

PROGETTO MODULO 6

Traccia:

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.

Esecuzione:

Analisi statica

- Quanti parametri sono passati alla funzione Main()?

```
.text:004011D0 argc      = dword ptr 8
.text:004011D0 argv     = dword ptr 0Ch
.text:004011D0 envp     = dword ptr 10h
```

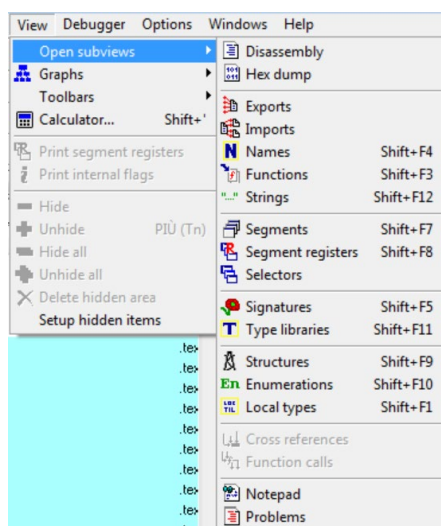
Aprendo IDA Pro per visionare il codice Assembly presente nel Malware in analisi, i parametri sono quelli riportati in figura. Essi sono parametri in quanto l'offset è positivo rispetto il puntatore EBP (Base Pointer) e sono di tipo Intero (DWORD) senza segno. Il fatto che siano positivi lo si può notare dai valori in verde alla destra del parametro.

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

```
.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
```

Al contrario dei parametri detto in precedenza, le variabili della funzione Main() hanno un offset negativo rispetto al puntatore EBP, quindi si trova rispettivamente prima del Base pointer. Il simbolo "-" presente su ogni numero di ogni variabile ne riporta l'offset.

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



Per identificare le sezioni utilizzate dal Malware in analisi, entro su View->Open Subviews->Segments. In automatico, IDA Pro apre la schermata seguente.

| Name | Start | End | R | W | X | D | L | Align | Base | Type | Class | AD | es | ss | ds | fs | gs |
|--------|------------------|------------------|---|---|---|---|---|-------|------|--------|-------|----|------|------|------|--------|--------|
| .text | 0000000000401000 | 0000000000407000 | R | . | X | . | L | para | 0001 | public | CODE | 32 | 0000 | 0000 | 0003 | FFF... | FFF... |
| .idata | 0000000000407000 | 00000000004070DC | R | . | . | . | L | para | 0002 | public | DATA | 32 | 0000 | 0000 | 0003 | FFF... | FFF... |
| .rdata | 00000000004070DC | 0000000000408000 | R | . | . | . | L | para | 0002 | public | DATA | 32 | 0000 | 0000 | 0003 | FFF... | FFF... |
| .data | 0000000000408000 | 000000000040C000 | R | W | . | . | L | para | 0003 | public | DATA | 32 | 0000 | 0000 | 0003 | FFF... | FFF... |

Delle sezioni riconosciute dal software, prendo in analisi le sezioni come richiesto dalla traccia, riportando una breve descrizione delle stesse:

- **.text:** Questa sezione del codice sorgente del Malware riporta le istruzioni del programma, ovvero il codice eseguibile.
 - **.rdata:** Questa sezione del codice sorgente del Malware contiene i dati di sola lettura del programma. Essi sono accessibile durante l'esecuzione dello stesso, ma non possono essere modificati. Utile per memorizzare delle costanti.
 - **.data:** Questa sezione del codice sorgente del Malware contiene le variabili globali e i dati modificabili durante l'esecuzione del programma.
- **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.**

| Address | Ordinal | Name | Library |
|-----------|---------|--------------------|----------|
| 000000... | | RegSetValueExA | ADVAPI32 |
| 000000... | | RegCreateKeyExA | ADVAPI32 |
| 000000... | | SizeofResource | KERNEL32 |
| 000000... | | LockResource | KERNEL32 |
| 000000... | | LoadResource | KERNEL32 |
| 000000... | | VirtualAlloc | KERNEL32 |
| 000000... | | GetModuleFileNameA | KERNEL32 |
| 000000... | | GetModuleHandleA | KERNEL32 |
| 000000... | | FreeResource | KERNEL32 |
| 000000... | | FindResourceA | KERNEL32 |
| 000000... | | CloseHandle | KERNEL32 |
| 000000... | | GetCommandLineA | KERNEL32 |
| 000000... | | GetVersion | KERNEL32 |
| 000000... | | ExitProcess | KERNEL32 |
| 000000... | | HeapFree | KERNEL32 |
| 000000... | | GetLastError | KERNEL32 |
| 000000... | | WriteFile | KERNEL32 |
| 000000... | | TerminateProcess | KERNEL32 |
| 000000... | | GetCurrentProcess | KERNEL32 |

