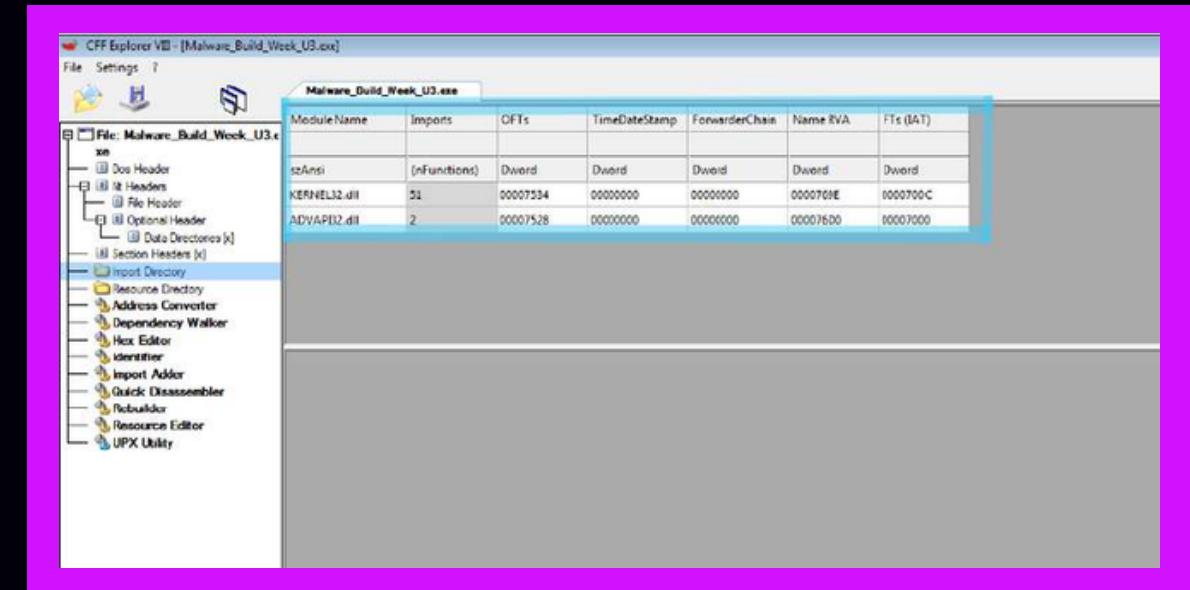
1. Chiamata a FindResourceA: Questa funzione è utilizzata per cercare una risorsa in un modulo, suggerendo che il malware sta cercando di individuare una specifica risorsa.
2. Chiamata a LoadResource: Dopo aver individuato la risorsa, questa funzione la carica in memoria, indicando che il malware necessita di questa risorsa per proseguire con le sue operazioni.
3. Chiamata a LockResource: Questa funzione blocca l'accesso in memoria alla risorsa specificata, permettendo al malware di manipolare direttamente i dati della risorsa.
4. Chiamata a SizeofResource: Questa funzione ottiene la dimensione della risorsa caricata, indicando che il malware deve conoscere la quantità di dati con cui sta lavorando.



GIORNO 3

Utilizzando strumenti di analisi statica come CFF Explorer, è possibile esaminare le funzioni e le librerie importate dal malware.

