

ANALISI COMPLETA DI UN MALWARE REALE

BUILD WEEK 3

Capogruppo:

Rovella Andrea

Collaboratori:

Castoldi Dario, Chieppa Marco, Ciulla Marco, Curatolo Samuele,
Derosas Antonio, Francese Bruno, Frau Salvatore, Poser Paola

Analisi completa di un Malware reale con nome “Malware_Build_Week_U3”, ovvero di un file eseguibile presente nella nostra macchina virtuale dedicata all’analisi dei Malware. Per poter studiare la natura e il comportamento di questo software malevolo, si andranno ad effettuare:

- I. Analisi statica del Malware;
 - II. Analisi dinamica del Malware.
-

I. ANALISI STATICA DEL MALWARE

Questo tipo di analisi permette di dedurre il comportamento del Malware senza la necessità di eseguirlo, ma utilizzando diversi tool che permettono di esaminare l’eseguibile e dedurre le funzionalità.

Con riferimento al suo file eseguibile, quindi, si analizzano i seguenti punti:

1. Analisi della funzione **Main()**, delle **sezioni** e delle **librerie** del Malware;
 2. **Analisi di basso livello** di alcune istruzioni in linguaggio **Assembly** con traduzione in **costrutto C** delle istruzioni tra gli indirizzi **00401027** e **00401029**;
 3. Riprendendo l’analisi del codice tra le locazioni di memoria **00401080** e **00401128**, effettuare ulteriori analisi e un **diagramma di flusso** delle tre funzioni viste finora.
-

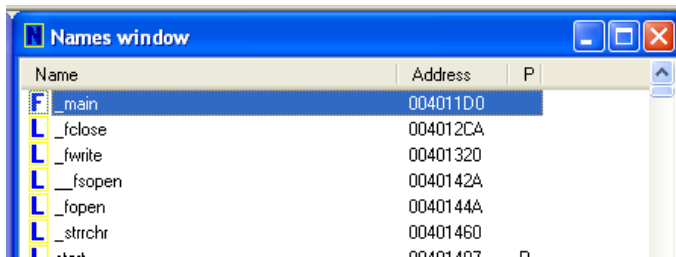
1. Analisi della funzione **Main()**, delle sezioni e delle librerie del Malware.

In particolare, si andranno a vedere i seguenti punti:

- a) Parametri passati alla funzione **Main()**;
- b) Variabili dichiarate all’interno della funzione **Main()**;
- c) **Sezioni** presenti all’interno del file eseguibile e descrizione di queste;
- d) **Librerie** importate dal Malware. Per ognuna delle librerie importate, fare delle ipotesi sulla base dell’analisi statica delle funzionalità che il Malware potrebbe implementare;

Di seguito l'analisi di questi componenti:

a) Parametri passati alla funzione **Main()**. Data la seguente funzione:



| Name | Address | P |
|---------|----------|-----|
| main | 004011D0 | |
| _fclose | 004012CA | |
| _fwrite | 00401320 | |
| _fsopen | 0040142A | |
| _fopen | 0040144A | |
| _strchr | 00401460 | |
| ... | ... | ... |

Ovvero in codice Assembly:

```
.text:004011D0
.text:004011D0 ; :!!!!!!!!!!!!!! SUBROUTINE !!!!!!!!!!!!!!!
.text:004011D0
.text:004011D0 ; Attributes: bp-based frame
.text:004011D0
.text:004011D0 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:004011D0 _main proc near ; CODE XREF: start+AF4p
.text:004011D0
.text:004011D0 hModule = dword ptr -11Ch
.text:004011D0 Data = byte ptr -118h
.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
```

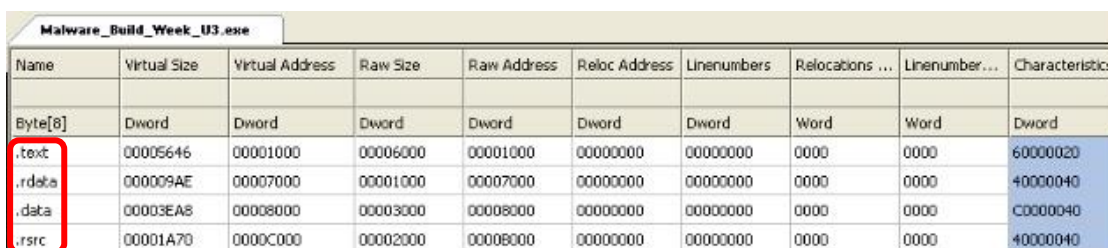
I suoi parametri sono i seguenti:

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

b) Le variabili dichiarate all'interno della funzione **Main()** sono le seguenti:

```
.text:004011D0 hModule = dword ptr -11Ch
.text:004011D0 Data = byte ptr -118h
.text:004011D0 var_8 = dword ptr -8
.text:004011D0 var_4 = dword ptr -4
```

c) Sezioni di cui si compone il **Malware**, informazioni che si possono ottenere il tool CFF Explorer (riquadro in rosso):



| Name | Virtual Size | Virtual Address | Raw Size | Raw Address | Reloc Address | Linenumbers | Relocations ... | Linenumber... | Characteristics |
|---------|--------------|-----------------|----------|-------------|---------------|-------------|-----------------|---------------|-----------------|
| Byte[8] | Dword | Dword | Dword | Dword | Dword | Dword | Word | Word | Dword |
| .text | 00005646 | 00001000 | 00006000 | 00001000 | 00000000 | 00000000 | 0000 | 0000 | 60000020 |
| .rdata | 000009AE | 00007000 | 00001000 | 00007000 | 00000000 | 00000000 | 0000 | 0000 | 40000040 |
| .data | 00003EAB | 00008000 | 00003000 | 00008000 | 00000000 | 00000000 | 0000 | 0000 | C0000040 |
| .rsrc | 00001A70 | 0000C000 | 00002000 | 0000B000 | 00000000 | 00000000 | 0000 | 0000 | 40000040 |

Dove le sezioni del Malware analizzato sono le seguenti:

- **.text**: contiene le righe di codice che la CPU esegue una volta che il software viene avviato;
- **.rdata**: include informazioni su librerie e funzioni importate ed esportate dal PE;
- **.data**: In questa sezione troviamo di solito le variabili e i dati del programma che devono essere disponibili da qualsiasi parte e queste variabili si dicono globali quando non è definita all'interno di una funzione ed è GLOBALMENTE DICHIARATA (quindi accessibile da qualsiasi funzione eseguibile).
- **.rsrc**: include le risorse utilizzate dall'eseguibile come immagini, icone, stringhe e menù che non sono parte dell'eseguibile stesso.

