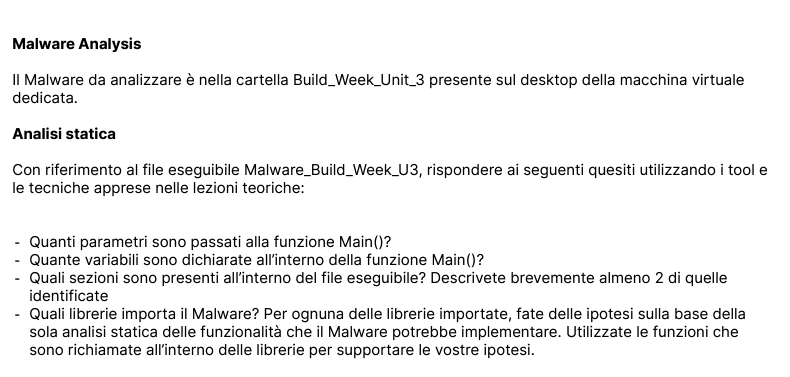
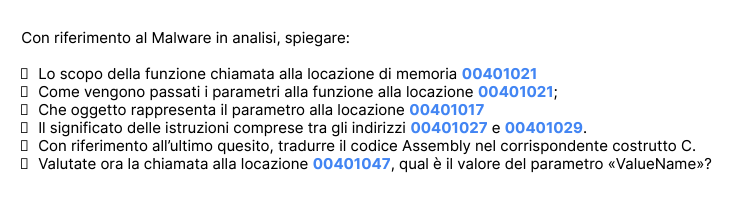
8/2/2024

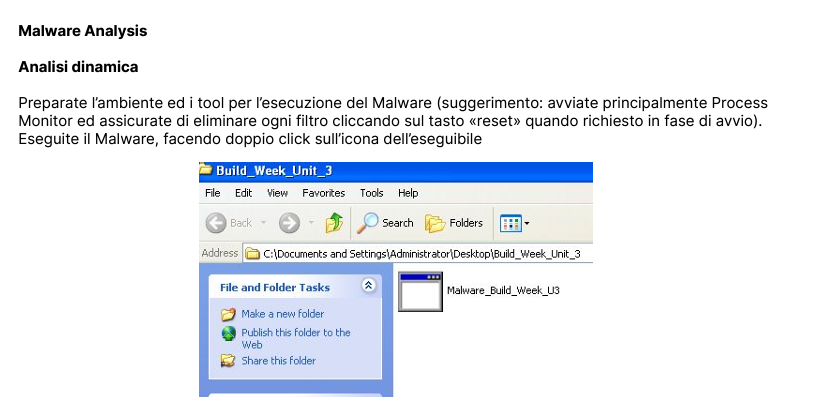
Michele Garzoni

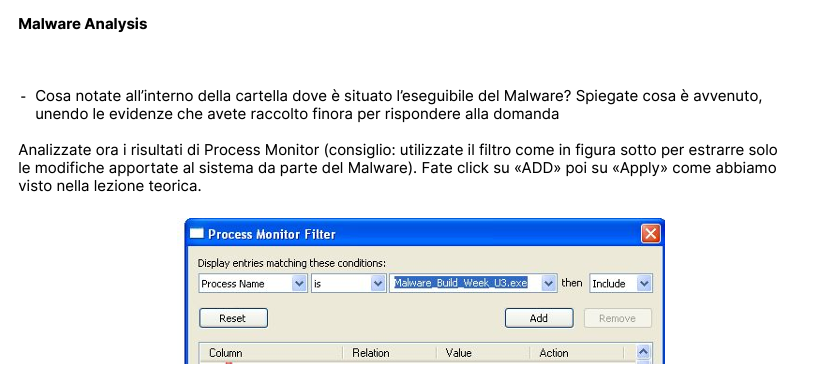
W24D4

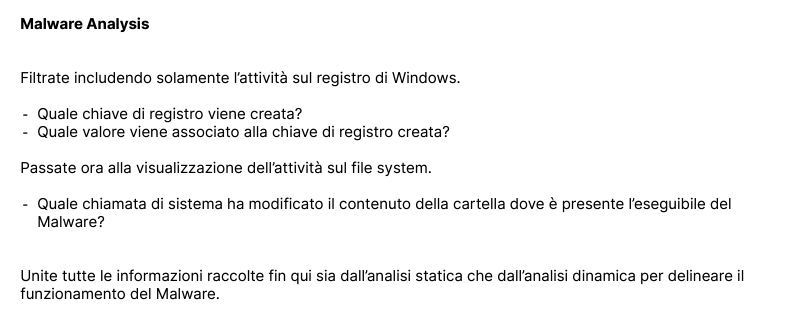
Progetto mese 6











L’esecuzione dell'esercizio richiede la combinazione di varie analisi viste durante il modulo 6 del corso.

Per questa prima parte servirà l'applicazione dell'anali-si statica basica ed avanzata.

L'analisi statica si riferisce all'ispezione del codice sorgente o del codice binario di un programma (in questo caso, un malware) per identificarne la funzionalità, le caratteristiche e le potenziali minacce senza eseguirlo, questo approccio si contrappone all'analisi dinamica, dove il codice viene eseguito in un ambiente controllato (sandbox) per osservare il suo comportamento.