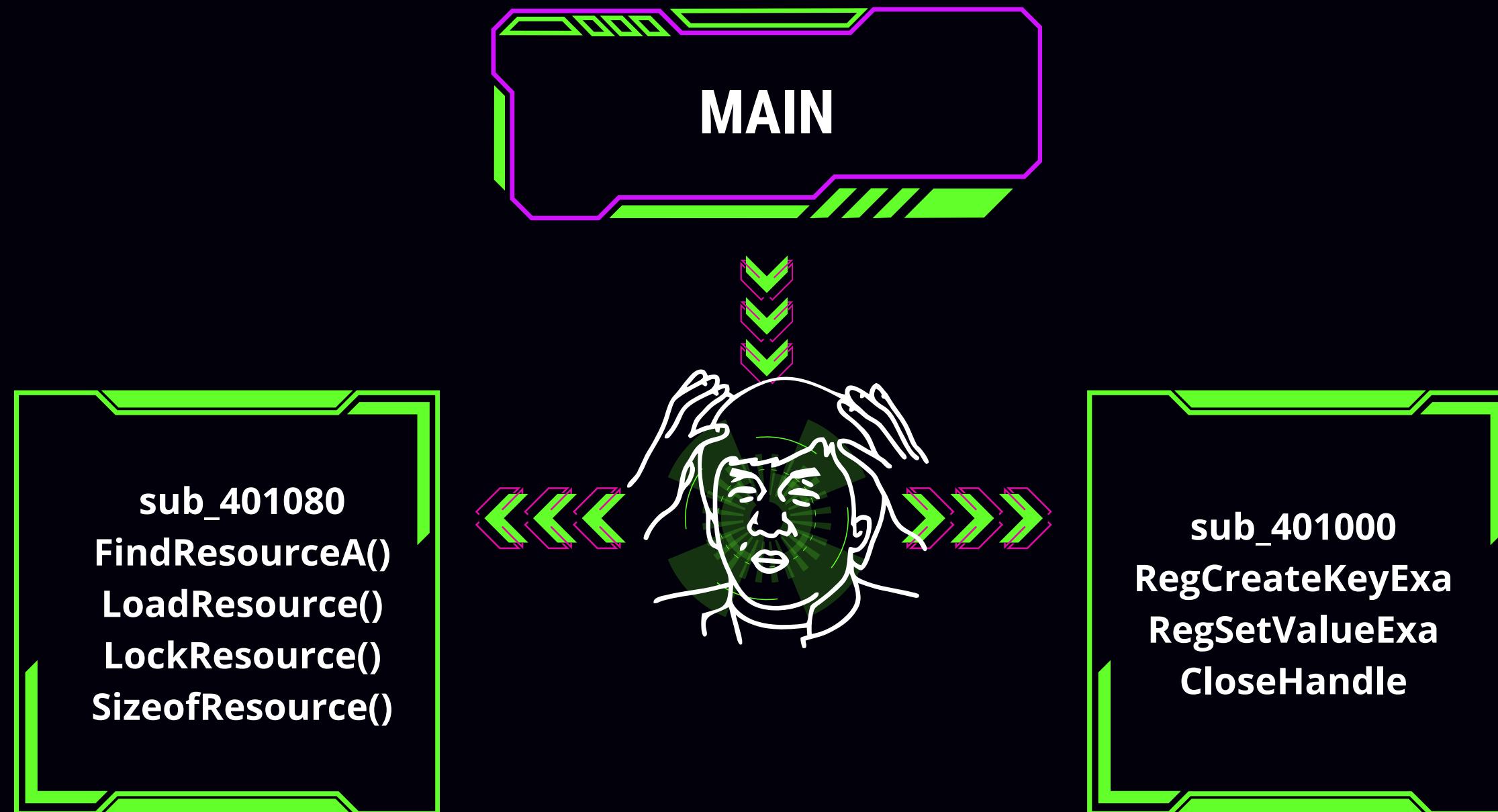
Questo fornisce una visione dettagliata delle risorse esterne utilizzate dal software dannoso, confermando le chiamate alle funzioni `FindResourceA`, `LoadResource`, `LockResource`, e `SizeofResource`.



OFTs	Fts (IAT)	Hint	Name
00007570	00007048	00007734	00007736
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	028B	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007604	00007604	001B	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA
000076F0	000076F0	0174	GetVersion
000076FE	000076FE	007D	ExitProcess
0000770C	0000770C	019F	HeapFree
00007718	00007718	011A	GetLastError
00007728	00007728	020F	WriteFile
00007734	00007734	029E	TerminateProcess

GIORNO 3

Disegnare un diagramma di flusso



GIORNO 3

Dall'analisi delle chiamate di funzione nel codice esaminato, emerge che il malware in questione potrebbe essere un dropper. I dropper sono programmi dannosi creati per installare altri tipi di malware, come virus o backdoor.



GIORNO 4

TRACCIA:

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 Analizzate ora i risultati di Process Monitor.

Fate click su «ADD» poi su «Apply» come abbiamo visto nella lezione teorica. Filtrate includendo solamente l'attività sul registro di Windows