d) Librerie importate dal Malware, informazioni ottenibili con lo stesso tool, CFF Explorer (nel riquadro in rosso):

| Malware_Build_Week_U3.exe | | | | | | |
|---------------------------|--------------|----------|---------------|----------------|----------|-----------|
| Module Name | Imports | OFTs | TimeDateStamp | ForwarderChain | Name RVA | FTs (IAT) |
| | | | | | | |
| szAnsi | (nFunctions) | Dword | Dword | Dword | Dword | Dword |
| KERNEL32.dll | 51 | 00007534 | 00000000 | 00000000 | 0000769E | 0000700C |
| ADVAPI32.dll | 2 | 00007528 | 00000000 | 00000000 | 000076D0 | 00007000 |

Il malware importa due librerie, KERNEL32.dll e ADVAPI32.dll.

KERNEL32.dll è una libreria che contiene le funzioni principali per comunicare con il sistema operativo, quali la gestione della memoria e la modifica dei file. In questo caso contiene al suo interno le seguenti funzioni:

| Malware_Build_Week_U3.exe | | | | | | |
|---------------------------|--------------|----------|---------------|----------------|----------|-----------|
| Module Name | Imports | OFTs | TimeDateStamp | ForwarderChain | Name RVA | FTs (IAT) |
| 0000769E | N/A | 000074EC | 000074F0 | 000074F4 | 000074F8 | 000074FC |
| szAnsi | (nFunctions) | Dword | Dword | Dword | Dword | Dword |
| KERNEL32.dll | 51 | 00007534 | 00000000 | 00000000 | 0000769E | 0000700C |
| ADVAPI32.dll | 2 | 00007528 | 00000000 | 00000000 | 000076D0 | 00007000 |

| OFTs | FTs (IAT) | Hint | Name | | | | |
|----------|-----------|------|--------------------------|----------|----------|------|-------------------------|
| Dword | Dword | Word | szAnsi | | | | |
| 00007632 | 00007632 | 0295 | SizeofResource | 00007816 | 00007816 | 0115 | GetFileType |
| 00007644 | 00007644 | 01D5 | LockResource | 00007824 | 00007824 | 0150 | GetStartupInfoA |
| 00007654 | 00007654 | 01C7 | LoadResource | 00007836 | 00007836 | 0109 | GetEnvironmentVariableA |
| 00007622 | 00007622 | 02BB | VirtualAlloc | 00007850 | 00007850 | 0175 | GetVersionExA |
| 00007674 | 00007674 | 0124 | GetModuleFileNameA | 00007860 | 00007860 | 019D | HeapDestroy |
| 0000768A | 0000768A | 0126 | GetModuleHandleA | 0000786E | 0000786E | 019B | HeapCreate |
| 00007612 | 00007612 | 00B6 | FreeResource | 0000787C | 0000787C | 02BF | VirtualFree |
| 00007664 | 00007664 | 00A3 | FindResourceA | 0000788A | 0000788A | 022F | RtlUnwind |
| 00007604 | 00007604 | 001B | CloseHandle | 00007896 | 00007896 | 0199 | HeapAlloc |
| 000076DE | 000076DE | 00CA | GetCommandLineA | 000078A2 | 000078A2 | 01A2 | HeapReAlloc |
| 000076F0 | 000076F0 | 0174 | GetVersion | 000078B0 | 000078B0 | 027C | SetStdHandle |
| 000076FE | 000076FE | 007D | ExitProcess | 000078C0 | 000078C0 | 00AA | FlushFileBuffers |
| 0000770C | 0000770C | 019F | HeapFree | 000078D4 | 000078D4 | 026A | SetFilePointer |
| 00007718 | 00007718 | 011A | GetLastError | 000078E6 | 000078E6 | 0034 | CreateFileA |
| 00007728 | 00007728 | 02DF | WriteFile | 000078F4 | 000078F4 | 00BF | GetCPInfo |
| 00007734 | 00007734 | 029E | TerminateProcess | 00007900 | 00007900 | 00B9 | GetACP |
| 00007748 | 00007748 | 00F7 | GetCurrentProcess | 0000790A | 0000790A | 0131 | GetOEMCP |
| 0000775C | 0000775C | 02AD | UnhandledExceptionFilter | 00007916 | 00007916 | 013E | GetProcAddress |
| 00007778 | 00007778 | 00B2 | FreeEnvironmentStringsA | 00007928 | 00007928 | 01C2 | LoadLibraryA |
| 00007792 | 00007792 | 00B3 | FreeEnvironmentStringsW | 00007938 | 00007938 | 0261 | SetEndOfFile |
| 000077AC | 000077AC | 02D2 | WideCharToMultiByte | 00007948 | 00007948 | 0218 | ReadFile |
| 000077C2 | 000077C2 | 0106 | GetEnvironmentStrings | 00007954 | 00007954 | 01E4 | MultiByteToWideChar |
| 000077DA | 000077DA | 0108 | GetEnvironmentStringsW | 0000796A | 0000796A | 01BF | LCMapStringA |
| 000077F4 | 000077F4 | 026D | SetHandleCount | 0000797A | 0000797A | 01C0 | LCMapStringW |
| 00007806 | 00007806 | 0152 | GetStdHandle | 0000798A | 0000798A | 0153 | GetStringTypeA |
| | | | | 0000799C | 0000799C | 0156 | GetStringTypeW |

Il Malware sembra essere un **dropper**, questo è deducibile in quanto utilizza funzioni tipo: **FindResourceA**, per trovare il Malware/contenuto malevolo contenuto nella sezione risorse (.rsrc) utilizzando come parametri il nome e il tipo della risorsa; **LoadResource**, recupera un handle che può essere usato per ottenere un puntatore al primo byte della risorsa specificata in memoria; **LockResource**, per recuperare un puntatore alla risorsa specificata; **SizeofResource**, per identificare la dimensione della risorsa; **FreeResource**, una funzione obsoleta, utilizzata sui sistemi Windows a 16 bit per liberare la memoria occupata dalla risorsa.

Altre funzioni come **CreateFileA/WriteFile** ci fanno capire dei tentativi di apertura/creazione di file e scrittura su altri file.