L’analisi statica basica consiste nell’esaminare un eseguibile senza vedere le istruzioni che lo compongono e la sua funzione è confermare se un dato file è malevolo e fornire informazioni generiche circa le sue funzionalità. L'analisi statica avanzata presuppone la conoscenza dei fondamenti di «reverse-engineering» al fine di identificare il comportamento di un malware a partire dall’analisi delle istruzioni che lo compongono. Questo passaggio è essenziale per capire esattamente cosa fa il malware a livello di istruzioni della cpu.

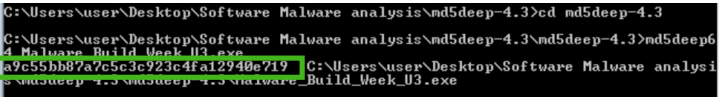
Si possono inoltre estrarre stringhe di testo, url, chiavi di cifratura, e altre risorse dal codice del malware, che possono indicare il suo comportamento o intento e se ne può esaminare il codice relativo alla rete per comprendere come il malware comunica.

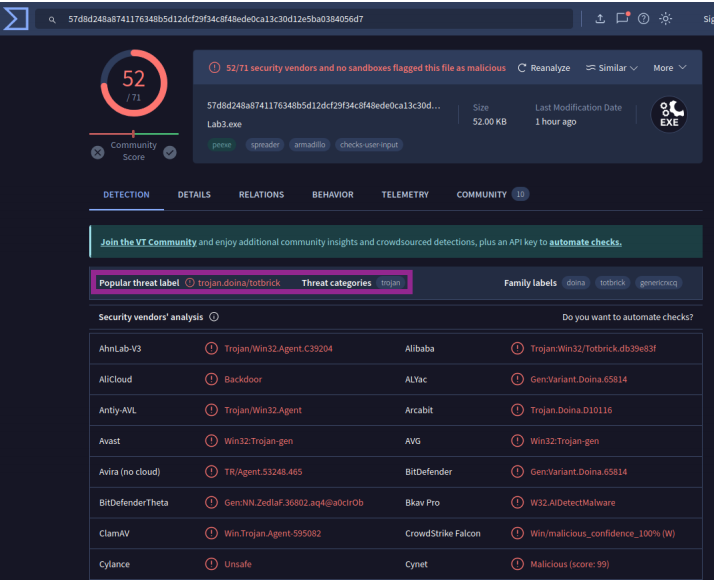
Prima di procedere all'analisi mi assicuro che sia di fatto un malware, estraendone l'hash con md5deep e controllando su virustotal la sua reputazione che si basa su vari riscontri di software antivirus.

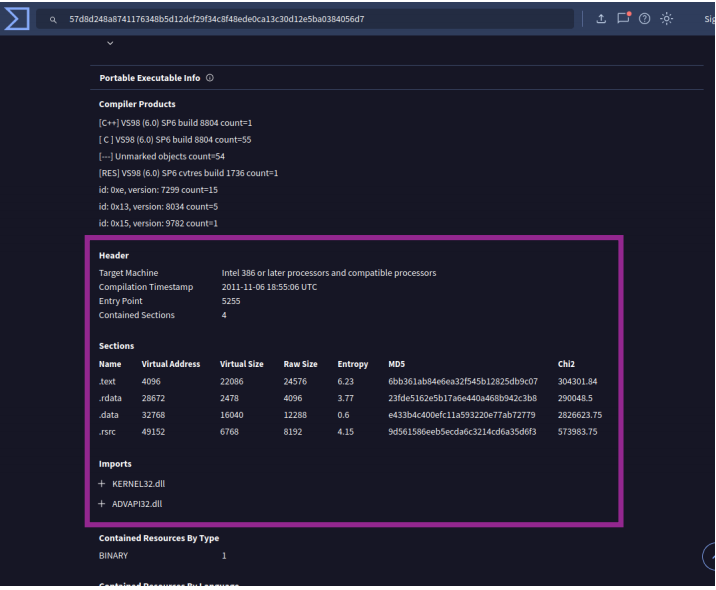
In questa prima analisi posso vedere che il virus è noto, si tratta di un malware di tipo trojan compilato in data 11-06-2011 in c++, analizzato l'ultima volta in data 17 aprile 2024. è un malware progettato per colpire la macchina intel 386 e processori sucessivi/compatibili. ha 5255 entry points, 4 sezioni ed importa 2 librerie:

**kernel32.dll** - una delle librerie fondamentali di windows, contiene numerose funzioni che gestiscono la memoria, i processi e i thread. i malware la utilizzano per manipolare i processi e per accedere a diverse api di sistema ed ottenere persistenza.

**advapi32.dll** - fornisce funzioni relative alla sicurezza e alla gestione di account, che i malware possono sfruttare per modificare permessi, accedere a token di sicu-rezza e alterare il registro di sistema, ad esempio per essere avviati all'avvio del sistema operativo. andrò a confermare tutti questi dati tramite tool dedicati come cffexplorer/exeinfope e ida pro.







CFF EXPLORER

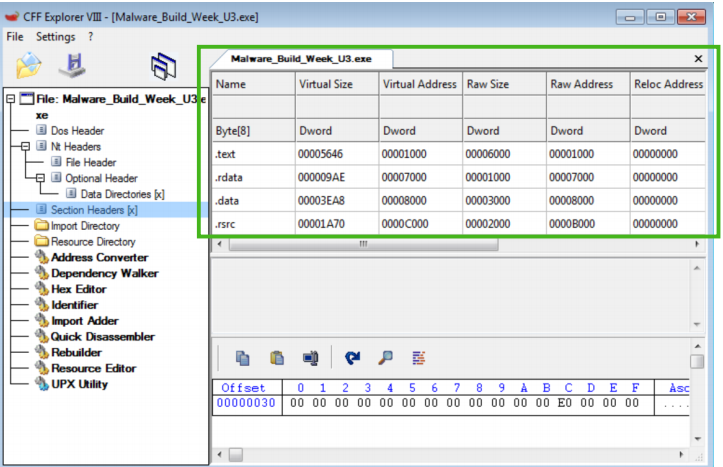
Per controllare le funzioni importate ed esportate da un malware, si può utilizzare cff explorer, un tool da installare sulle macchine virtuali dedicate all’analisi dei malware. Se mi sposto su «import directory» posso controllare le librerie e le funzioni importate, mentre su «section headers» posso vedere le sezioni presenti all'interno del file eseguibile:

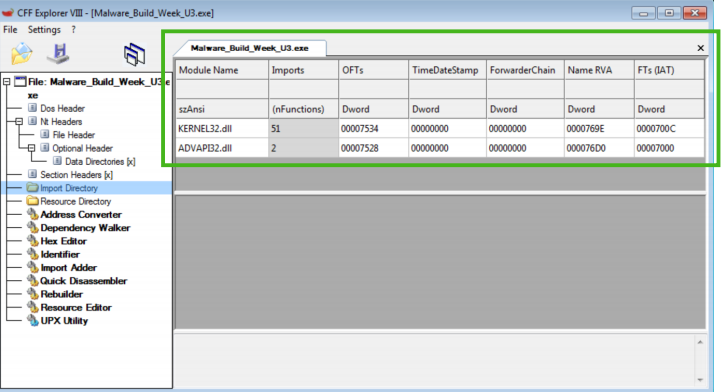
.text - contiene le righe di codice, istruzioni, che la cpu eseguirà all'avvio del malware.

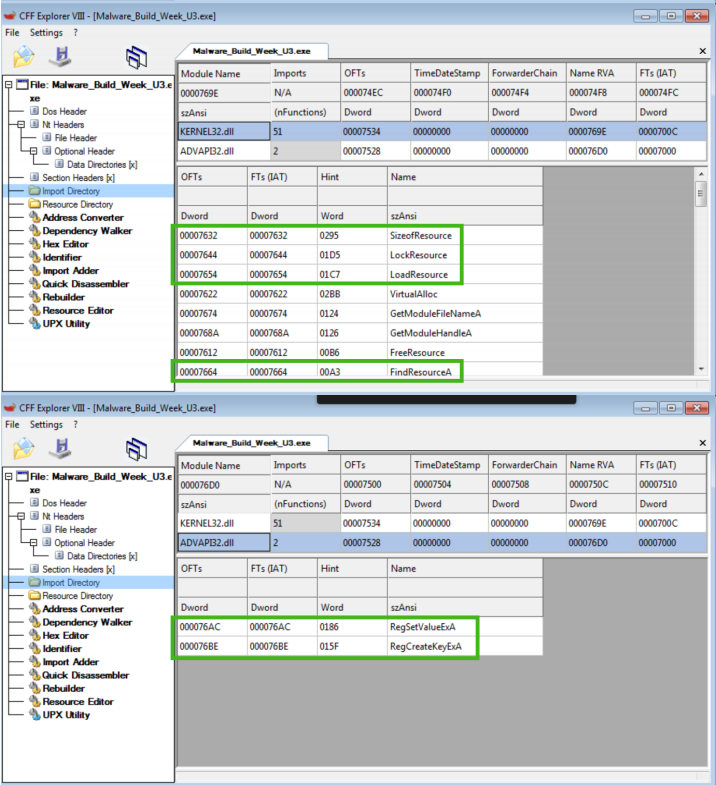
.rdata - sezione che include le informazioni circa le librerie e le funzioni importate ed esportate dall’eseguibile, dati che il programma legge mentre è in funzione.

.data - contiene i dati / le variabili globali del program-ma eseguibile, che devono essere disponibili da qualsiasi parte del programma e possono essere modificati.

.rsrc - include le risorse utilizzate dall’eseguibile come ad esempio icone, immagini, menu e stringhe che non sono parte dell’eseguibile stesso.