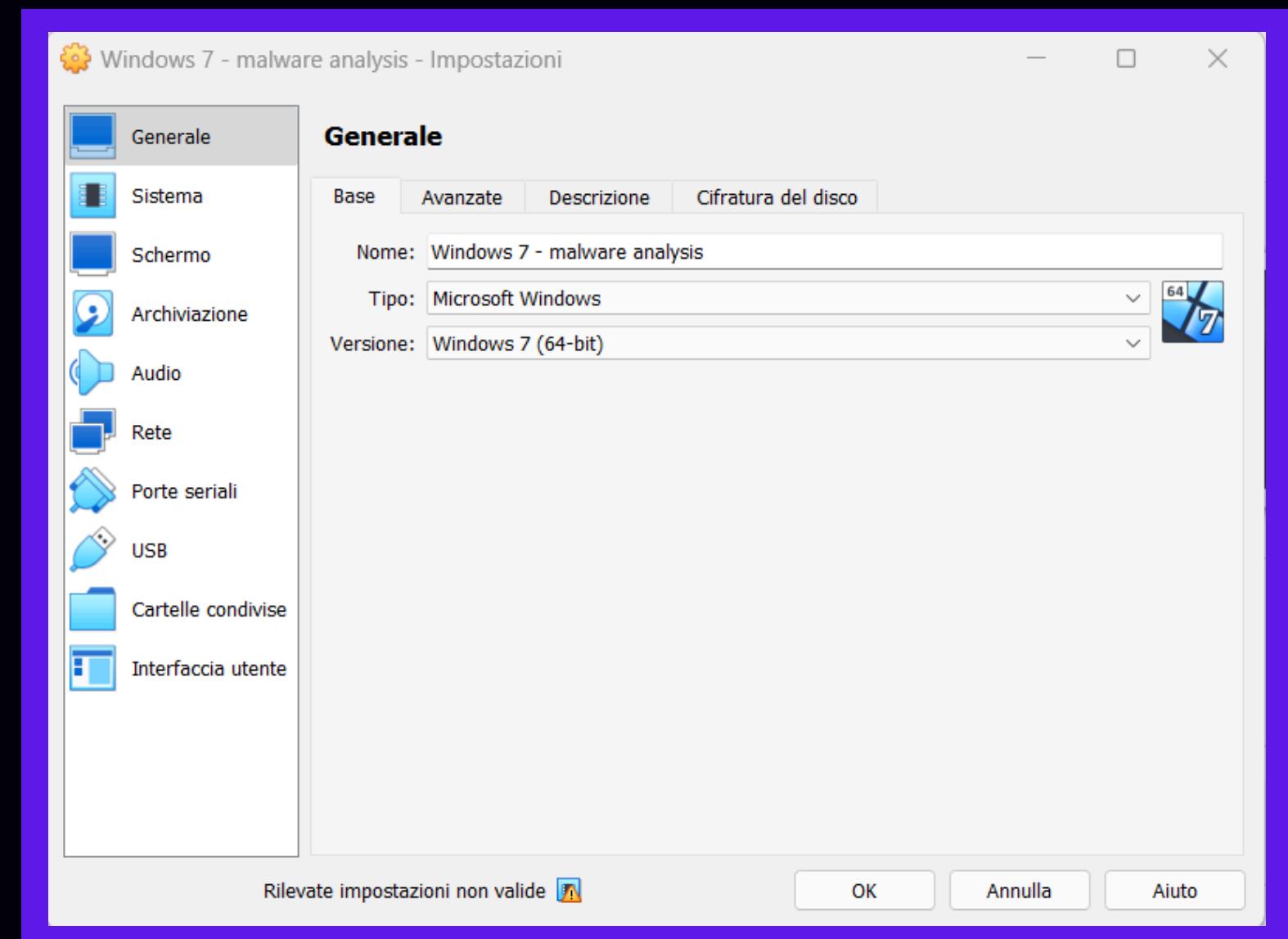


GIORNO 4

L'**analisi dinamica** del **malware** prevede l'esecuzione del malware in un ambiente controllato e sicuro per osservare e studiare il suo comportamento e le sue funzionalità in tempo reale

Preparazione dell'Ambiente di Analisi:

- scheda di rete: disabilitiamo la scheda di rete in modo che la macchina non abbia accesso diretto ad internet, né alle altre macchine presenti sulla rete.
- creare instantanee: Prima di eseguire il malware, creamo uno snapshot della VM per poter ripristinare facilmente lo stato iniziale.
- Disabilitazione del Controller USB: Quando un dispositivo USB viene collegato alla macchina fisica, potrebbe essere riconosciuto anche dall'ambiente di test, consentendo potenzialmente al malware di sfruttare il dispositivo per propagarsi. Per evitare questo rischio, disabilitiamo il controller USB nella configurazione della macchina virtuale.
- Cartelle Condivise: Le cartelle condivise tra l'host e il guest possono essere sfruttate dal malware per propagarsi al di fuori dell'ambiente di test. per prevenire questo disabilitiamo le cartelle condivise tra la macchina fisica e la VM per evitare la diffusione del malware.