La libreria **ADVAPI32.dll** contiene, invece, tutte le funzioni che permettono di interagire con i registri e i servizi del sistema operativo. Contiene due funzioni al suo interno:

| Malware_Build_Week_U3.exe | | | | | | |
|---------------------------|--------------|----------|-----------------|----------------|----------|-----------|
| Module Name | Imports | OFTs | TimeStamp | ForwarderChain | Name RVA | FTs (IAT) |
| 000076D0 | N/A | 00007500 | 00007504 | 00007508 | 0000750C | 00007510 |
| szAnsi | (nFunctions) | Dword | Dword | Dword | Dword | Dword |
| KERNEL32.dll | 51 | 00007534 | 00000000 | 00000000 | 0000769E | 0000700C |
| ADVAPI32.dll | 2 | 00007528 | 00000000 | 00000000 | 000076D0 | 00007000 |
| OFTs | FTs (IAT) | Hint | Name | | | |
| Dword | Dword | Word | szAnsi | | | |
| 000076AC | 000076AC | 0186 | RegSetValueExA | | | |
| 000076BE | 000076BE | 015F | RegCreateKeyExA | | | |

Il Malware richiama solo due funzioni tramite questa libreria: **RegCreateKeyExA**, per creare una chiave di registro specificata o aprirla se è già esistente, e **RegSetValueExA**, per impostarne i valori della chiave.

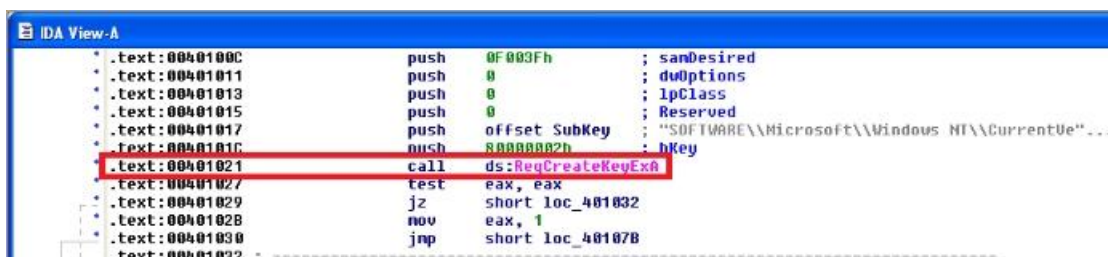
2. Analisi di basso livello di alcune istruzioni in linguaggio Assembly con traduzione in costrutto C delle istruzioni tra gli indirizzi 00401027 e 00401029.

Si analizza:

- 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»?;
- Nel complesso delle due funzionalità appena viste, spiegate quale funzionalità sta implementando il **Malware** in questa sezione.

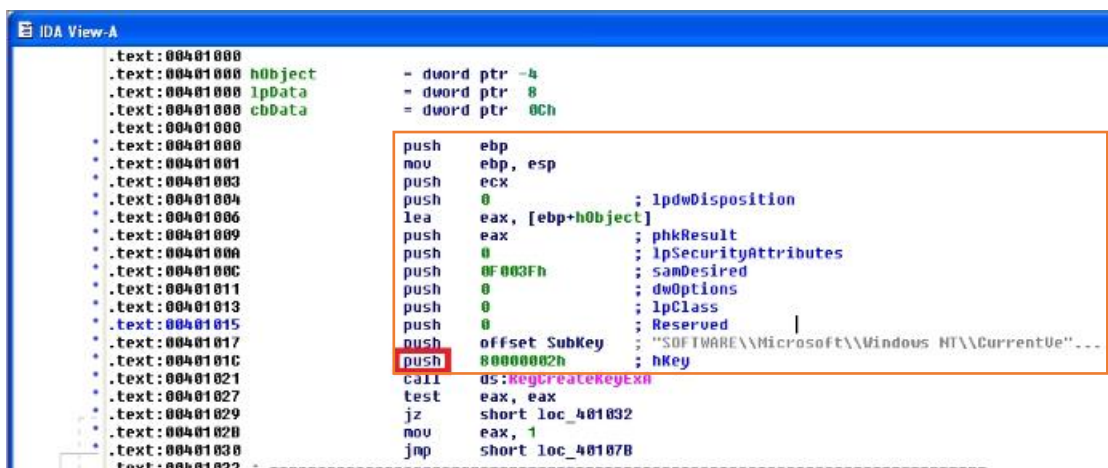
Di seguito l'analisi delle diverse istruzioni:

a) Alla locazione di memoria **00401021**, possiamo notare la chiamata di funzione **RegCreateKeyExA**, che sta a significare che il Malware sta tentando di creare o aprire una chiave di registro:



```
.text:0040100C      push    0F003Fh      ; sanDesired
.text:00401011      push    0            ; dwOptions
.text:00401013      push    0            ; lpClass
.text:00401015      push    0            ; Reserved
.text:00401017      push    offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentUe"...
.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      ;
```

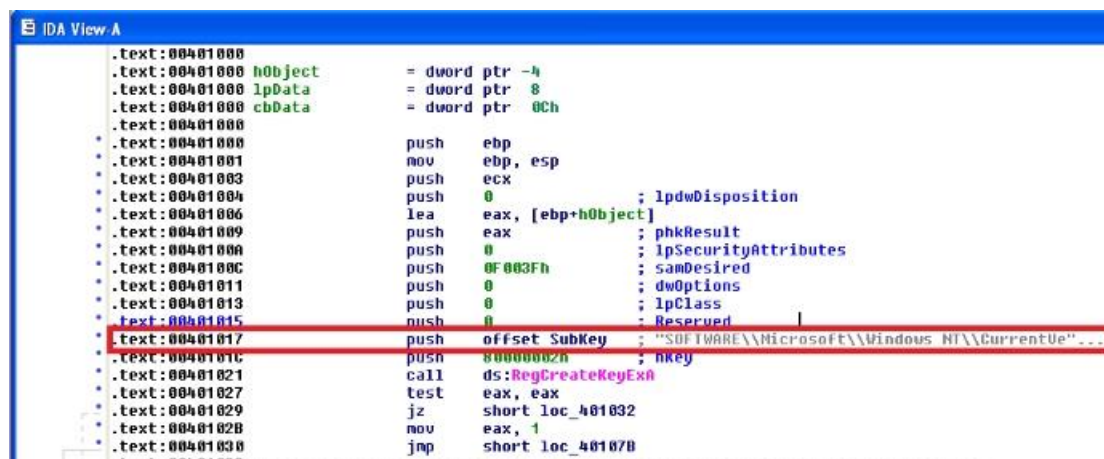
b) I parametri della **funzione** alla locazione **00401021** vengono passati tramite l'istruzione **push**. Il suo compito è appunto spingere (inserire) un valore sullo stack della funzione chiamata:



```
.text:00401000      hObject      = dword ptr -4
.text:00401000      lpData       = dword ptr 8
.text:00401000      cbData      = dword ptr 0Ch
.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       ; sanDesired
.text:00401011      push    0            ; dwOptions
.text:00401013      push    0            ; lpClass
.text:00401015      push    0            ; Reserved
.text:00401017      push    offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentUe"...
.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      ;
```