Vado su **Imports** di IDA Pro, è possibile vedere le librerie importate dal Malware per l'esecuzione. In totale esso usa n.2 librerie importanti, essere le seguenti:

- **ADVAPI32:** libreria che contiene funzioni per interagire con i servizi ed i registri del sistema operativo.
- **KERNEL32:** contiene funzioni principali per la manipolazione di file e memorie.

Queste librerie contengono funzioni importanti utilizzate dal sistema operativo (Microsoft) che ne permettono il regolare funzionamento e la gestione delle attività e dei processi ordinari. Prendendo anche visione delle funzionalità che il Malware cita (es. RegSetValueAX, GetModuleFileNameA etc...), ipotizzo che il programma, una volta che viene avviata la PE stessa, tenti di creare oppure di modificare chiavi di registro del sistema e che crei un file associato a tale chiave. Ciò permetterebbe al Malware stesso di insediarsi nel sistema e di avviarsi ad ogni accensione del PC. Se ben configurato, potrebbe non solo avviarsi quando è l'utente infetto ad accedere, ma ad ogni utente che esegue l'accesso in quella determinata macchina. Ipotizzo che sia solo anche a fine di esecuzione della macchina e che non presenti alcun tipo di Backdoor, in quanto non vedo librerie come **Wsock32.dll** citate che permettono la creazione di socket oppure librerie connessioni a server o macchine remote. In aggiunta, credo si tratti di tipo di chiamate alle funzioni di tale librerie a tempo di esecuzione, in quanto la PE non è molto pesante e nel codice analizzato non presenta la libreria intera.

Con riferimento al Malware in analisi, spiegare:

- Lo scopo della funzione chiamata alla locazione di memoria 00401021

```

.text:00401015      push    0                ; lpSecurityAttributes
.text:00401017      push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
.text:0040101C      push    80000002h        ; hKey
.text:00401021      call    ds:RegCreateKeyExA
.text:00401027      test    eax, eax
.text:00401029      jz      short loc_401032
.text:0040102B      mov     eax, 1
.text:00401030      jmp     short loc_40107B
.text:00401032 ; -----

```

Lo scopo di tale chiamata alla funzione **RegCreateKeyExA** permette al programma in esecuzione di creare/modificare una chiave di registro del sistema utilizzando i parametri caricati nello stack.

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

```

.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

```

I parametri vengono caricati precedentemente nello **stack** tramite una serie di **push**, dove ogni push eseguito contiene un determinato valore e corrisponde ad ogni singolo parametro che la funzione **RegCreateKeyExA** necessita per ricercare la chiave o crearne una nuova.

- Che oggetto rappresenta il parametro alla locazione 00401017

```

.text:00401017      push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...

```

Essa rappresenta il nome della sottochiave del registro che si desidera creare o aprire. Quello che viene riportato alla destra di tale comando è il percorso dove è presente (o verrà creata) la chiave di registro richiesta. Tale chiave verrà eseguita a seconda di come è configurato **hKey**, che rappresenta a quale chiave di registro principale dovrà fare riferimento questa sottochiave (come ad esempio HKEY_LOCAL_MACHINE, in tal caso farebbe parte delle impostazioni comuni per tutti gli utenti che accedono a quella macchina).

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

```

.text:00401027      test    eax, eax
.text:00401029      jz      short loc_401032

```

Queste due istruzioni Assembly significa che il programma effettua un di controllo AND fra il registro EAX e se stesso, non modificando i valori contenuti in esso ma, in base al risultato dell'operatore logico, setta il Zero Flag in base al risultato. Ovvero Zero Flag = "1" se la condizione del test riporta un zero, Zero Flag = "0" se il risultato del test riporta un 1 (quindi non è uguale). Il comando successivo **jz short loc_401032** esegue un salto all'indirizzo di memoria citato se il flag risulta 1, altrimenti continua per l'indirizzo successivo a quest'ultimo. Trattasi di **jump condizionale**.