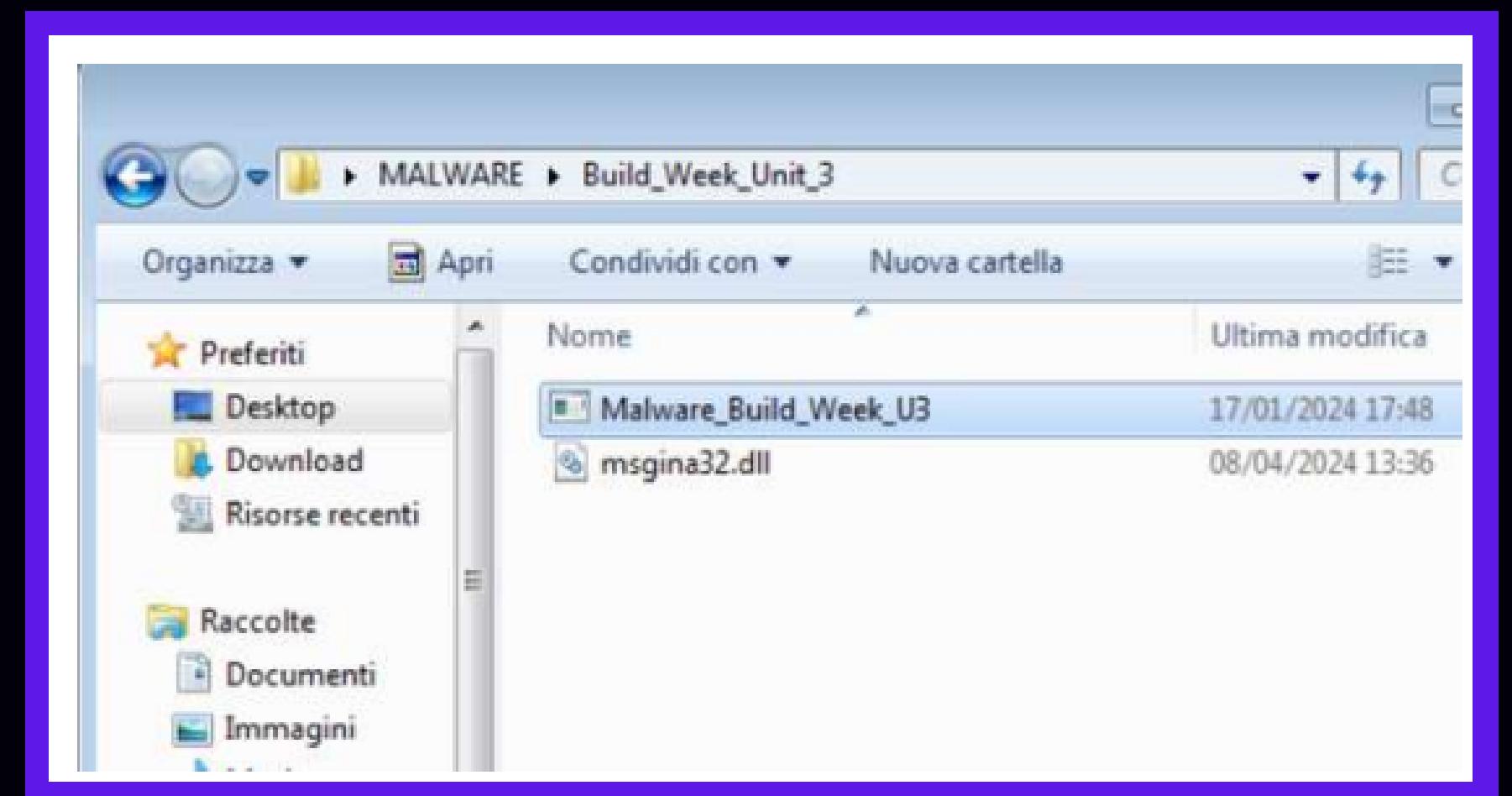


GIORNO 4

Iniziamo aprendo **ProcessMonitor**, o “**procmon**”, un tool avanzato per **Windows** che permette di monitorare i processi ed i thread attivi, l’attività di rete, l’accesso ai file e le chiamate di sistema.

Il team ha effettuato l’esecuzione del Malware che si trova al path:
C\Users\user\Desktop\MALWARE\Build_Week_Unit_3.

Una volta avviato abbiamo potuto notare che è stato creato un file chiamato **msgina32.dll** nella stessa cartella, che è un **.dll** malevolo il cui scopo probabilmente è quello di intercettare le credenziali dell’utente.



GIORNO 4

Abbiamo avviato **procmon** e impostato i filtri in modo da visualizzare solo i processi attivi del malware e le attività di registro.

Il malware apporta diverse modifiche al sistema:

Crea un file **msgina32.dll** nella cartella:

C\Users\user\Desktop\MALWARE\Build_Week_Unit_3.

Modifica il registro di sistema, impostando il valore della chiave **GinaDLL** sotto:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon

Al percorso del **dll msgina32.dll** -> però possiamo notare **ACCESS DENIED**.

14:54: [!]	Malware_Build_Week_U3.exe	2948	RegOpenKey	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Diagnos	NAME NOT FOUND
14:54: [!]	Malware_Build_Week_U3.exe	2948	RegQueryKey	HKLM	SUCCESS
14:54: [!]	Malware_Build_Week_U3.exe	2948	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
14:54: [!]	Malware_Build_Week_U3.exe	2948	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
14:54: [!]	Malware_Build_Week_U3.exe	2948	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
14:54: [!]	Malware_Build_Week_U3.exe	2948	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	ACCESS DENIED
14:54: [!]	Malware_Build_Week_U3.exe	2948	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
14:54: [!]	Malware_Build_Week_U3.exe	2948	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options	SUCCESS
14:54: [!]	Malware_Build_Week_U3.exe	2948	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options	SUCCESS
14:54: [!]	Malware_Build_Week_U3.exe	2948	RegCloseKey	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions	SUCCESS
14:54: [!]	Malware_Build_Week_U3.exe	2948	RegCloseKey	HKLM	SUCCESS

GIORNO 4

Utilizzando OllyDBG, possiamo osservare che il malware modifica il registro di sistema per ottenere persistenza.

Specificamente, imposta il valore della chiave **GinaDLL** sotto **HKEY_LOCAL_MACHINE\“SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon”** al percorso della **DLL malevola**.



In questo modo, la **DLL** viene caricata ogni volta che l'utente effettua un accesso o un logout.

GIORNO 5

TRACCIA:

GINA (Graphical identification and authentication) è un componente legittimo di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica - ovvero permette agli utenti di inserire username e password nel classico riquadro Windows.

- Cosa può succedere se il file .dll legittimo viene sostituito con un file .dll malevolo, che intercetta i dati inseriti?

Sulla base della risposta sopra, delineate il profilo del Malware e delle sue funzionalità. Unite tutti i punti per creare un grafico che ne rappresenti lo scopo ad alto livello.