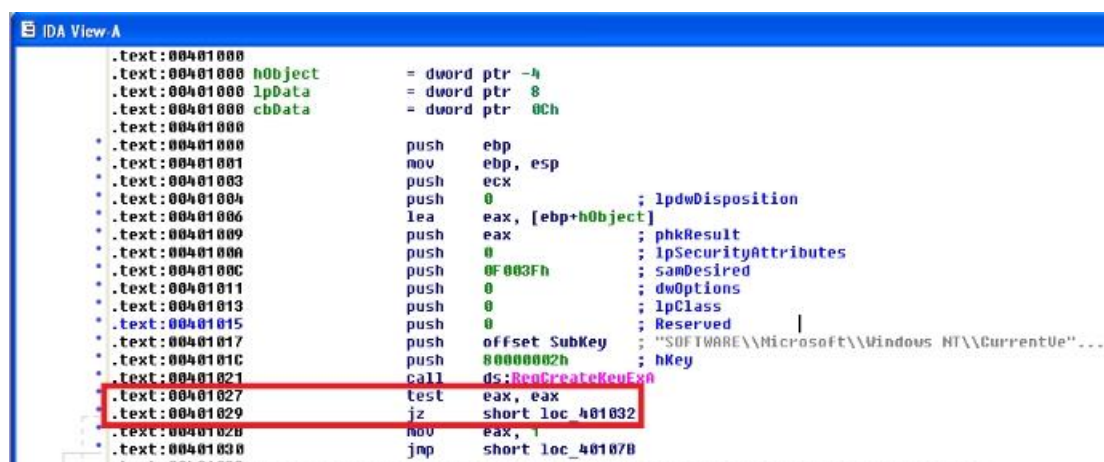

c) Il parametro alla locazione **00401017** rappresenta il nome della **sottochiave** che questa funzione apre o crea. Come si può vedere dallo screenshot seguente il parametro passato è la seguente directory:

“SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon”



```
.text:00401000
.text:00401000 hObject      = dword ptr -4
.text:00401000 lpData       = dword ptr 8
.text:00401000 cbData       = dword ptr 0Ch
.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\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
.text:00401030      jmp      short loc_401070
.text:00401032
```

d) La funzione **RegCreateKeyExA()** restituisce come risultato il valore “0” nel caso l’operazione sia stata completata con successo, questo valore viene inserito nel registro **eax**. Le istruzioni **test** e **jz**, comprese tra gli indirizzi **00401027** e **00401029**, controllano se il registro **eax** sia uguale a zero utilizzando l'istruzione "**test eax, eax**" che si comporta come l'istruzione **AND**, con la differenza che l'istruzione test non modifica il valore di alcun operando. Poi imposta il flag **ZF** del registro **FLAGS** (Se il risultato è 0, ZF è impostato su 1, altrimenti viene impostato su 0) mentre il risultato dell'AND viene scartato. Successivamente, viene eseguito un salto condizionale ("**jz**" che significa "**jump if zero**") alla locazione **00401032** se il flag ZF = 1. In sintesi, queste istruzioni controllano se il registro **eax** contiene il valore zero e, in caso positivo, eseguono un salto a un'altra locazione del codice (**loc_401032**), altrimenti continua con il normale flusso del programma:



```
.text:00401000
.text:00401000 hObject      = dword ptr -4
.text:00401000 lpData       = dword ptr 8
.text:00401000 cbData       = dword ptr 0Ch
.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\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
.text:00401030      jmp      short loc_401070
.text:00401032
```

e) Con riferimento al quesito precedente, riportiamo la rappresentazione del codice **Assembly** nel corrispondente costruito **C**:

```
if (eax == 0)
    { funct_401032(); }
else
    {
        eax = 1;
        funct_40107B();
    }
```

f) Valutando la chiamata alla locazione **00401047**, il valore del parametro «**ValueName**» che troviamo è "**GinaDLL**", una libreria dinamica **Win32** che opera nel contesto del processo **Winlogon** e che, pertanto, viene caricata molto presto nel processo di avvio. Winlogon è un componente Windows responsabile della gestione della SAS (Secure Attention Sequence), caricando il profilo utente al momento del login, e possibilmente bloccando il PC durante l'esecuzione dello screensaver. Invece, lo scopo di una DLL GINA è fornire procedure personalizzabili di identificazione e autenticazione dell'utente.

| | | | |
|-----------------|------|--------------------|-------------|
| *.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 | |

g) Apparentemente, il Malware apre la chiave di registro:

"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion"

ed aggiunge parametri e/o modifica, in qualche modo, la libreria "GinaDLL" richiamata all'indirizzo di memoria 440103E.

3. Riprendendo l'analisi del codice tra le locazioni di memoria 00401080 e 00401128:

- Determinare il valore del parametro "**ResourceName**" passato alla funzione **FindResourceA()**;
- Determinare le **funzionalità** del Malware in questa sezione di codice;
- Se è possibile identificare questa funzionalità utilizzando l'**analisi statica basica**, elencando le **evidenze a supporto**;
- Disegnare un **diagramma di flusso** delle tre funzioni principali viste fin'ora.

