ESERCITAZIONE FINE MODULO 6
BARRECA MONICA

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:

1. Lo scopo della funzione chiamata alla locazione di memoria 00401021
2. Come vengono passati i parametri alla funzione alla locazione 00401021;
3. Che oggetto rappresenta il parametro alla locazione 00401017
4. Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.
5. Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C.
6. 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

1. 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.

2. Quale chiave di registro viene creata?
3. 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.

ESERCITAZIONE FINE MODULO 6
BARRECA MONICA

Esecuzione:

Per prima cosa apriamo IDA Pro e carichiamo il file eseguibile per generare il linguaggio Assembly del nostro malware e poterlo analizzare.

Analisi statica

1. **Quanti parametri sono passati alla funzione Main()?**
2. **Quante variabili sono dichiarate all'interno della funzione Main()?**

Per rispondere alle prime due domande occorre fare una distinzione: IDA Pro facilita molto l'identificazione dei parametri che passano a una funzione e quali variabili sono dichiarate. La differenza sta nell'offset rispetto al EBP (Base Pointer) dello stack: i parametri hanno un offset positivo (vedi rettangolo rosso), mentre le variabili hanno un offset negativo (vedi rettangolo blu). Quindi nel malware oggetto di analisi alla funzione Main() sono passati 3 parametri e 5 variabili.

```
.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
```

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

Entriamo su View, poi su Open Subwiews e infine Segments per vedere quali sezioni sono presenti nel nostro Malware:

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...

- La sezione **.text** è quella che contiene le istruzioni del programma, il vero e proprio codice eseguibile.
- La sezione **.data** è quella che contiene le variabili globali e i dati del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma.
- La sezione **.rdata** è quella che di solito include le informazioni sulle funzioni importate ed esportate e le librerie utilizzate. Questi dati non possono essere modificati.

4. **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

ESERCITAZIONE FINE MODULO 6
BARRECA MONICA

Entrando su **Imports** possiamo vedere le librerie importate:

- **ADVAPI32**: la libreria contenente le funzioni per interagire con servizi e registri del sistema operativo.
- **KERNEL32**: la libreria contenente le funzioni principali per la manipolazione di file e memorie.

In base alle funzioni che vengono richiamate all'interno delle librerie si potrebbe ipotizzare che:

- Come un dropper, il file eseguito inizi la sua esecuzione estraendo il Malware che contiene per salvarlo sul disco (es. SizeofResource, LockResource, LoadResource, FindResourceA)
- Una volta avviato tenti poi di accedere alle chiavi di registro per ottenere la persistenza, facendo in modo che lo stesso SO li avvii automaticamente (es. RegSetValueAX permetterebbe infatti di aggiungere un nuovo valore al registro con la possibilità di settare i rispettivi dati).

Con riferimento al Malware in analisi, spiegare:

5. **Lo scopo della funzione chiamata alla locazione di memoria 00401021**
6. **Come vengono passati i parametri alla funzione alla locazione 00401021**

RegCreateKeyExA permette di creare/modificare una chiave di registro del sistema.
Dalla figura possiamo vedere come prima di chiamare la funzione, con una serie di push vengono caricati sullo stack dei valori diversi corrispondenti ai parametri necessari alla funzione stessa per ricercare/modificare/creare una chiave di registro.

```
.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
```

7. **Che oggetto rappresenta il parametro alla locazione 00401017**

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

Il parametro rappresenta la sottochiave del registro che si vuole creare/aprire. A destra si trova anche il path dove è presente (o verrà creata) la chiave di registro richiesta.

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

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

"test" è un'istruzione condizionale simile all'operatore AND, con la sola differenza che però non modifica il contenuto degli operandi. Quindi il programma effettua questa operazione di controllo fra il registro EAX e sé stesso, non modifica i valori contenuti al loro interno ed in base al risultato setta lo Zero Flag. Quest'ultimo sarà uguale a 1 se e solo se l'AND restituisce 0; mentre sarà uguale a 0 se l'AND restituisce un valore diverso da 0.

"jz" è un tipo di jump condizionale che dipende dal valore dello Zero Flag: salta alla locazione indicata se ZF = 1.

ESERCITAZIONE FINE MODULO 6
BARRECA MONICA

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

Presupponendo che la nostra variabile **a** sia il valore contenuto nel registro EAX, possiamo tradurre il codice assembly in linguaggio C sviluppando il costrutto **IF**:

if (a == 0) {

(esegui le istruzioni alla loc_401032)

} **else** {

(continua con le istruzioni che seguono dopo la loc_401029)

}

10. **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
```

La funzione **RegSetValueExA** permette di aggiungere un nuovo valore all'interno del registro. In questo caso il valore del parametro **ValueName** è "GinaDLL" che corrisponde al nome della nuova chiave di registro aggiunta.

Analisi dinamica

1. **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.

1 Prima esecuzione

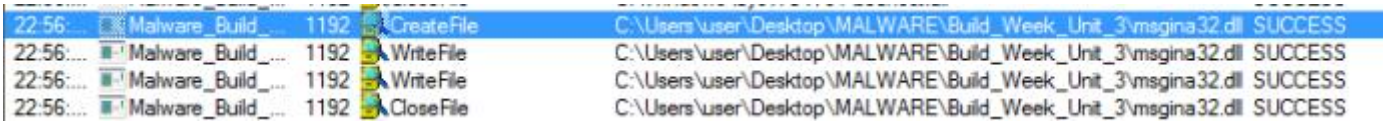
2 Dopo esecuzione

Eseguendo il Malware notiamo la presenza di un nuovo file che prima non c'era: "**msgina32.dll**". Un file di libreria creato liberamente dal software malevolo. Si può ipotizzare quindi che il Malware abbia creato questo nuovo file attraverso una serie di operazioni come:

- **CreateFile:** crea un nuovo file con un nome definito, in questo caso "msgina32.dll".

ESERCITAZIONE FINE MODULO 6
BARRECA MONICA

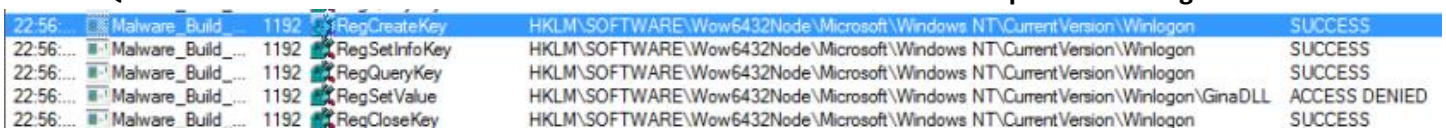
- **WriteFile:** scrive o modifica il file appena creato impostando o inserendo valori contenuti nel codice sorgente del Malware.
- **CloseFile**: una volta eseguite tutte le modifiche il nuovo file viene salvato e chiuso.



Con l'analisi attraverso ProcMon possiamo confermare l'ipotesi dei punti precedenti per cui il malware tenta di insediarsi in modo **permanente** nel sistema sfruttando le librerie che importa.

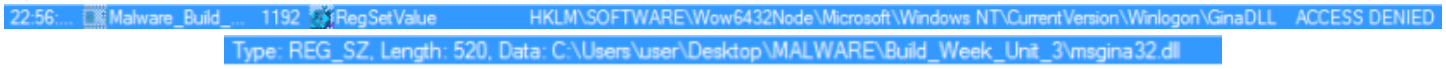
Filtrate includendo solamente l'attività sul registro di Windows.

2. **Quale chiave di registro viene creata?**
3. **Quale valore viene associato alla chiave di registro creata?**
4. **Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware?**



Impostando il filtro su **Show Registry Activity** come richiesto possiamo verificare quali operazioni/azioni sono state eseguite sul registro.

Con i processi **RegCreateKey, RegSetInfoKey, RegQueryKey e RegSetValue** viene creata una nuova chiave di registro con il nome **"GinaDLL".**



Nei Details di ProcMon vediamo come il valore associato tramite il comando **RegSetValue** è contenuto all'interno del file di libreria creato non appena avviato il file eseguibile. Per poterlo vedere bisogna utilizzare il tool CFF Explorer.

La chiamata di sistema che ha modificato il contenuto della cartella è **CreateFile** che come detto in precedenza, ha creato, scritto e salvato il file "msgina32.dll" nel path indicato.

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

Malware_Build_Week_U3.exe è sicuramente un programma malevolo di tipo **dropper**: utilizza prima il set di API's di Windows per estrarre il malware contenuto nella sezione risorse e lo salva su disco per utilizzi futuri. Tramite le librerie che il malware importa una volta avviato possiamo capire che cerca di interagire con i servizi e i registri al fine di rendersi persistente sicuramente, ma non solo. Avendo accesso alle chiavi di registro e potendole modificare, potrebbe garantire accesso facilitato ad alcune funzioni del sistema a chiunque abbia accesso alla macchina. Inoltre, importa anche quelle funzioni che gli permettono di manipolare file e memorie; quindi, potrebbe potenzialmente rendere i dati inaccessibili, danneggiati o infetti. Avvia diversi processi che raccolgono dati sulla macchina infetta. Si potrebbe classificare come un rootkit.