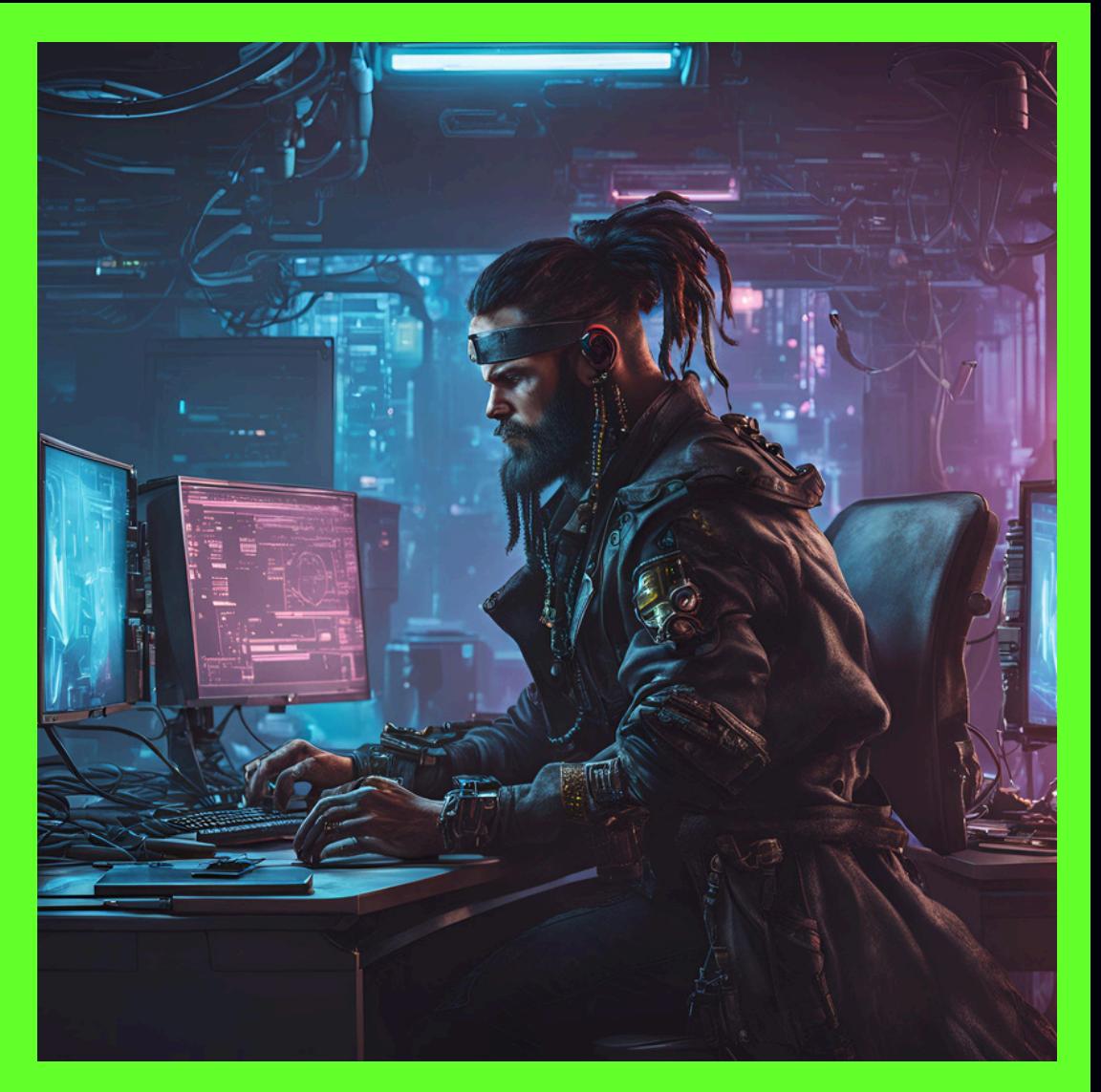
GIORNO 5

GINA (Graphical Identification and Authentication)

GINA.dll, acronimo di Graphical Identification and Authentication, era un componente di Windows utilizzato per gestire il processo di autenticazione degli utenti. Questo componente era cruciale nelle versioni precedenti del sistema operativo, come Windows XP e Windows Server 2003.

La funzione principale di GINA.dll era fornire un'interfaccia grafica per l'autenticazione degli utenti durante l'accesso al sistema, gestendo la schermata di login dove venivano inserite le credenziali degli utenti (nome utente e password).

Con l'avvento di Windows Vista e delle versioni successive, il meccanismo di autenticazione è stato rivisitato, sostituendo GINA.dll con i Credential Providers. Questi offrono maggiore flessibilità e supporto per vari metodi di autenticazione, inclusi quelli biometrici e basati su certificati.



GIORNO 5

Implicazioni della Sostituzione di GINA.dll con un File Malevolo

Quando il file legittimo GINA.dll, responsabile dell'autenticazione degli utenti, viene rimpiazzato da un file .dll malevolo progettato per intercettare i dati inseriti, diverse conseguenze possono emergere:

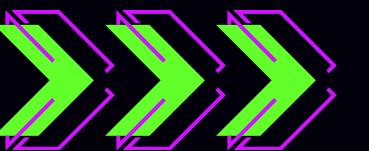
- 1. Violazione della Sicurezza:** Il file malevolo può segretamente catturare le credenziali di accesso degli utenti, come nomi utente e password, durante la loro digitazione. Queste informazioni riservate potrebbero essere trasmesse a un server remoto controllato dagli aggressori, compromettendo la sicurezza del sistema e le informazioni personali degli utenti.
- 2. Accesso Illegittimo:** Gli aggressori, utilizzando le credenziali rubate, possono accedere in maniera non autorizzata al sistema o ai servizi utilizzati dagli utenti, provocando ulteriori violazioni della sicurezza, sottrazione di dati sensibili e compromissione di risorse cruciali.
- 3. Esposizione del Sistema a Ulteriori Attacchi:** La presenza di un file .dll malevolo può aprire il sistema a ulteriori forme di attacco o compromettere la sicurezza complessiva. Gli aggressori possono usare il file per installare altri malware o per ottenere un accesso persistente al sistema, rendendo difficile la rimozione del file malevolo e il ripristino della sicurezza del sistema.