Di seguito le diverse analisi:

a) Il parametro che viene passato alla funzione **FindResourceA** come “ResourceName” è **TGAD**:

```
.text:00401088 loc_401088:      ; CODE XREF: sub_401080+2F7j
* .text:00401088      nov     eax, lpType      ; lpType
* .text:0040108D      push    eax
* .text:0040108E      nov     ecx, lpName     ; lpName
* .text:004010C4      push    ecx
* .text:004010C5      nov     edx, [ebp+hModule] ; hModule
* .text:004010C8      push    edx
* .text:004010C9      call    ds:FindResourceA ; lpName, dd offset aTgad ; DATA XREF: sub_401080+3E7j
* .text:004010CF      nov     [ebp+hResInfo], eax ; "TGAD"
* .text:004010D2      cmp     [ebp+hResInfo], 0
* .text:004010D6      jnz     short loc_4010DF
* .text:004010D8      xor     eax, eax
* .text:004010DA      jmp     loc_4011BF
```

Ecco il parametro TGAD all'interno della sezione .data

```
.data:00408030      ; "BINARY"
* .data:00408034      ; LPCSTR lpName
* .data:00408034      lpName      dd offset aTgad ; DATA XREF: sub_401080+3E7j
* .data:00408034      ; "TGAD"
* .data:00408038      aTgad      db 'TGAD',0 ; DATA XREF: .data:lpNamefo
* .data:0040803D      align 10h
```

b) Dall'indirizzo **00401080** all'indirizzo **00401128** il Malware chiama in ordine le funzioni **FindResourceA**, **LoadResource**, **LockResource** e **SizeofResource**. Queste chiamate ci fanno capire che la funzionalità che sta implementando il Malware è il caricamento di codice malevolo contenuto all'interno della sezione risorse (.rsrc) dell'eseguibile: il Malware è un **dropper**.

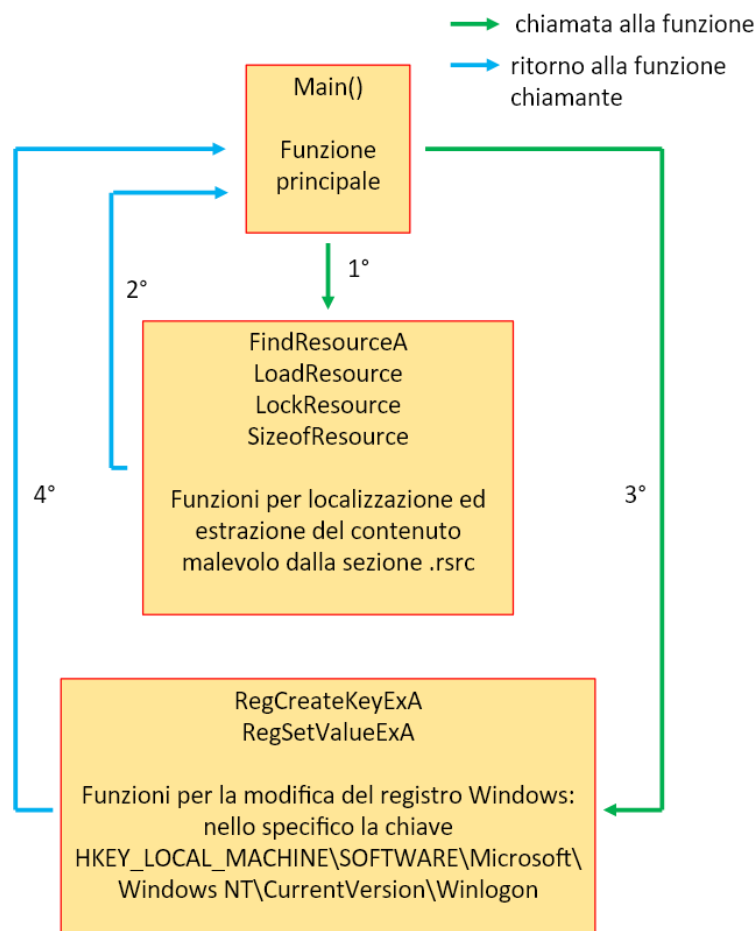
```
* .text:004010C9      call    ds:FindResourceA
* .text:004010CF      nov     [ebp+hResInfo], eax
* .text:004010D2      cmp     [ebp+hResInfo], 0
* .text:004010D6      jnz     short loc_4010DF
* .text:004010D8      xor     eax, eax
* .text:004010DA      jmp     loc_4011BF
* .text:004010DF      ;
* .text:004010DF      loc_4010DF:      ; CODE XREF: sub_401080+567j
* .text:004010DF      nov     eax, [ebp+hResInfo]
* .text:004010E2      push    eax ; hResInfo
* .text:004010E3      nov     ecx, [ebp+hModule]
* .text:004010E6      push    ecx ; hModule
* .text:004010E7      call    ds:LoadResource
* .text:004010ED      nov     [ebp+hResData], eax
* .text:004010F0      cmp     [ebp+hResData], 0
* .text:004010F4      jnz     short loc_4010FB
* .text:004010F6      jmp     loc_4011A5
* .text:004010FB      ;
* .text:004010FB      loc_4010FB:      ; CODE XREF: sub_401080+747j
* .text:004010FB      nov     edx, [ebp+hResData]
* .text:004010FE      push    edx ; hResData
* .text:004010FF      call    ds:LockResource
* .text:00401105      nov     [ebp+var_8], eax
* .text:00401108      cmp     [ebp+var_8], 0
* .text:0040110C      jnz     short loc_401113
* .text:0040110E      jmp     loc_4011A5
* .text:00401113      ;
* .text:00401113      loc_401113:      ; CODE XREF: sub_401080+8C7j
* .text:00401113      nov     eax, [ebp+hResInfo]
* .text:00401116      push    eax ; hResInfo
* .text:00401117      nov     ecx, [ebp+hModule]
* .text:0040111A      push    ecx ; hModule
* .text:0040111B      call    ds:SizeofResource
```

c) Con l'utilizzo dell'**analisi statica basica** è possibile ipotizzare questa funzionalità: andando ad analizzare l'eseguibile con CFF Explorer possiamo vedere che oltre alla libreria importata dal Malware, **KERNEL32.dll**, si possono anche vedere le funzioni che vengono importate da quella

libreria. Tra queste si possono trovare **FindResourceA**, **LoadResource**, **LockResource** e **SizeofResource**.