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

Prendendo quanto citato in precedenza, il suo costrutto sarà un **IF** che ipotizzo essere il seguente:

```
if (a == 0) {
```

```
(istruzioni presenti alla loc_401032)
```

```
} else {
```

```
(istruzioni consecutive all'indirizzo successivo, ovvero loc_40102°)
```

```
}
```

Dove la variabile **a** corrisponde al valore contenuto nel registro **EAX** in Assembly.

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

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

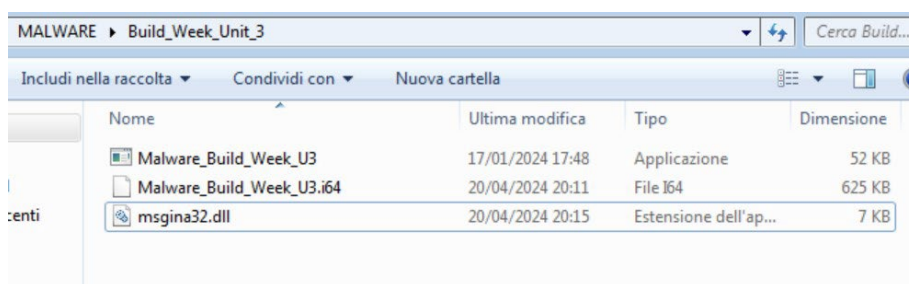
Il valore impostato del parametro **ValueName** è "GinaDLL". Ciò significa che una volta raggiunto l'handle della chiave di registro, il nome di tale registro sarà "GinaDLL".

Analisi dinamica

Preparate l'ambiente ed i tool per l'esecuzione del Malware. 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.



| Nome | Ultima modifica | Tipo | Dimensione |
|---------------------------|------------------|-----------------------|------------|
| Malware_Build_Week_U3 | 17/01/2024 17:48 | Applicazione | 52 KB |
| Malware_Build_Week_U3.i64 | 20/04/2024 20:11 | File i64 | 625 KB |
| msgina32.dll | 20/04/2024 20:15 | Estensione dell'ap... | 7 KB |

Ciò che avviene con l'esecuzione del Malware è la creazione di un nuovo file nominato "**msgina32.dll**". Ciò corrisponde ad un file di libreria creato liberamente dal software malevolo. Si può evincere che tale creazione è avvenuta con una serie di processi nominati, che riporto di seguito:

- **CreateFile**: funzione che crea un file se non esiste con un nome definito, in questo caso "**msgina32.dll**".
- **WriteFile**: scrive o modifica il file appena creato impostando o inserendo valori che sono contenuti nel codice sorgente della PE malevola.
- **CloseFile**: una volta eseguite tutte le modifiche eseguite dalla funzione **WriteFile**, lo salva e chiude con quest'ultima funzione.

Il percorso che fa riferimento alla directory ove è presente la PE malevola. (Le istruzioni citate le riporto di seguito con una immagine della cattura eseguita con ProcMon).

| | | | | | |
|-----------|---------------------------|------|------------|--|---------|
| 0.4068505 | Malware_Build_Week_U3.exe | 1948 | CreateFile | C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll | SUCCESS |
| 0.4073563 | Malware_Build_Week_U3.exe | 1948 | WriteFile | C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll | SUCCESS |
| 0.4074072 | Malware_Build_Week_U3.exe | 1948 | ReadFile | C: | SUCCESS |
| 0.4074444 | Malware_Build_Week_U3.exe | 1948 | WriteFile | C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll | SUCCESS |
| 0.4074649 | Malware_Build_Week_U3.exe | 1948 | CloseFile | C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll | SUCCESS |

Di fatto il programma malevolo a utilizzato le funzioni delle librerie citate in precedenza (**ADVAPI32** ed **KERNEL32**) per per la creazione e modifica delle chiavi di registro esistenti. Con questa analisi, viene confermata l'ipotesi vista in precedenza nei punti precedenti e ci conferma che il codice malevolo tenta di insediarsi in modo **permanente**. Con ciò deduco che ad ogni avvio della macchina, il sistema attiverà il codice malevolo ora presente.