In sintesi, la sostituzione di GINA.dll con un file .dll malevolo costituisce una minaccia significativa alla sicurezza informatica, con il potenziale di compromettere dati sensibili, permettere accessi non autorizzati e causare danni rilevanti al sistema e agli utenti. È essenziale implementare misure di sicurezza efficaci, come l'uso di software antivirus e il monitoraggio continuo dei file di sistema, per prevenire e rilevare tali minacce.



GIORNO 5

Grafico di rappresentazione ad alto livello



- Infezione del Sistema
(Phishing, Download, Exploit)
- Esecuzione del Malware
- Creazione del file Gina.dll
- Ottenimento della Persistenza
(Chiave di Registro)
- Furto delle Credenziali



GIORNO 5

Subroutine di WINLOGON

Analizzando il nuovo file GINA.dll con IDA, si osserva che la libreria ha il controllo sull'intero sistema di autenticazione della macchina, eseguendo arbitrariamente subroutine del componente Winlogon.

Questo livello di controllo permette al file GINA.dll di interagire direttamente con le funzioni fondamentali del sistema operativo responsabili dell'accesso e della gestione degli utenti, potenzialmente compromettendo la sicurezza dell'intero sistema.

Winlogon è un componente critico del sistema operativo Windows, progettato per gestire il processo di login e logout degli utenti. È responsabile di assicurare che l'accesso al sistema sia sicuro e affidabile.

Tra le sue funzioni principali, Winlogon gestisce l'autenticazione dell'utente, il caricamento dei profili utente, la gestione delle sessioni e il controllo degli eventi di sistema correlati all'accesso e alla disconnessione degli utenti.

Vediamo in dettaglio alcune delle funzioni chiave di Winlogon:



GIORNO 5

- **WlxLoggedOnSAS:** Questa funzione viene attivata quando un utente esegue una Secure Attention Sequence (SAS), come la combinazione di tasti Ctrl+Alt+Canc, mentre è già loggato nel sistema. La funzione può gestire eventi come il cambio utente o l'apertura di una nuova sessione di login. Quando viene attivata una SAS, WlxLoggedOnSAS gestisce l'evento e può eseguire azioni specifiche in risposta, come mostrare una finestra di dialogo per confermare l'azione richiesta dall'utente o avviare un'applicazione specifica. Questa funzione è essenziale per mantenere la sicurezza del sistema, impedendo l'accesso non autorizzato durante la sessione utente.

- **WlxLogoff:** Questa funzione viene chiamata quando un utente esegue il logout dal sistema. Gestisce il processo di disconnessione dell'utente, assicurandosi che tutte le risorse associate alla sessione utente vengano liberate correttamente. Inoltre, garantisce che l'ambiente di lavoro sia ripulito in modo appropriato, prevenendo potenziali problemi di sicurezza e stabilità del sistema. Il corretto funzionamento di WlxLogoff è cruciale per mantenere l'integrità del sistema operativo, soprattutto in ambienti condivisi o multiutente.

```
.text:10001350 ; ===== SUBROUTINE =====
.text:10001350
.text:10001350
.text:10001350 public WlxLoggedOnSAS
.text:10001350 WlxLoggedOnSAS proc near
.text:10001350     push    offset aWlxloggedons_0 ; "WlxLoggedOnSAS"
.text:10001355     call    sub_10001000
.text:1000135A     jmp    eax
.text:1000135A WlxLoggedOnSAS endp
```

```
.text:10001360 ; ===== SUBROUTINE =====
.text:10001360
.text:10001360
.text:10001360
.text:10001360 public WlxLogoff
.text:10001360 WlxLogoff proc near
.text:10001360     push    offset aWlxlogoff_0 ; "WlxLogoff"
.text:10001365     call    sub_10001000
.text:1000136A     jmp    eax
.text:1000136A WlxLogoff endp
```

GIORNO 5

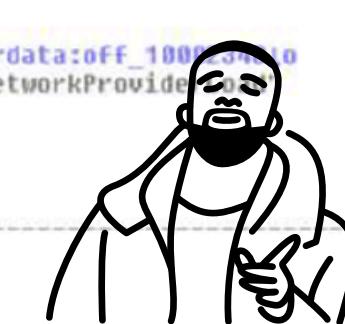
- **WlxNegotiate:** Questa funzione viene chiamata durante il processo di autenticazione dell'utente per negoziare i parametri di autenticazione tra Winlogon e un modulo di autenticazione personalizzato, come un modulo di autenticazione fornito da terze parti. La negoziazione permette al processo di autenticazione personalizzato di comunicare i suoi requisiti e capacità a Winlogon, stabilendo un protocollo di autenticazione sicuro e appropriato. Questa flessibilità è importante per supportare una vasta gamma di metodi di autenticazione, inclusi quelli basati su biometria e certificati, garantendo che l'accesso al sistema rimanga sicuro e adattabile alle esigenze specifiche dell'organizzazione.

- **WlxNetworkProviderLoad:** Questa funzione è parte del sottosistema Winlogon e viene chiamata per caricare un provider di rete durante il processo di autenticazione dell'utente. Tale provider può essere utilizzato per consentire l'accesso a risorse di rete, come file condivisi o stampanti, una volta eseguito l'accesso al sistema. La funzione WlxNetworkProviderLoad garantisce che il provider di rete sia disponibile e caricato correttamente, permettendo all'utente di accedere alle risorse in modo sicuro e affidabile durante la sessione di lavoro. Questa funzione è essenziale per gli ambienti di rete aziendali, dove l'accesso sicuro e gestito alle risorse condivise è fondamentale per la produttività e la sicurezza dei dati.

```
.text:10001370 ; ----- S U B R O U T I N E -----
.text:10001370
.text:10001370
.text:10001370
.text:10001370
.text:10001370 public WlxNegotiate
.text:10001370 WlxNegotiate proc near      ; DATA XREF: .rdata:OFF_100013410
.text:10001370     push    offset aWlxnegotiate_0 ; "WlxNegotiate"
.text:10001370     call    sub_10001000
.text:10001370     jmp    eax
.text:10001370
.text:10001370 WlxNegotiate endp
.text:1000137A
```



```
.text:10001380 ; ----- S U B R O U T I N E -----
.text:10001380
.text:10001380
.text:10001380
.text:10001380
.text:10001380 public WlxNetworkProviderLoad
.text:10001380 WlxNetworkProviderLoad proc near      ; DATA XREF: .rdata:OFF_100013410
.text:10001380     push    offset aWlxnetworkpr_0 ; "WlxNetworkProvider"
.text:10001385     call    sub_10001000
.text:1000138A     jmp    eax
.text:1000138A WlxNetworkProviderLoad endp
.text:1000138A
.text:1000138C align 10h
.text:10001390 ; Exported entry 45. WlxReconnectNotify
.text:10001390
```



GIORNO 5

- **WlxNegotiate:** Questa funzione viene chiamata durante il processo di autenticazione dell'utente per negoziare i parametri di autenticazione tra Winlogon e un modulo di autenticazione personalizzato, come un modulo di autenticazione fornito da terze parti. La negoziazione permette al processo di autenticazione personalizzato di comunicare i suoi requisiti e capacità a Winlogon, stabilendo un protocollo di autenticazione sicuro e appropriato. Questa flessibilità è importante per supportare una vasta gamma di metodi di autenticazione, inclusi quelli basati su biometria e certificati, garantendo che l'accesso al sistema rimanga sicuro e adattabile alle esigenze specifiche dell'organizzazione.



```
.text:10001370 ; ----- S U B R O U T I N E -----
.text:10001370
.text:10001370
.text:10001370     public WlxNegotiate
.text:10001370     proc near
.text:10001370         push    offset aWlxnegotiate_0 ; "WlxNegotiate"
.text:10001370         call    sub_10001000
.text:10001370         jnp    eax
.text:10001370     endp
.text:10001370 WlxNegotiate
.text:10001370
```

- **WlxNetworkProviderLoad:** Questa funzione è parte del sottosistema Winlogon e viene chiamata per caricare un provider di rete durante il processo di autenticazione dell'utente. Tale provider può essere utilizzato per consentire l'accesso a risorse di rete, come file condivisi o stampanti, una volta eseguito l'accesso al sistema. La funzione WlxNetworkProviderLoad garantisce che il provider di rete sia disponibile e caricato correttamente, permettendo all'utente di accedere alle risorse in modo sicuro e affidabile durante la sessione di lavoro. Questa funzione è essenziale per gli ambienti di rete aziendali, dove l'accesso sicuro e gestito alle risorse condivise è fondamentale per la produttività e la sicurezza dei dati.



```
.text:10001380 ; ----- S U B R O U T I N E -----
.text:10001380
.text:10001380
.text:10001380     public WlxNetworkProviderLoad
.text:10001380     WlxNetworkProviderLoad proc near      ; DATA XREF: .rdata:off_1000234810
.text:10001380         push    offset aWlxnetworkpr_0 ; "WlxNetworkProviderLoad"
.text:10001380         call    sub_10001000
.text:10001380         jmp    eax
.text:10001380     WlxNetworkProviderLoad endp
.text:10001380
.text:10001380     align 10h
.text:10001390 ; Exported entry 45. WlxReconnectNotify
.text:10001390
```

GIORNO 5

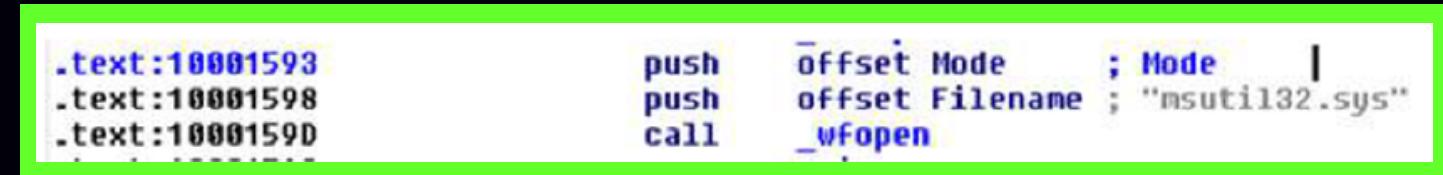
Nelle analisi finali, si è riscontrato che la libreria effettua una chiamata a un file di sistema denominato msutil32.sys, il quale sembra lecito ma non è un file di sistema standard. Questo file potrebbe trattarsi di un file malevolo mascherato per eludere la rilevazione. La presenza di un file .dll malevolo che effettua chiamate a msutil32.sys indica un tentativo di registrare informazioni sensibili, come le credenziali di accesso, in un file nascosto nel sistema.