| Malware_Build_Week_U3.exe | | | | | | |
|---------------------------|--------------|----------|--------------------|----------------|----------|-----------|
| Module Name | Imports | OFTs | TimeDateStamp | ForwarderChain | Name RVA | FTs (IAT) |
| 0000769E | N/A | 000074EC | 000074F0 | 000074F4 | 000074F8 | 000074FC |
| szAnsi | (nFunctions) | Dword | Dword | Dword | Dword | Dword |
| KERNEL32.dll | 51 | 00007534 | 00000000 | 00000000 | 0000769E | 0000700C |
| ADVAPI32.dll | 2 | 00007528 | 00000000 | 00000000 | 000076D0 | 00007000 |
| OFTs | FTs (IAT) | Hint | Name | | | |
| 000075A0 | 00007078 | 00007836 | 00007838 | | | |
| Dword | Dword | Word | szAnsi | | | |
| 00007632 | 00007632 | 0295 | SizeofResource | | | |
| 00007644 | 00007644 | 01D5 | LockResource | | | |
| 00007654 | 00007654 | 01C7 | LoadResource | | | |
| 00007622 | 00007622 | 02BB | VirtualAlloc | | | |
| 00007674 | 00007674 | 0124 | GetModuleFileNameA | | | |
| 0000768A | 0000768A | 0126 | GetModuleHandleA | | | |
| 00007612 | 00007612 | 00B6 | FreeResource | | | |
| 00007664 | 00007664 | 00A3 | FindResourceA | | | |

d) Diagramma di flusso delle 3 funzioni principali analizzate fino ad ora:



II. ANALISI DINAMICA

L'analisi dinamica di malware è il processo di esecuzione del software malevolo all'interno di un ambiente controllato per rilevare le sue azioni e comportamenti. Si utilizzano diverse tecniche, come l'iniezione di codice, l'utilizzo di debugger, i sandboxing e gli strumenti di monitoraggio del traffico di rete, per eseguire una analisi dinamica. L'obiettivo principale è quello di determinare le funzionalità del malware, inclusi i suoi metodi di propagazione e le sue capacità di danneggiare il sistema infetto. Questa analisi è fondamentale per identificare le minacce attuali e sviluppare soluzioni efficaci di sicurezza informatica.

Con riferimento al file eseguibile del Malware in questione, si analizzano i seguenti punti:

1. **Analisi dinamica** del Malware e successiva unione delle informazioni raccolte finora da analisi statica e dinamica del Malware stesso;
2. Analizzando il componente **Gina DLL** e le azioni che un eventuale attaccante può fare su questo file, si delinea il profilo del Malware e si crea un grafico che ne rappresenti lo scopo.

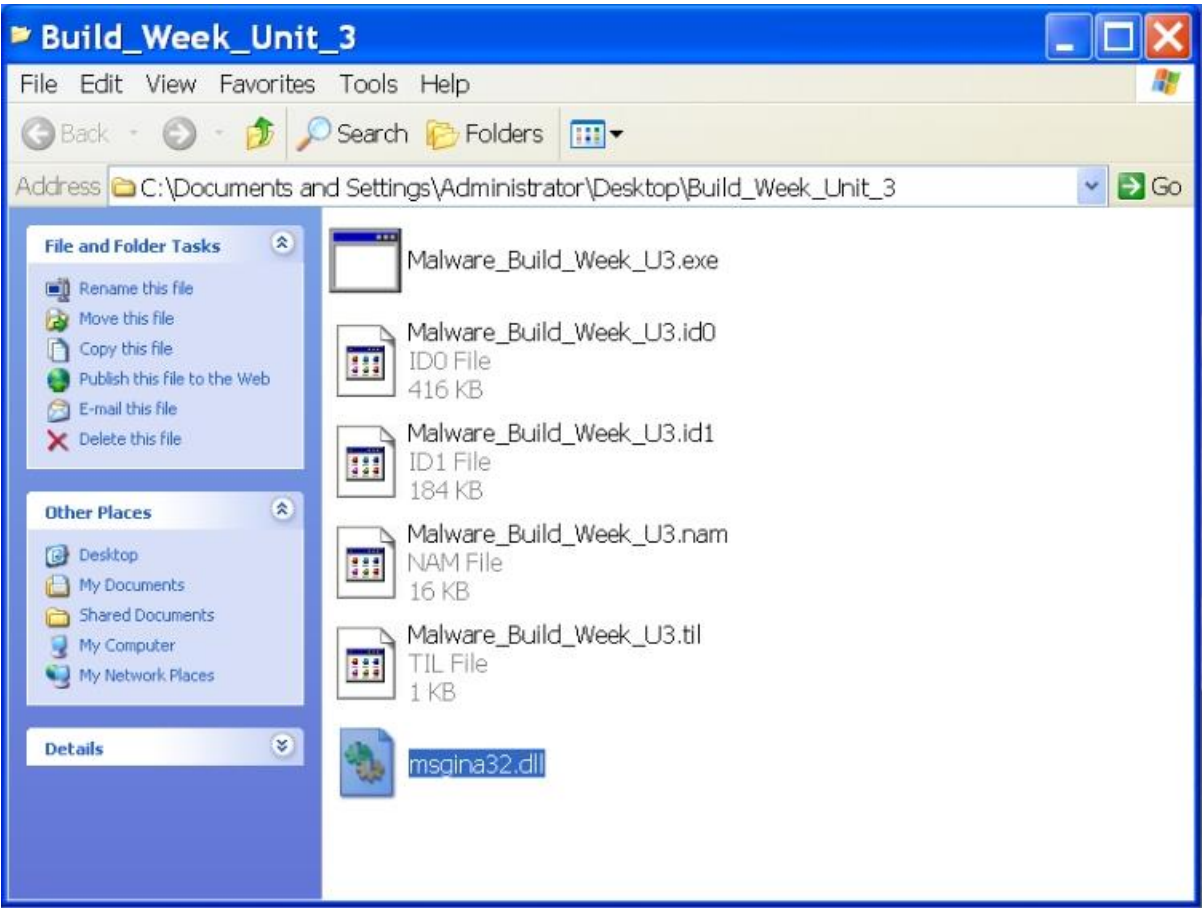
1. Analisi dinamica del Malware e successiva unione delle informazioni raccolte finora da analisi statica e dinamica.

Per effettuare l'analisi dinamica del Malware (principalmente con ProcessMonitor) lo si **esegue**, innanzitutto, **in ambiente sicuro**. Dopo averlo avviato:

- a) evidenziare i **cambiamenti** all'interno della cartella in cui si trova il Malware;
- b) Analizzando le attività sul **registro Windows** identificare quale **chiave di registro** viene creata;
- c) Analizzando le attività sul **registro Windows** identificare il **valore associato** alla chiave di registro creata;
- d) Analizzando le attività sul **file system** identificare quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware;
- e) **Unire tutte le informazioni** raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware.

