Progetto Modulo 6
Malware Analysis

Ettore Farris - 21/04/2023

1) Traccia

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

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

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 costrutto C.*
- *Valutate ora la chiamata alla locazione **00401047**, qual è il valore del parametro «ValueName»?*

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*

Analizzate ora i risultati di Process Monitor (consiglio: utilizzate il filtro come in figura sotto per estrarre solo le modifiche apportate al sistema da parte del Malware). Fate click su «ADD» poi su «Apply» come abbiamo visto nella lezione teorica.

Filtrate includendo solamente l'attività sul registro di Windows.

- *Quale chiave di registro viene creata?*
- *Quale valore viene associato alla chiave di registro creata?*

Passate ora alla visualizzazione dell'attività sul file system.

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

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

2) Svolgimento

Analisi statica

- ***Quanti parametri sono passati alla funzione Main()?***

Per rispondere a questo quesito ci avvaliamo del disassembler IDA Pro in modo da vedere il codice assembly del malware. Analizzando la funzione *main(),* situata all'indirizzo di memoria *004011D0,* notiamo, a giudicare dall'offset positivo, che ci sono **3 parametri**, *argc, argv* ed *envp.*

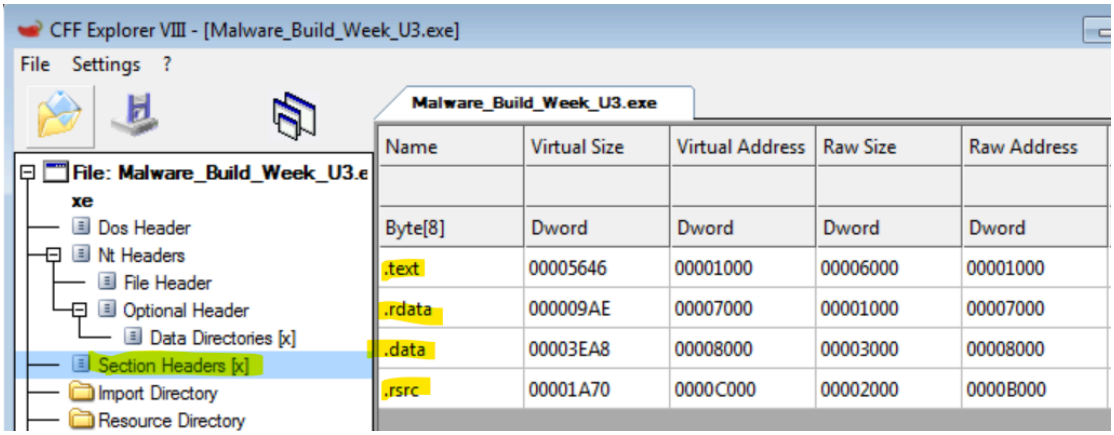
```
.text:004011D0 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:004011D0 _main           proc near               ; CODE XREF: start+AF↓p
.text:004011D0
.text:004011D0 hModule         = dword ptr -11Ch
.text:004011D0 Data            = byte ptr -118h
.text:004011D0 var_117         = byte ptr -117h
.text:004011D0 var_8           = dword ptr -8
.text:004011D0 var_4           = dword ptr -4
.text:004011D0 argc            = dword ptr  8
.text:004011D0 argv            = dword ptr  0Ch
.text:004011D0 envp            = dword ptr  10h
.text:004011D0
```

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

Le **variabili** dichiarate invece sono **5**, *hModule, Data, var_117, var_8* e *var_4.* Si arriva a conclusione considerando l'offset negativo delle variabili (vedi immagine del punto precendente).

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

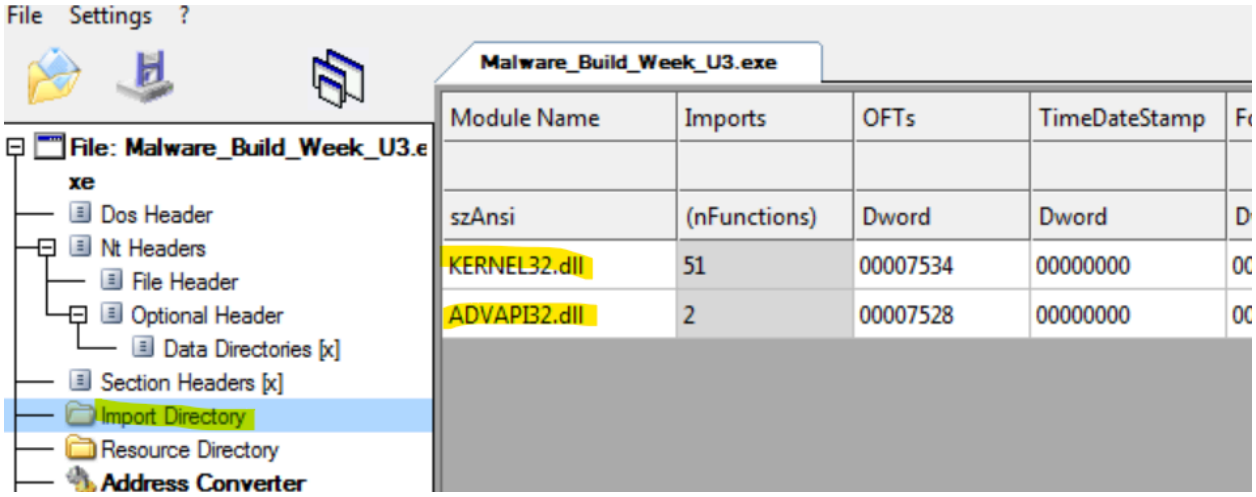
Per ottenere questa informazione, apriamo il tool *CFFExplorer,* che ci consente di ottenere informazioni sull'eseguibile e sugli *header.* Apriamo quindi il file e navighiamo sulla sezione *Section Headers.*



Le sezioni dell'eseguibile trovate sono:

- *.text*: sezione dell'eseguibile nel quale sono presenti le istruzioni che la CPU eseguirà. In questa sezione, in breve, sono contenute le righe di codice che compongono il malware. Informazioni trovate
  - La sezione ha un *entry point* alla locazione *00001487;*
  - I dati sulla Raw Size (dimensione su disco) e Virtual Size (dimensione in memoria) sono riportate nell'immagine di sopra;
- *.rdata*: in questa sezione sono presenti in genere le informazioni sulle librerie e le relative funzioni importate ed esportate dal malware (che analizzeremo in seguito). Dalle informazioni della sezione sappiamo che:
  - La import Directory (puntatore alle *.dll*): *000074EC;*
  - *Import Address Table Directory* (contiene gli indirizzi dei puntatori alle funzioni importate citate sopra): *00007000;*
  - I dati sulla Raw Size (dimensione su disco) e Virtual Size (dimensione in memoria) sono riportate nell'immagine di sopra;
- *.data*: in questa sezione sono presenti le variabili globali usate dal malware;
- *.rsrc*: parte dell'eseguibile in cui sono presenti risorse utili come asset grafici o stringhe.
  - La directory in cui queste sono presenti è alla locazione *0000C000.*
  - I dati sulla Raw Size (dimensione su disco) e Virtual Size (dimensione in memoria) sono riportate nell'immagine di sopra;

- ***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.***



Sempre su *CFFExplorer*, navighiamo nella sezione *Import Directory*. Troviamo due librerie:

- *KERNEL32.dll*, libreria base utilizzata per interagire con il sistema operativo che consente la gestione file e la manipolazione della memoria. Dalle funzioni invocate ipotizziamo che:
  - Il malware può **manipolare il filesystem** tramite le funzioni *CreateFile, ReadFile* e *WriteFile;*
  - Il malware può **importare risorse esterne** tramite le funzioni *GetProcAddress* and *LoadLibraryA;*
  - Può **manipolare la memoria** tramite le funzioni *VirtualAlloc, HeapAlloc, VirtualFree* e *HeapFree;*

- Può **gestire i processi** con funzioni come *ExitProcess, GetCurrentProcess* e *TerminateProcess.*

- *ADVAPI32.dll*, libreria che consente l'interazione con i registri di Windows.
  - Le funzioni presenti nel dettaglio di questa libreria sono *RegSetValueAEx* e *RegCreateKeyExA.* Questo suggerisce che il malware può tentare di ottenere la **persistenza** e quindi resistere ai reboot agendo sui valori delle chiavi di registro di Windows.

OFTs	FTs (IAT)	Hint	Name
00007528	00007000	000076AC	000076AE
Dword	Dword	Word	szAnsi
000076AC	000076AC	0186	RegSetValueExA
000076BE	000076BE	015F	RegCreateKeyExA

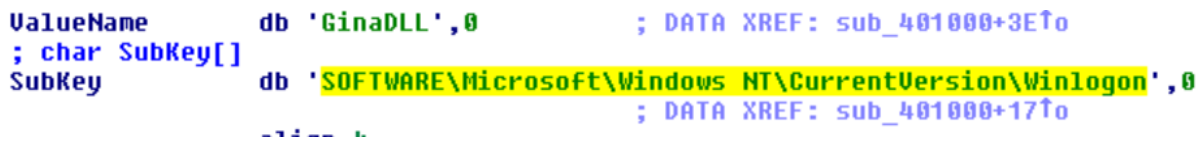
- ***Scopo della funzione chiamata alla locazione di memoria 00401021;***

Tramite IDA Pro ricerchiamo la locazione di memoria *00401021.* Scopriamo che la funzione chiamata è ***RegCreateKeyExA.***

```
.text:00401000
.text:00401000                push    ebp
.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\\CurrentVe"...
.text:0040101C                push    80000002h       ; hKey
.text:00401021                call    ds:RegCreateKeyExA
```

Questa funzione fa parte dell'API di Windows e consente di *creare* la chiave di registro *"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"* per aggiungere in seguito il valore *"GinaDLL"*. L'invocazione di questa funzione serve

per ottenere persistenza tramite la modifica del registro, dato che viene aggiunta una libreria *.dll* all'avvio del sistema.



- ***Come vengono passati i parametri alla funzione alla locazione 00401021;***

Essendo una STDCALL, i parametri sono passati alla funzione sullo stack tramite istruzioni *push* prima della *call* della funzione.

Che oggetto rappresenta il parametro alla locazione 00401017;

Il parametro alla locazione *00401017* è *"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon".* Tramite la modifica del suo valore viene aggiunta una libreria all'avvio del sistema. *Winlogon.exe* è un processo che implementa la funzione di Windows di gestione dei login e dei logout di Windows.

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

Le istruzioni comprese tra gli indirizzi 00401027 e 00401029 sono:

test eax, eax
jz short loc_401032

Queste istruzioni controllano se il registro EAX è 0 (quindi ZF = 1). L'istruzione *test* non modifica il contenuto del registro. Se lo *ZeroFlag* è valorizzato, il programma salterà alla locazione di memoria *loc_401032.* Riassumendo: Se EAX è zero, il programma salta all'indirizzo *loc_401032*, mentre se non è zero, continuerà con l'istruzione successiva senza saltare.

- ***Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C.***

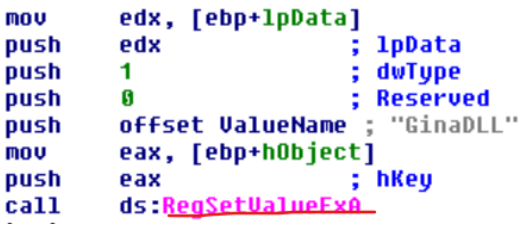
```c
int eax; //valore di eax



if (eax == 0) {

    RegSetValueExA(/*parametri*/);

} else {

    eax = 1;

    // salta alla locazione loc_40107B

}

}
```

In C il codice assembly è traducibile con un costrutto *if* che esegue la funzione *RegSetValueExA()* se il registro EAX è 0. Diversamente (nel blocco *else*), imposterà EAX a 1 e salterà alla locazione *loc_40107B* (vedi immagine sotto per contesto).

```
.text:00401029                 jz      short loc_401032
.text:0040102B                 mov     eax, 1
.text:00401030                 jmp     short loc_40107B
.text:00401032 ; ---------------------------------------------------------------------------
.text:00401032
.text:00401032 loc_401032:                             ; CODE XREF: sub_401000+29↑j
.text:00401032                 mov     ecx, [ebp+cbData]
.text:00401035                 push    ecx             ; cbData
.text:00401036                 mov     edx, [ebp+lpData]
.text:00401039                 push    edx             ; lpData
.text:0040103A                 push    1               ; dwType
.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
```

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

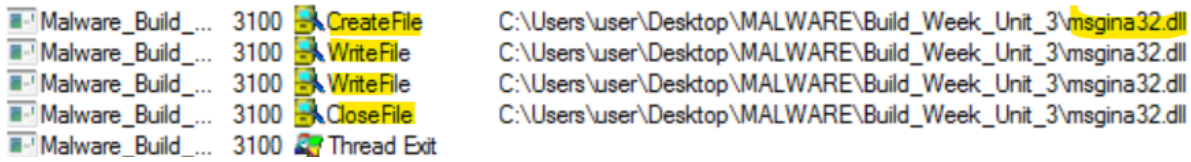
Il valore del parametro ***ValueName*** è ***GinaDLL***.



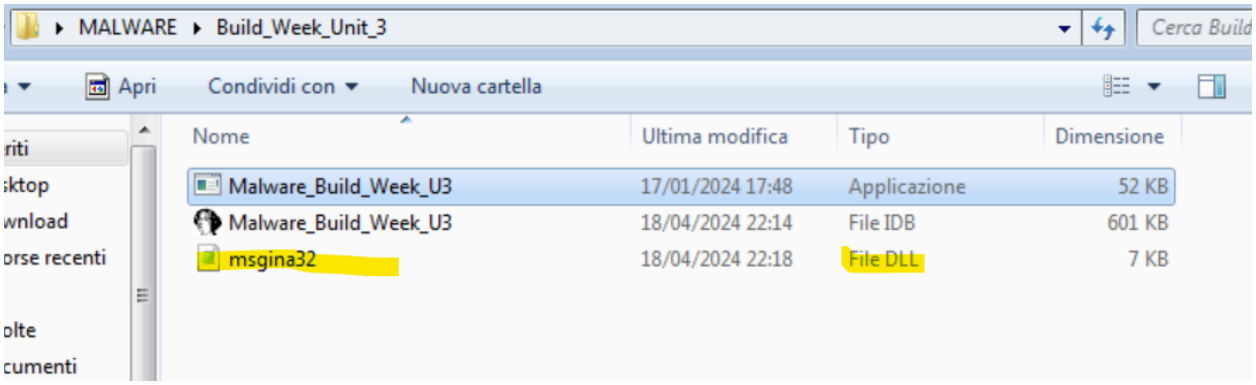
Analisi dinamica

- ***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.***

Tramite *ProcMon*, filtriamo per eventi nel filesystem. Notiamo che il malware ha creato un file chiamato *msgina32.dll.*

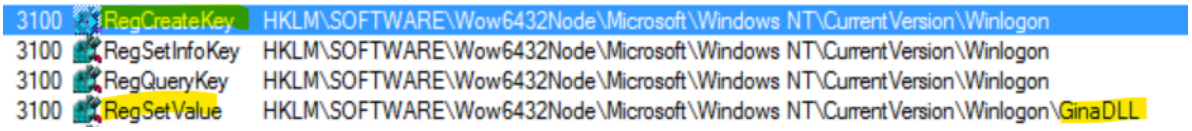


Verificando nella directory in cui c'è l'eseguibile, confermiamo la creazione del file.



Questo file è una libreria generata dal malware che, tramite *winlogon,* verrà caricata all'avvio del sistema (alla chiave di registro vista nella sezione dell'analisi statica corrisponde il *ValueName GinaDLL*).

- ***Quale chiave di registro viene creata?***



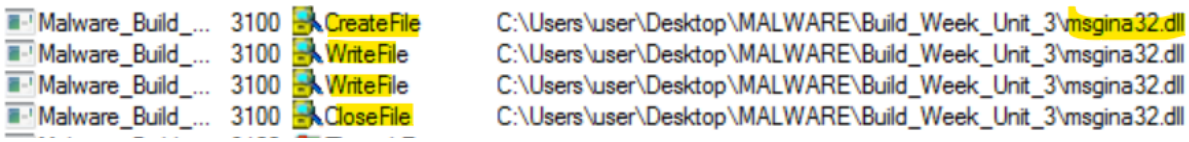
Con ProcMon confermiamo quanto discusso nella parte dell'analisi statica. Viene creata la chiave di registro *"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"* tramite l'istruzione *RegCreateKey.* Possiamo ottenere la stessa informazione utilizzando altri tool come *RegShot.* Per semplicità continueremo ad usare l'analisi effettuata con *ProcMon.*

- ***Quale valore viene associato alla chiave di registro creata?***

Alla chiave di registro creata viene settato il valore *GinaDLL* tramite l'istruzione *RegSetValue,* come visibile nell'immagine al punto precedente.
"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL"

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

CreateFile(), *WriteFile*() e *CloseFile*() sono le chiamate di sistema responsabili della creazione e modifica del file *msgina32.dll* presente nella cartella dopo l'esecuzione del malware. In questo caso, la call di sistema è la *CreateFile* dell'immagine di sotto che, in questo caso, ha creato un file *msgina32.dll* nuovo dato che prima dell'esecuzione del malware non esisteva.

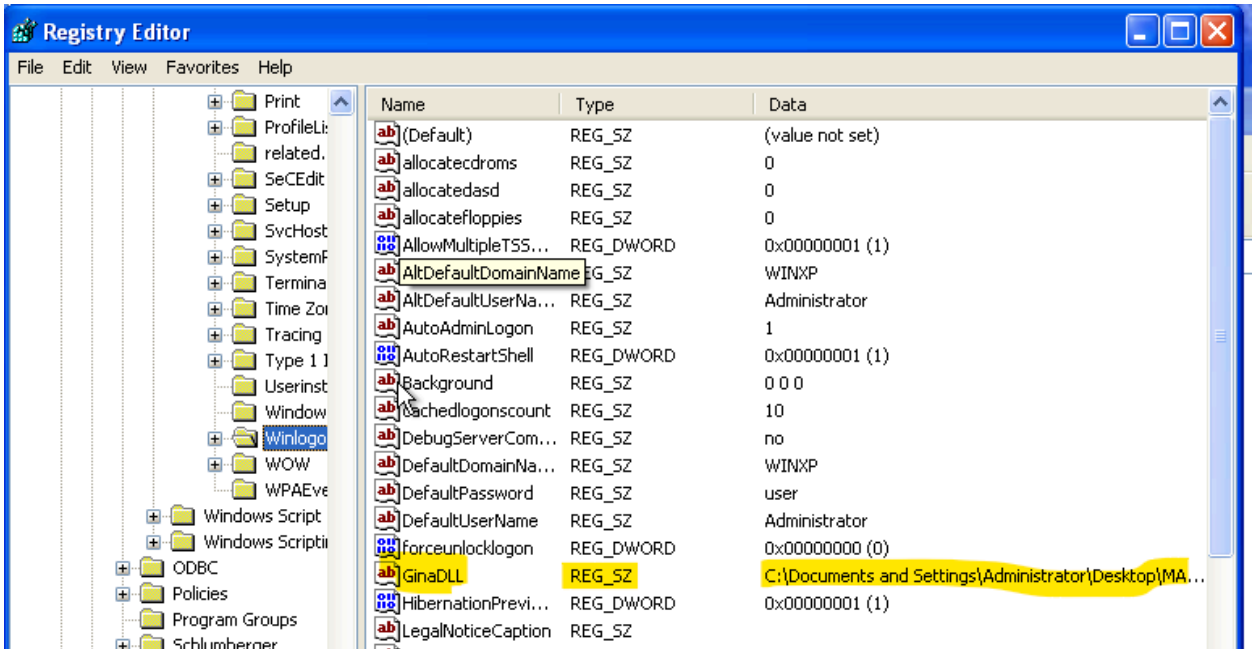


Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware.

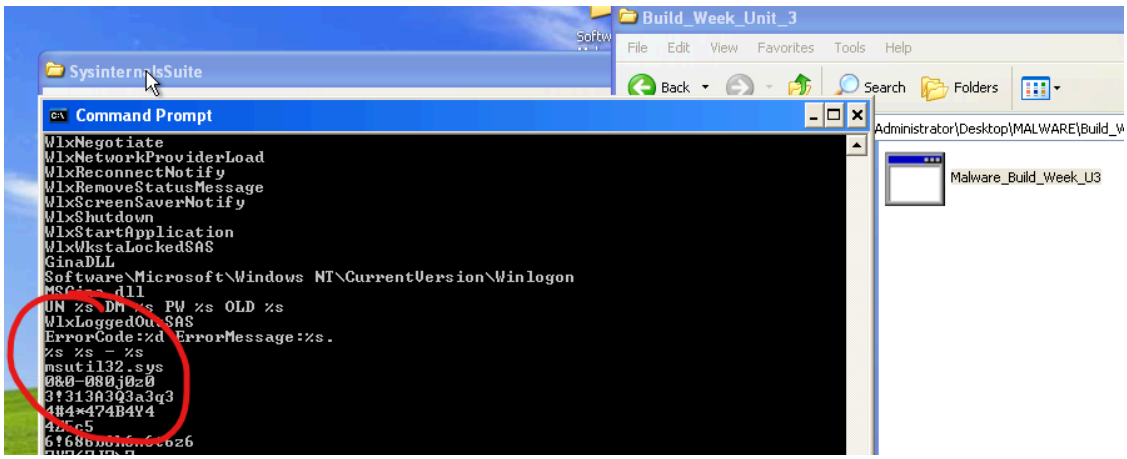
Prima dell'avvio del malware ho lanciato ***Wireshark***, ***ApateDNS*** e non ho rilevato ***nessun evento di rete***.

Caricando la libreria *msgina.32.dll,* il programma persiste all'avvio ed effettua una ***GINA Interception***. E' una tecnica utilizzata per **rubare le credenziali di login degli utenti** e salvandole in un file. Ho notato che il Malware non è persistente sulla macchina Windows 7, in quanto al riavvio il valore della chiave di registro non è presente. Tuttavia su Windows XP funziona. L'esito del test effettuato su questa macchina è il seguente:

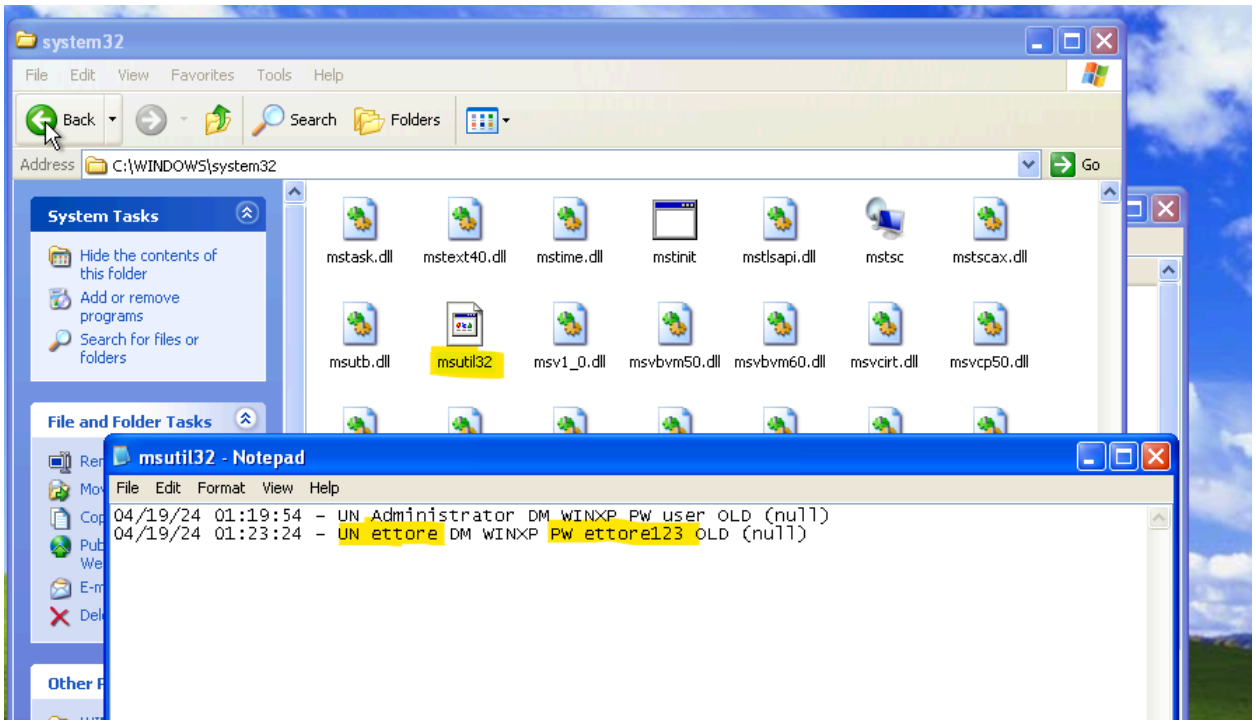
- Dopo aver lanciato il malware e riavviato il sistema, il valore *GinaDLL* è ancora presente.



- Lanciamo il comando strings alla ricerca di stringhe interessanti che possono darci un indizio sul salvataggio delle credenziali. Troviamo una stringa interessante, *msutil132.sys.*



- Testiamo il malware, effettuando il logoff e il login. Notiamo che andando ad ispezionare il file *msutil32.sys* troviamo le credenziali di accesso salvate.
    - *Username*: *ettore*
    - *Password*: *ettore123*



Link utili:

https://learn.microsoft.com/en-us/windows/win32/secauthn/loading-and-running-a-gina-dll

https://neuralfeed.wordpress.com/2014/07/15/gina-interception-stolen-credentials-and-the-malware-that-almost-got-away/