Continuando l'analisi, possiamo osservare che la libreria apre il file per scrivere il nome utente, il dominio, la password e la password precedente. I valori raccolti vengono formattati in una stringa e poi passati alla subroutine situata a text:10001570. Questa subroutine elabora un'altra stringa e la salva nel file C:\Windows\System32\msutil32.sys.

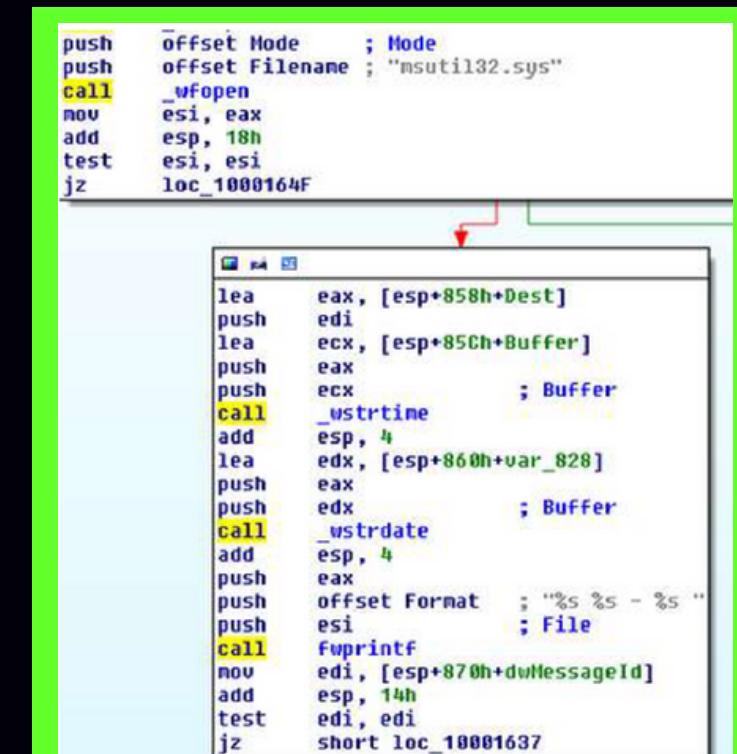
La libreria registra informazioni di accesso in **msutil32.sys** nel formato:

- Data - Ora - Nome Utente - Gruppo - Password - Vecchia Password

Questo formato di registrazione delle credenziali rappresenta una grave violazione della sicurezza, poiché permette all'attaccante di raccogliere informazioni sensibili che possono essere utilizzate per ulteriori attacchi o accessi non autorizzati.



```
.text:10001593 push offset Mode ; Mode
.text:10001598 push offset Filename ; "msutil32.sys"
.text:1000159D call _wfopen
```

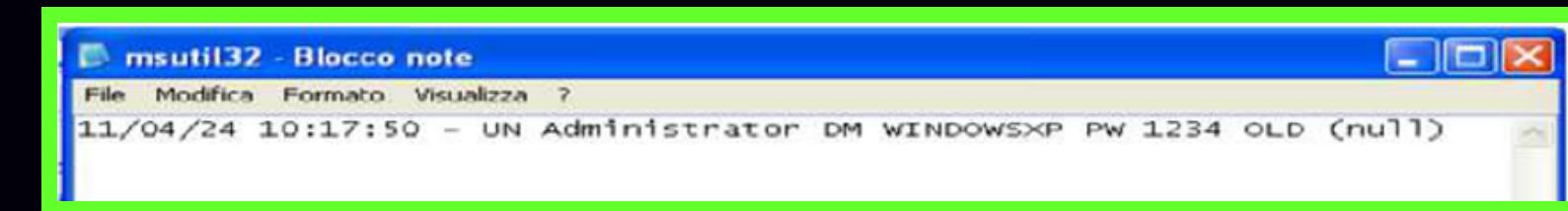


```
push offset Mode ; Mode
push offset Filename ; "msutil32.sys"
call _wfopen
mov esi, eax
add esp, 18h
test esi, esi
jz loc_1000164F

lea eax, [esp+858h+Dest]
push edi
lea ecx, [esp+85Ch+Buffer]
push eax
push ecx
call _vstrtime
add esp, 4
lea edx, [esp+860h+var_828]
push edx
push eax
push edx
call _vstrdate
add esp, 4
push eax
push offset Format ; "%S %S - %S "
push esi
push eax
call fwprintf
edi, [esp+870h+dwMessageId]
add esp, 14h
edi, edi
short loc_10001637
```

In questa immagine, possiamo vedere come la libreria manipola i dati:

- push offset del nome del file: "msutil32.sys"
- call a una routine per aprire il file
- mov valore nel registro
- lea carica l'indirizzo effettivo nel buffer
- call per scrivere i dati nel file
- push per formattare la stringa



OUR TEAM



Alberto Guimp



Luca Lenzi



Giovanni Sannino



Max Aldrovandi



Morgan Petrelli



Michele Covi