Di seguito le diverse analisi:

a) Dopo aver avviato l’eseguitibile, si può notare come venga creato un **file** all’interno della cartella contenente il Malware:



b) Analizzando con **ProcessMonitor** le attività sul registro di Windows possiamo vedere la creazione della chiave “**HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon**”:

| | | | | |
|-----|------|----------------|---|----------------|
| ... | 1960 | RegOpenKey | HKLM\System\CurrentControlSet\Control\Terminal Server | SUCCESS |
| ... | 1960 | RegQuery\Value | HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat | SUCCESS |
| ... | 1960 | RegCloseKey | HKLM\System\CurrentControlSet\Control\Terminal Server | SUCCESS |
| ... | 1960 | RegOpenKey | HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Secur32.dll | NAME NOT FOUND |
| ... | 1960 | RegOpenKey | HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\RPCRT4.dll | NAME NOT FOUND |
| ... | 1960 | RegOpenKey | HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ADVAPI32.dll | NAME NOT FOUND |
| ... | 1960 | RegOpenKey | HKLM\System\CurrentControlSet\Control\Terminal Server | SUCCESS |
| ... | 1960 | RegQuery\Value | HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat | SUCCESS |
| ... | 1960 | RegQuery\Value | HKLM\System\CurrentControlSet\Control\Terminal Server\TSUserEnabled | SUCCESS |
| ... | 1960 | RegCloseKey | HKLM\System\CurrentControlSet\Control\Terminal Server | SUCCESS |
| ... | 1960 | RegOpenKey | HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon | SUCCESS |
| ... | 1960 | RegQuery\Value | HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\LeakTrack | NAME NOT FOUND |
| ... | 1960 | RegCloseKey | HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon | SUCCESS |
| ... | 1960 | RegOpenKey | HKLM | SUCCESS |
| ... | 1960 | RegOpenKey | HKLM\Software\Microsoft\Windows NT\CurrentVersion\Diagnostics | NAME NOT FOUND |
| ... | 1960 | RegOpenKey | HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ntdll.dll | NAME NOT FOUND |
| ... | 1960 | RegOpenKey | HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\kernel32.dll | NAME NOT FOUND |
| ... | 1960 | RegCreateKey | HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon | SUCCESS |
| ... | 1960 | RegSetValue | HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL | SUCCESS |
| ... | 1960 | RegCloseKey | HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon | SUCCESS |

c) Il valore che viene associato a questa chiave è il **path** della libreria “corrotta” creata precedentemente dal Malware nella propria cartella:

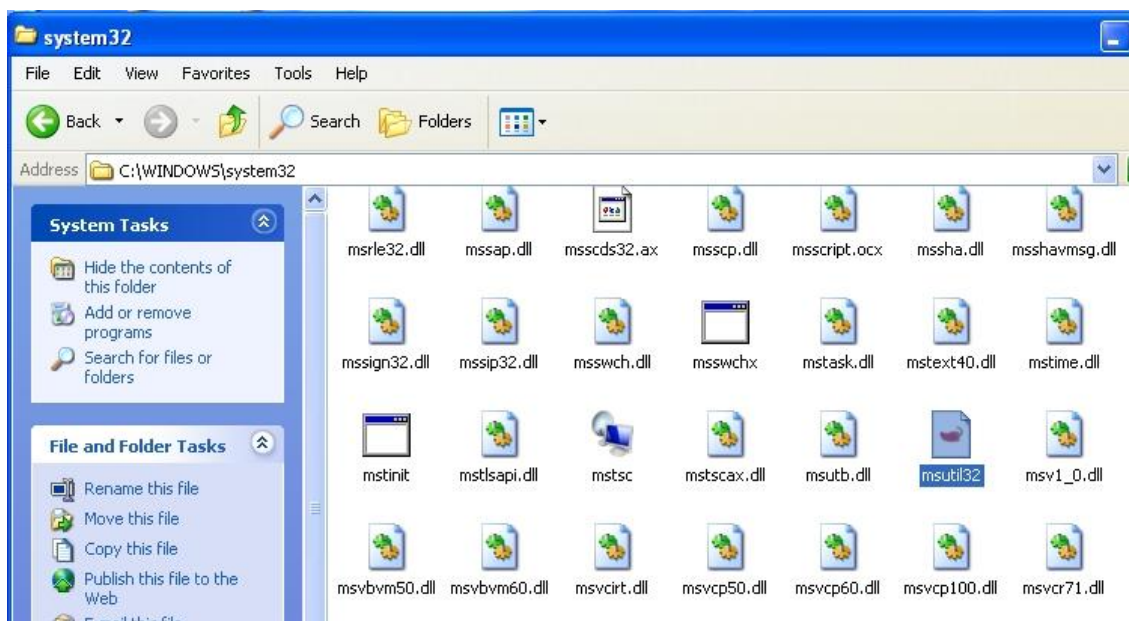
| | | |
|------|--------------|---|
| 1960 | RegOpenKey | HKLM |
| 1960 | RegOpenKey | HKLM\Software\Microsoft\Windows NT\CurrentVersion\Diagnostics |
| 1960 | RegOpenKey | HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\nt |
| 1960 | RegOpenKey | HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ke |
| 1960 | RegCreateKey | HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon |
| 1960 | RegSetValue | HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL |
| 1960 | RegCloseKey | HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon |

| | |
|----------------|---|
| SUCCESS | Desired Access: Maximum Allowed |
| SUCCESS | Desired Access: Read |
| NAME NOT FOUND | Desired Access: Read |
| NAME NOT FOUND | Desired Access: Read |
| NAME NOT FOUND | Desired Access: Read |
| SUCCESS | Desired Access: All Access |
| SUCCESS | Type: REG_SZ, Length: 520, Data: C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll |
| SUCCESS | |

d) Passando all’analisi delle attività sul **File System** troviamo la chiamata di funzione che modifica il contenuto della cartella contenente il Malware:

| | | | |
|------|-------------------|---|----------------|
| 1960 | CreateFile | C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3 | SUCCESS |
| 1960 | FileSystemControl | C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3 | SUCCESS |
| 1960 | QueryOpen | C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe.Local | NAME NOT FOUND |
| 1960 | CreateFile | C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll | SUCCESS |
| 1960 | CreateFile | C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3 | SUCCESS |
| 1960 | CloseFile | C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3 | SUCCESS |
| 1960 | WriteFile | C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll | SUCCESS |
| 1960 | WriteFile | C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll | SUCCESS |
| 1960 | CloseFile | C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll | SUCCESS |

e) Dalle informazioni raccolte con l’**analisi statica** e l’**analisi dinamica** possiamo stabilire che il Malware analizzato è un **dropper** contenente un malware che, una volta scaricato dalla sezione risorse del malware al PC della vittima, utilizza la libreria **ADVAPI** e le sue funzioni per modificare una **chiave di registro**, cambiandone il parametro “legittimo” con il path della libreria modificata e caricata dall’eseguibile. Questa libreria modificata catturerà le credenziali di accesso inserite dagli utenti e le registrerà in un file di log nella cartella **WINDOWS\System32** con nome **msutil32.sys**.