Filtrate includendo solamente l'attività sul registro di Windows.

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

| | | | | |
|------------------|---------------------------|------|---------------|--|
| 20:15:50,4076602 | Malware_Build_Week_U3.exe | 1948 | RegQueryKey | HKLM |
| 20:15:50,4076860 | Malware_Build_Week_U3.exe | 1948 | RegCreateKey | HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon |
| 20:15:50,4077265 | Malware_Build_Week_U3.exe | 1948 | RegSetInfoKey | HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon |
| 20:15:50,4077365 | Malware_Build_Week_U3.exe | 1948 | RegQueryKey | HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon |
| 20:15:50,4077455 | Malware_Build_Week_U3.exe | 1948 | RegSetValue | HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL |

Impostando i filtri di ricerca su **Show Registry Activity** posso verificare che operazioni/azioni sono state eseguite dal codice malevolo durante la sua esecuzione. Nell'immagine sopra, riporto evidenziata la riga con il comando **RegCreateKey**. Con questo processo ed i successivi 3 (RegSetInfoKey, RegQueryKey e RegSetValue) viene creata una nuova chiave di registro con nome GinaDLL.

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

| | | |
|---|-------------|--|
| 952 | RegSetValue | HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL |
| Type: REG_SZ, Length: 520, Data: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll | | |

Il valore della chiave di registro la trovo tramite il comando **RegSetValue** nei Details di Process monitor. Di fatto il valore associato alla chiave creata in precedenza è contenuta del file di libreria presente nella cartella ove è contenuto il malware. Il file .dll è possibile visionarlo con il programma CFF Explorer.

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

| | | | | | |
|-----------|---------------------------|------|------------|--|---------|
| 0.4068505 | Malware_Build_Week_U3.exe | 1948 | CreateFile | C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll | SUCCESS |
| 0.4073563 | Malware_Build_Week_U3.exe | 1948 | WriteFile | C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll | SUCCESS |
| 0.4074072 | Malware_Build_Week_U3.exe | 1948 | ReadFile | C: | SUCCESS |
| 0.4074444 | Malware_Build_Week_U3.exe | 1948 | WriteFile | C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll | SUCCESS |
| 0.4074649 | Malware_Build_Week_U3.exe | 1948 | CloseFile | C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll | SUCCESS |

La chiamata di sistema che ha modificato il contenuto della cartella è **CreateFile** eseguito nel relativo percorso scritto di fianco alla chiamata. Come detto in precedenza, la funzione CreateFile crea un file se non esiste con un nome definito, in questo caso "msgina32.dll".

Le istruzioni successive, scrivono il contenuto della libreria e poi ne salvano il file all'interno della cartella designata.

Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware: Il Malware_Build_Week_U3 è di tipologia persistente una volta che esso viene avviato nella macchina target, in quanto crea una sottochiave di registro nella HKEY della macchina locale, rendendosi così presente ogni qualvolta che il PC viene avviato. Non sfrutta protocolli o servizi di rete e di per se non è un tipo di malware backdoor. Esso importa le librerie dinamicamente sfruttandole quando lo richiede il sorgente. Rientra nei Rootkit e ipotizzo inoltre che esso, dalle operazioni che svolge in esecuzione con ProcMon attivo, esso modifichi varie chiavi di sistema e potrebbe permettere accessi facilitati ad alcune funzioni della macchina da parte di qualsiasi utente che si collega. Inoltre avvia anche molti processi che raccolgono dati sulla macchina (come sistema operativo, versioni, componenti etc..). Potrebbe inoltre anche corrompere file presenti nella memoria rendendoli inaccessibili oppure danneggiarli o infettarli volutamente. Se quest'ultimo avesse anche la libreria di rete importata o caricata nel codice trasmetterebbe tutti i dati di configurazione di sistema, la versione, i processi che elabora e altre varie informazioni che portano un potenziale attaccante ad avere anche pieno accesso alla macchina e alle sue vulnerabilità.