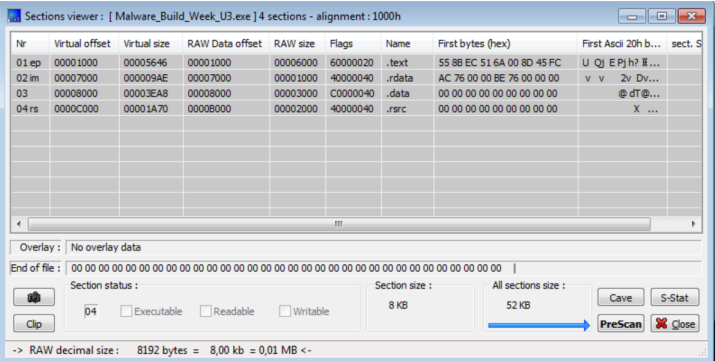


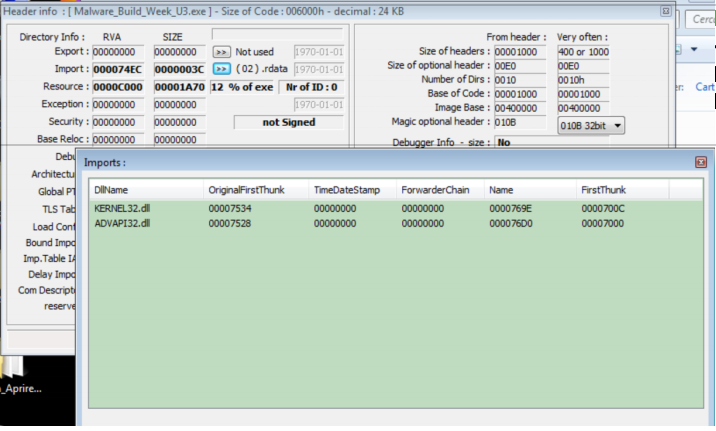


Per quel che riguarda le due librerie importate, che sono kernel32.dll (1) e advapi32.dll (2), posso ipotizzare che il malware cerchi di ottenre persistenza e modificare le chiavi di registro (regsetvalueexa/regcreatekeyexa) per poter essere eseguito in autonomia. Inoltre la presenza di funzioni come sizeofresource/lockresource/loadre-source/findresourcea fa presupporre che sia un dropper, cioè un malware che contiene al suo interno un altro malware, ed utilizza queste api che permettono di localizzare all’interno della sezione «risorse» il malware da estrarre, e successivamente da caricare in memoria per l’esecuzione.

EXEINFOPE

Si confermano gli stessi dati di CFF





IDAPRO

Ora per informazioni su variabili locali e parametri della funzione main (), userò ida pro (interactive disassembler professional) che è uno strumento avanzato di reverse engineering software che offre capacità di disassemblaggio, debugging e analisi statica.

Variabili locali: le variabili locali in una funzione assembly sono tipicamente allocate nello stack. Questo è spesso fatto attraverso istruzioni di tipo push all'i-nizio di una funzione o con un'istruzione sub che aumenta il puntatore dello stack (sp) per creare spazio.

Parametri: solitamente i parametri sono passati ai regi-stri o attraverso l'uso dello stack prima della chiama-ta della funzione. I commenti nel codice possono forni-re indicazioni su dove e quanti parametri sono passati.

Nell'assembly, le etichette che iniziano con var\_ tendono a indicare variabili locali, mentre quelle che iniziano con arg\_ indicano argomenti o parametri passati alla funzione.

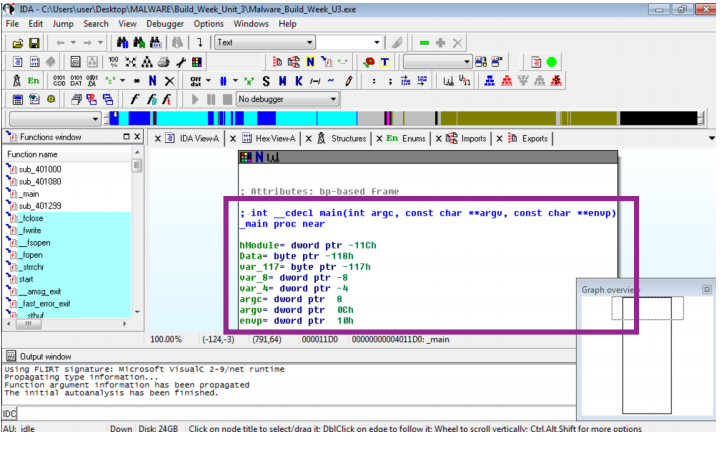
Valori offset: gli offset (come var\_54h) sono utilizzati per accedere a dati specifici sullo stack. Gli offset negativi rispetto all'indirizzo base del frame della funzione (ad esempio ebp su x86) di solito indicano variabili locali, mentre gli offset positivi indicano parametri.

In questo caso le variabili locali in evidenza che possiamo rilevare ad un'occhiata sono 5:

1. hmodule
2. data
3. var\_117
4. var\_8
5. var\_4.

Mentre i parametri sono 3

1. argc
2. argv
3. envp.



Proseguendo con l'analisi statica del codice del malware grazie all'utilizzo di ida pro, troviamo alla locazione di memoria 00401021 la funzione regcreatekeyexa, che è una funzione nota che fa parte delle api di windows che permettono alle applicazioni di interagire con il registro di windows.

Queste funzioni sono utilizzate per creare nuove chiavi di registro o per aprire chiavi esistenti per modificare i loro valori. Nello specifico il malware può utilizzare regcreatekeyexa per ottenere persistenza creando nuove chiavi di registro o modificando chiavi esistenti ed assicurandosi che il codice malevolo venga eseguito ogni volta che il sistema viene avviato; per esempio, potrebbe aggiungere una voce nella chiave run per eseguire automaticamente il malware all'avvio del sistema.

Per quel che riguarda il metodo in cui vengono passati i parametri alla suddetta funzione, nel modulo si è visto che la convenzione di chiamata più comune in molti sistemi operativi quando si utilizza l'architettura x86 è «pushare» i parametri sullo stack prima della chiamata (call) alla funzione regcreatekeyexa. I parametri verranno letti dalla funzione in ordine inverso, quindi a partire da hkey per finire con lpdwdisposition.



Troviamo alla locazione di memoria 00401017 - push offset subkey - l'indirizzo della sottochiave di registro che viene spinto nello stack. Il nome vicino al codice indica che potrebbe essere il nome della sottochiave che verrà creata dal processo hkey in winlogon -> componente del sistema operativo windows che si occupa della gestione della sessione di accesso (login) e disconnessione (logout) degli utenti.



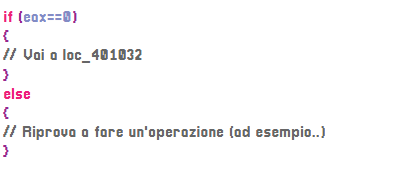
Per quel che riguarda il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029 vediamo:

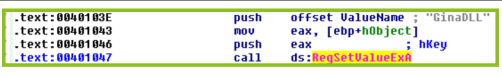
Un'istruzione condizionale test, che è simile all’istruzione and logico bit a bit, ma non va a modificare il contenuto degli operandi (cioè eax e se stesso). Modifica invece il flag zf (zero flag) del registro eflags, che viene settato ad 1 se e solo se il risultato dell’and è 0. (0 and 1 = 0 \* 1 = 0 -> zeroflag = 1). Viene di fatto utilizzato per controllare se un valore è zero o meno. Se test è zero, lo zf è 1.

Un conditional jump di tipo jz, che nel flusso di controllo salta ad una determinata locazione di memoria (in questo caso 00401032) se zf è pari a uno. Ed a meno che il valore contenuto nel registro eax non sia 0, il salto non avverrà.

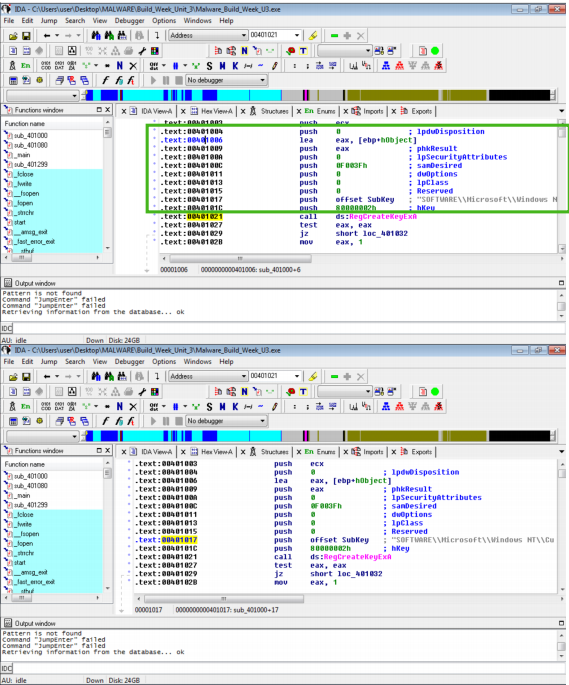


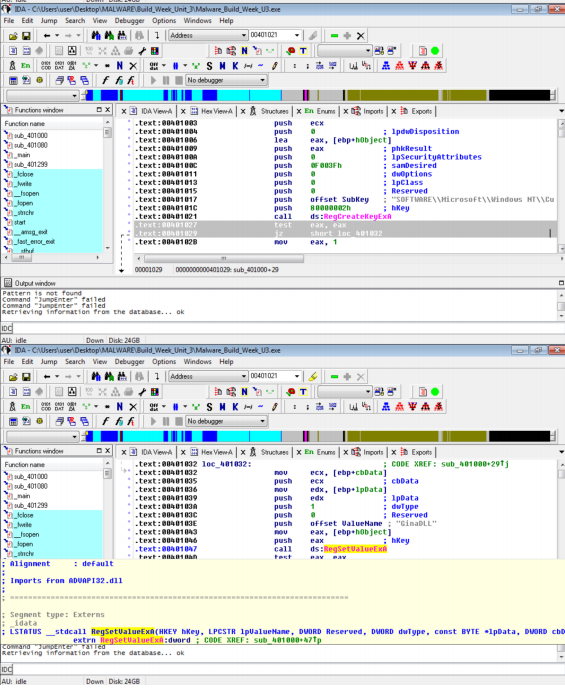
Questa operazione equivale ad un ciclo if in c come quello seguente (qui, eax rappresenta una variabile in c che contiene il valore che era nel registro eax):



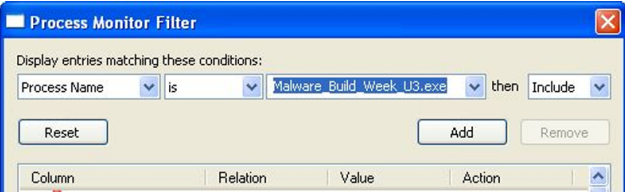


Per quel che riguarda la chiamata alla locazione 00401047 posso vedere che si tratta di una call alla funzione regsetvalueexa, che è una funzione dell'api di windows che imposta il valore di una voce nel registro di sistema. Il prefisso ds: indica che l'indirizzo della funzione è preso dal registro ds, che è il segmento dati. Questa funzione, quindi, impartisce istruzioni affinchè offset valuename abbia valore "**ginadll**" in una specifica chiave di registro identificata da hkey.





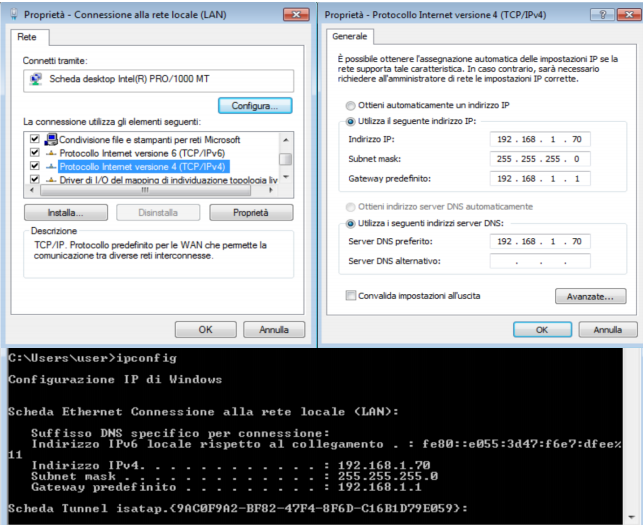
PROCMON

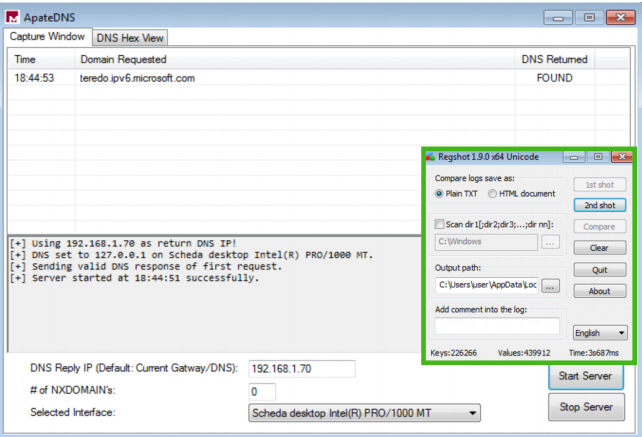


Ricreo un'istantanea da virtualbox della macchina windows prima di iniziare, per poter ripristinare in caso di problemi, e visto che andrò ad eseguire il malware mi assicuro di rispettare i seguenti accorgimenti:

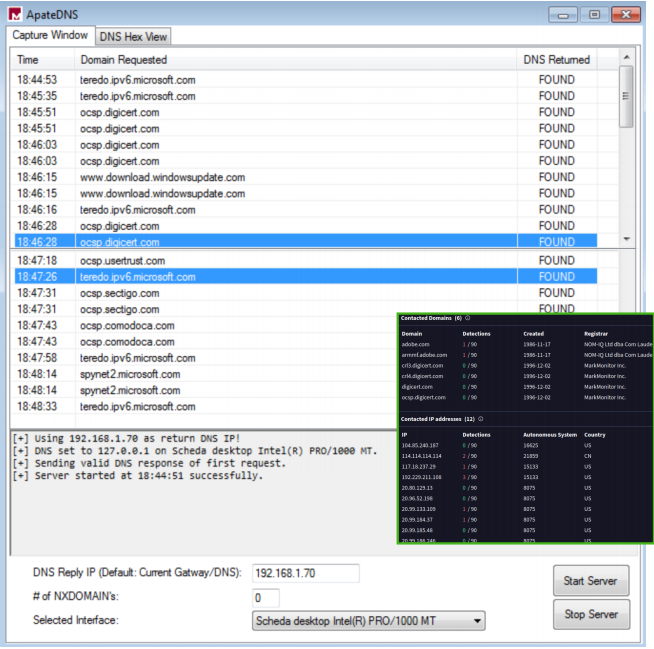
* 1. Disattivare controller usb
  2. Disattivare comunicazione con la rete (solo int)
  3. Disabilitare la condivisione delle cartelle
  4. Disabilitare appunti condivisi (copia / incolla)

Prima di aprire il malware, mi assicuro di andare a catturare tutti gli eventi aprendo procmon, al fine di identificare eventuali azioni del malware su processi e thread, e modifiche registro; avvio regshot per confrontare tramite screenshot (prima / dopo) le modifiche che avverranno a livello di sistema, ed imposto un ip statico per vedere tramite apatedns le chiamate che il malware andrà a fare nel web (quali siti). Avendolo isolato dalla rete ovviamente non potrà eseguire tutte le sue funzioni.





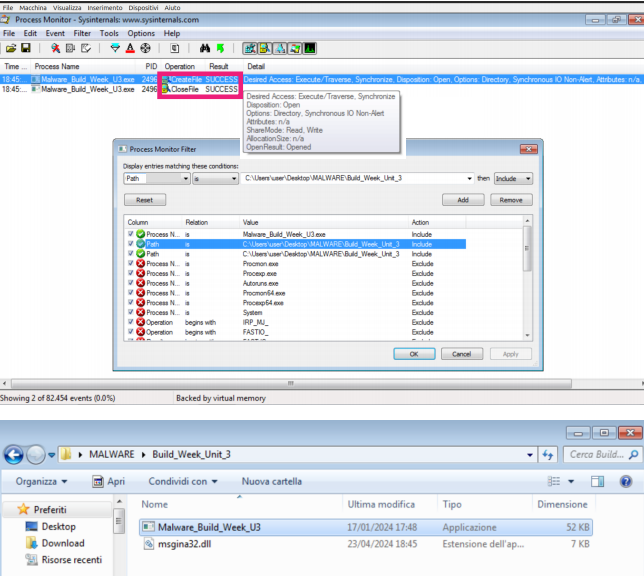
Di seguito alcune delle chiamate intercettate:

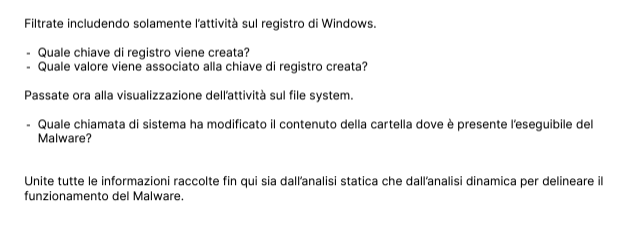


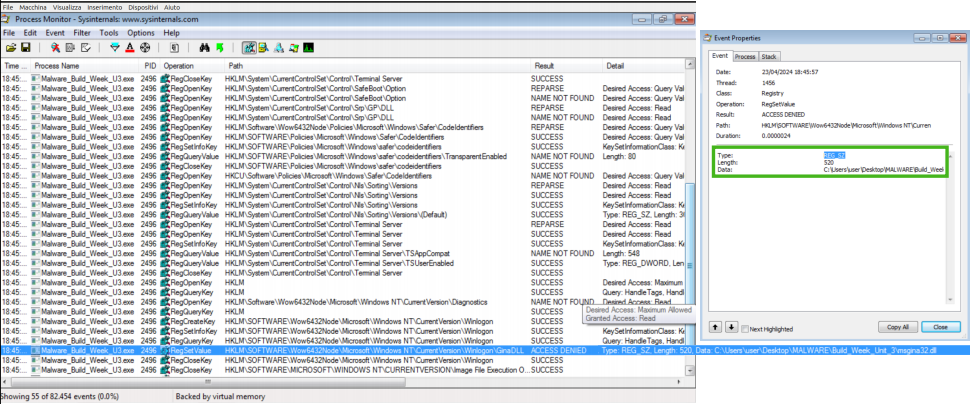
Vediamo subito che all'interno della cartella dove è situato l'exe del malware viene creato un file, ovvero msgina.dll (acronimo di "microsoft graphical identification and authentication") che gestisce il processo di accesso, e nello specifico l'interfaccia utente di logon interattiva, che include la schermata di accesso classica di inserimento nome utente / password.

Funziona nel contesto del processo winlogon e viene caricato all'inizio del processo di avvio del sistema, è responsabile di fornire procedure personalizzabili per l'identificazione e l'autenticazione degli utenti. Ed è proprio andando a modificare questo file che il malware ottiene persistenza all'avvio del sistema operativo.

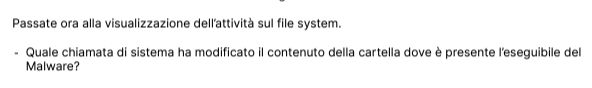
Filtrando per path su procmon vediamo la creazione del file nella cartella scelta tramite processo "createfile".



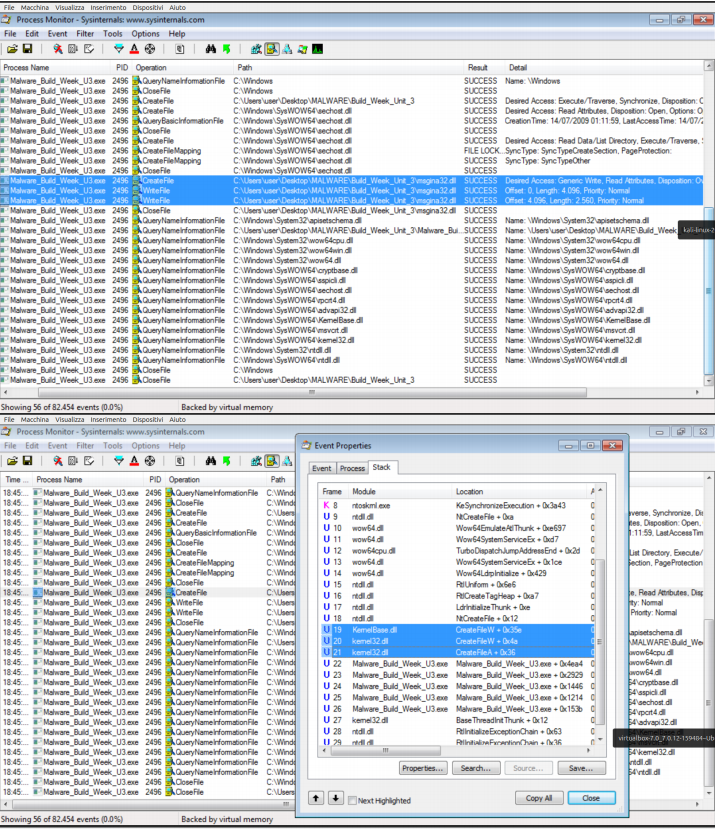


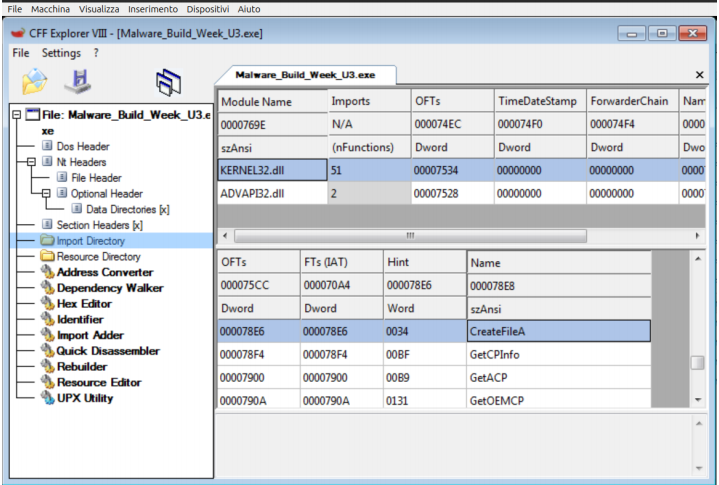


Filtrando per chiavi di registro si può notare che ad un certo punto viene passata la funzione regsetvalue che è una funzione tipica della libreria importata dal malware advapi32.dll che, come detto in precedenza, fornisce funzioni relative alla sicurezza e alla gestione di account, che i malware possono sfruttare per modificare permessi, accedere a token di sicurezza e alterare il registro di sistema. Posso vedere la chiave di registro reg\_sz (identificata da hkey) a cui, sempre come visto in precedenza, viene assegnato il valore di gina.dll. Questo valore "malevol0" verrà sovrascritto nella libreria msgina32.dll andandone sostituire la copia "sana". Il malware in tal modo potrà avviarsi ad ogni login nel contesto del processo winlogon.

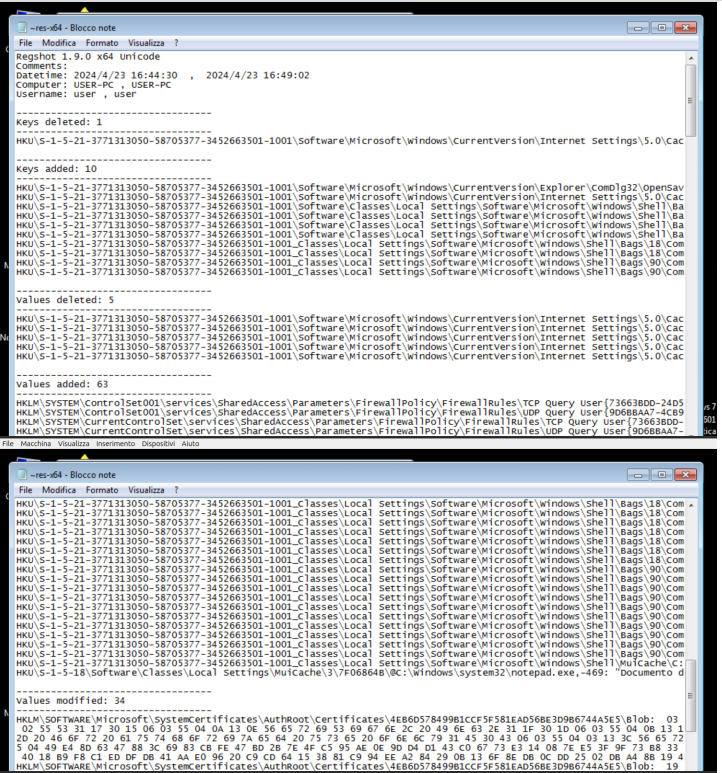


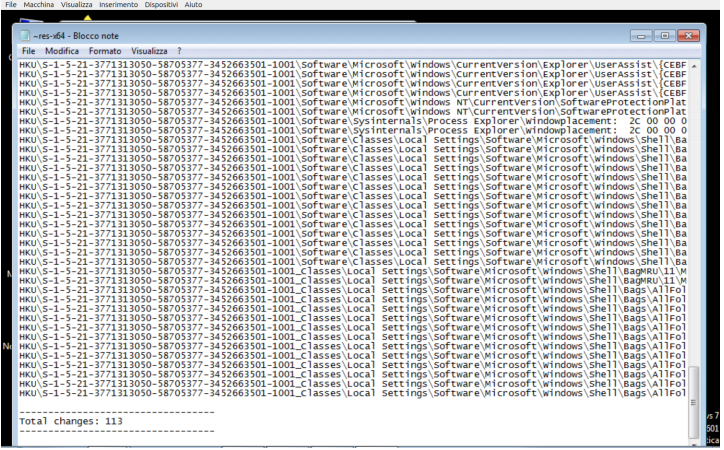
Filtrando per attività sul file system vedo una createfile (funzione che modifica un file esistente, o se non esiste ne crea uno nuovo) al file di msgina32.dll. Questa funzione è tipica della libreria kernel32.dll, che appunto viene evocata dal malware.





Dopo un confronto ottenuto tramite regshot posso vedere che sono stati modificati 113 elementi tra cui 10 chiavi aggiunte, 1 cancellata; valori aggiunti 63, cancellati 5, modificati 34.





In conclusione sommando i dati ottenuti dalla'analisi statica + quella dinamica posso evincere che:

* Il malware è un trojan/dropper
* All'avvio genera un file "sporco" nella cartella del suo eseguibile
* Cerca di fare un privilege escalation ed ottenere "poteri" amministrativi
* Ottiene persistenza (esecuzione ad ogni avvio del sistema operativo windows) modificando una chiave di msgina32.dll responsabile del login nel contesto del processo winlogon
* Evade la difesa infiltrandosi in un processo comune
* Può ottenere l'accesso alle credenziali degli utenti
* Può raccogliere dati (che salva in un file chiamato msutil32.sys -> si può vederne la presenza con l'utility strings) e monitorare le utenze