Ecco cosa contiene il file, aperto con blocco note:

```
08/16/22 14:35:24 - UN Administrator DM VICTIM-69AE6052 PW AVictim OLD (null)
08/16/22 14:43:14 - UN Administrator DM VICTIM-69AE6052 PW AVictim OLD (null)
08/16/22 15:17:32 - UN Administrator DM VICTIM-69AE6052 PW malware OLD (null)
08/16/22 15:36:24 - UN Administrator DM VICTIM-69AE6052 PW malware OLD (null)
08/16/22 15:52:13 - UN Administrator DM VICTIM-69AE6052 PW malware OLD (null)
08/17/22 15:48:41 - UN Administrator DM VICTIM-69AE6052 PW malware OLD (null)
08/18/22 10:39:53 - UN Administrator DM VICTIM-69AE6052 PW malware OLD (null)
08/18/22 14:42:42 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/18/22 16:32:53 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 16:57:15 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 17:16:29 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
```

2. Analizzando il componente Gina DLL e le azioni che un eventuale attaccante può fare su questo file, si delinea il profilo del Malware e si crea un grafico che ne rappresenti lo scopo:

- a) Cosa succede se il file “.dll” lecito di GINA viene sostituito con un file “.dll” malevolo, che intercetta i dati inseriti;
- b) Sulla base della risposta che si ottiene al punto 1, delineare il profilo del Malware e delle sue funzionalità.
- c) Unire tutti i punti per creare un grafico che ne rappresenti lo scopo ad alto livello.

Svolgimento della traccia:

a) GINA (Graphic Identification & Authentication) è un componente lecito di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica, ovvero permette agli utenti di inserire **username** e **password** nel classico riquadro Windows, come quello in figura sottostante.



Se il file “.dll” legittimo di GINA viene sostituito con un file “.dll” malevolo che intercetta i dati inseriti dagli utenti, l'attacker potrebbe ottenere le credenziali di accesso degli utenti e **accedere** alla loro macchina o addirittura alla rete aziendale, compromettendone la sicurezza. Inoltre, l'attacker potrebbe utilizzare le credenziali rubate per accedere ad altre **risorse sensibili** all'interno della rete aziendale.

b) Dai dati che abbiamo raccolto, possiamo evincere che probabilmente si tratti di un **dropper**. Ricordiamo che il dropper è un tipo di Malware che contiene al suo interno ulteriori software o file malevoli, che vengono **estratti** nel sistema operativo ospite al momento dell'esecuzione.

In questo caso, il malware contenuto all'interno del **dropper** sembra essere un **trojan** che, mascherandosi da libreria lecita, intercetta le credenziali di accesso degli utenti e le registra in un file di log appositamente nascosto nella cartella **System32**: un attaccante potrebbe usare queste credenziali ottenute per accedere al sistema e modificarne contenuti di file, scaricare altri malware e/o modificare le credenziali di accesso per “chiudere fuori” l'utente legittimo.

Infiltrarsi in un sistema informatico mascherandosi da programma legittimo, al fine di estrapolare informazioni personali o danneggiare il dispositivo della vittima, è il comportamento tipico di un **trojan**.

The screenshot displays the VirusTotal analysis page for a file named 'Malware_Build_Week_U3.exe'. At the top left, a 'Community Score' of 47 is shown. A notification states that 47 security vendors and no sandboxes flagged the file as malicious. The file's SHA-256 hash is 57d8d248a8741176348b5d12dcf29f34c8f48ede0ca13c30d12e5ba0384056d7, with a size of 52.00 KB and a submission date of 2023-02-16 00:13:25 UTC. The file type is identified as EXE. Below this, tabs for 'DETECTION', 'DETAILS', 'RELATIONS', 'BEHAVIOR', and 'COMMUNITY' are visible. A section for 'Popular threat label' identifies the file as 'trojan.r014c0dc21/genericrxq'. The 'Security vendors' analysis table lists detections from various vendors, including AhnLab-V3, ALYac, Arcabit, AVG, BitDefender, ClamAV, Cylance, DrWeb, and Emsisoft, all identifying the file as a Trojan or Dropper.

| Security vendors' analysis | Threat categories | Family labels |
|----------------------------|------------------------------------|---------------------------------|
| AhnLab-V3 | Trojan.Win32.Agent.C39204 | Trojan.Win32/Tiggre.5880570c |
| ALYac | Dropped:Trojan.Generic.6200673 | Trojan.Win32.Agent |
| Arcabit | Trojan.Generic.D5E9D61 | Win32:Trojan-gen |
| AVG | Win32:Trojan-gen | TR/Agent.53248.465 |
| BitDefender | Dropped:Trojan.Generic.6200673 | Gen.NN.ZedlaF.36276.aq4@a0clrOb |
| ClamAV | Win.Trojan.Agent-595082 | Malicious.87a7c5 |
| Cylance | Unsafe | Malicious (score: 99) |
| DrWeb | BackDoor.Siggen2.1689 | Malicious (high Confidence) |
| Emsisoft | Dropped:Trojan.Generic.6200673 (B) | Dropped:Trojan.Generic.6200673 |

Oltre a questo, un **trojan** può essere in grado di:

- **Creare backdoor**, consentendo all'attaccante di accedere al sistema operativo della vittima da remoto, senza essere rilevato, tramite una "porta posteriore".
- **Installare ulteriori software dannosi**, come ad esempio adware, spyware, ransomware, worm, virus, ecc.
- **Prendere il controllo del sistema**, modificandone la configurazione e cambiandone le credenziali di accesso.
- **Spiare la vittima**, tenendo sotto controllo le attività dell'utente o accedendo a periferiche multimediali, come microfoni e webcam.

c) Riportiamo di seguito una **rappresentazione grafica** del processo di esecuzione del malware Malware_Build_Week_U3:

