



CORE TECH S.R.L.

2024

CSS 0124

S12 BUILDWEEK

THANKS TO:

Luigi Benvenuti

Marco De Falco

Daniele D'Esposito

Alessandro Marasca

Anthony Midea

INDICE

<u>1.1 PARAMETRI</u>	
pag. 03	
<u>1.2 VARIABILI</u>	
pag. 03	
<u>1.3 SEZIONI</u>	
pag. 04	
<u>1.4.0 LIBRERIE</u>	
pagg. 04-05	
<u>1.4.1 IPOTESI COMPORTAMENTALE</u>	
pagg. 05-06	
<u>2.1 SCOPO FUNZIONE</u>	
pag. 07	
<u>2.2 PASSAGGIO PARAMETRI</u>	
pag. 08	
<u>2.3 LOCAZIONE OGGETTO</u>	
pag. 08	
<u>2.4 SIGNIFICATO ISTRUZIONI</u>	
pag. 09	
<u>2.5 RAPPRESENTAZIONE IN C</u>	
pag. 09	
<u>2.6 VALORE DI “VALUENAME”</u>	
pag. 09	
<u>2.7 FUNZIONALITA’ IMPLEMENTATE DAL MALWARE</u>	
pag. 10	
<u>3.1 PARAMETRO RESOURCE NAME</u>	
pagg. 11-12	
<u>3.2 FUNZIONALITA’ IMPLEMENTATE</u>	
pag. 12	
<u>3.3 ANALISI STATICÀ BASICA</u>	
pag. 12	
<u>3.4 EVIDENZE A SUPPORTO</u>	
pag. 12	
<u>3.5 DIAGRAMMA DI FLUSSO</u>	
pag. 13	
<u>4 - ANALISI DINAMICA</u>	
pagg. 14-16	
<u>5- CONSIDERAZIONI FINALI</u>	
pagg. 17-18	



Nell'indice sono presenti collegamenti ipertestuali verso i capitoli. I vocaboli sottolineati riportano al glossario di pagg. 19-21

GIORNO 1

Con riferimento al file eseguibile **Malware_Build_Week_U3** rispondere ai seguenti quesiti utilizzando i **tool** e le **tecniche** apprese nelle lezioni teoriche:

1. Quanti **parametri** sono passati alla **funzione Main()**?
2. Quante **variabili** sono dichiarate all'interno della funzione Main()?
3. Quali **sezioni** sono presenti all'interno del file eseguibile? Descrivete brevemente **almeno 2** di quelle identificate.
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.

1.1 PARAMETRI

Viene preso il file **Malware_Build_Week_U3** dalla cartella **Build_Week_Unit_3** e caricato all'interno del disassembler **IDA**. Quest'operazione permette di estrarre il codice Assembly dell'eseguibile stesso. In figura la porzione di codice che rappresenta la funzione principale **Main()**.

```
.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
```

I parametri passati alla funzione **Main()** sono tre ed è possibile identificarli sia nella dichiarazione di funzione, sia nella lista degli elementi della funzione stessa e si presentano nella forma indirizzo di **EBP + offset**.

1.2 VARIABILI

Si possono distinguere le variabili dai parametri perché esse sono evidenziate da un valore negativo sottratto all'indirizzo dell' **EBP** (Extended Base Pointer).

In questa porzione di codice ne vengono identificate cinque.



1.3 SEZIONI

Per controllare le singole sezioni del Malware ci si può affidare a due tool:

1. [IDA](#) (precedentemente osservato)

2. [CFF Explorer](#).

Ne osserviamo i risultati nella figura che segue.

Name	Start	End	R	W	X	D	L	Align	Base	Type	Class	AD	es	ss	ds	fs	gs
.text	00000000000401000	00000000000407000	R	.	X	.	L	para	0001	public	CODE	32	0000	0000	0003	FFF...	FFF...
.idata	00000000000407000	000000000004070DC	R	.	.	.	L	para	0002	public	DATA	32	0000	0000	0003	FFF...	FFF...
.rdata	000000000004070DC	00000000000408000	R	.	.	.	L	para	0002	public	DATA	32	0000	0000	0003	FFF...	FFF...
.data	00000000000408000	0000000000040C000	R	W	.	.	L	para	0003	public	DATA	32	0000	0000	0003	FFF...	FFF...

Le sezioni incontrate sono:

1- **.text**: sezione contenente le righe di codice;

2- **.rdata**: sezione info librerie;

3- **.data**: sezione variabili globali;

4- **.rsrc/idata**: contiene le risorse utilizzate dal programma;

1.4.0 LIBRERIE

Utilizzando **CFF EXPLORER** è possibile visualizzare anche le librerie caricate all'interno dell'eseguibile.

Per fare ciò è sufficiente selezionare dal menù a sinistra la cartella “**Import directory**”.

In figura è possibile osservarne il contenuto:

Module Name	Imports	OFTs	TimeStamp	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

Sono presenti due librerie:

1- **Kernel32.dll** (contiene le funzioni principali per l'interazione col sistema).

2- **ADVAPI32.dll** (una libreria avanzata che supporta numerose [API](#) tra cui alcune relative all'ottenimento della [persistenza](#)).

Si noti come alcune fra le funzioni implementate dalla libreria **kernel32.dll** possano essere degli indicatori utili per la classificazione del malware stesso. Fra queste vi sono:

1. **SizeofResource()**: indica le dimensioni della risorsa ricercata.
2. **FindResourceA()**: cerca una specifica risorsa.
3. **LockResource()**: assegna un puntatore a una specifica risorsa.
4. **LoadResource()**: carica la risorsa precedentemente ricercata.
5. **CreateFile()**: crea una file.
6. **WriteFile()**: scrive un file.
7. **LoadLibraryA()**: permette di caricare una libreria in runtime.
8. **GetProcAddress()**: permette di ottenere l'indirizzo di un processo attualmente in esecuzione.

Queste funzioni sono spesso indizi della presenza di un **dropper**, ovvero un trojan che scarta un eseguibile apparentemente innocuo e ne esegue un secondo malevolo.

Oltre a queste funzioni ne troviamo altre due relative alla libreria **ADVAPI32.dll**, che sono:

- **RegSetValueExA()**: imposta il valore di una chiave di registro;
- **RegCreateKeyExA()**: crea una nuova chiave di registro.

L'utilizzo di queste funzioni suggerisce che il Malware stia cercando di aggiungersi ai programmi da eseguire all'avvio del sistema operativo.

1.4.1 IPOTESI COMPORTAMENTALE

Dopo aver ipotizzato che il malware possa essere classificato come dropper si pone l'attenzione sulla ricerca di un secondo file eseguibile all'interno della sezione .rsrc.

Si effettuano dunque ulteriori analisi per ottenere maggiori informazioni riguardanti il programma.

Per prima cosa si cerca di ottenere le stringhe presenti all'interno del codice utilizzando il tool **Strings** presente nella cartella **Software_Malware_Analysis** della macchina **Windows 7**. In figura vediamo alcuni dei risultati più significativi.

```
GetModuleFileNameA
GetModuleHandleA
KERNEL32.dll
RegSetValueExA
RegCreateKeyExA
ADVAPI32.dll
GetCommandLineA
GetVersion
ExitProcess
HeapFree
GetLastError
WriteFile
TerminateProcess
GetCurrentProcess
UnhandledExceptionFilter
FreeEnvironmentStringsA
FreeEnvironmentStringsW
WideCharToMultiByte
GetEnvironmentStrings
GetEnvironmentStringsW
SetHandleCount
GetStdHandle
GetFileType
GetStartupInfoA
GetEnvironmentVariableA
GetVersionExA
HeapDestroy
HeapCreate
VirtualFree
RtlUnwind
HeapAlloc
HeapReAlloc
SetStdHandle
FlushFileBuffers
SetFilePointer
CreateFileA
GetCPIInfo
GetACP
GetOEMCP
GetProcAddress
LoadLibraryA
SetEndOfFile
ReadFile
ExitProcess
GetProcAddress
FreeLibrary
LoadLibraryW
lstrcpyW
GetSystemDirectoryW
DisableThreadLibraryCalls
lstrlenW
GetModuleFileNameW
lstrcpyW
LocalFree
FormatMessageW
KERNEL32.dll
??2@YAPAXIEZ
fclose
fwprintf
_wstrdate
_wstrtime
_wfopen
_vsnwprintf
MSVCRT.dll
free
_initterm
malloc
_adjust_fdiv
RegCloseKey
RegSetValueExW
RegCreateKeyW
ADVAPI32.dll
wsprintf
USER32.dll
!_#%z&'<>*,-./01
gina.dll
DllRegister
DllUnregister
ShellShutdownDialog
WlxActivateUserShell
WlxDisconnectNotify
WlxDisplayLockedNotice
WlxDisplaySASNotice
WlxDisplayStatusMessage
```

Tra le stringhe trovate ce ne sono alcune relative alla libreria **GINA.dll**, ovvero una libreria standard di windows che si occupa della gestione dei servizi di **log on** su server.

Inoltre è possibile individuare una funzione chiamata **WlxLoggedOutSAS** che da analisi più approfondita è risultata essere comunemente sfruttata per sostituire ingannevolmente la libreria autorizzata alla gestione delle operazioni di autenticazione.

Una volta ottenute queste informazioni recuperiamo la firma univoca dell'eseguibile al fine di controllarne precedenti identificazioni tramite **Virus Total**, ovvero un sito web che permette l'analisi gratuita di file per scovare eventuali Malware.

La firma del file si può ottenere utilizzando il tool **MD5Deep**, in grado di restituirci la stringa hash associata univocamente all'eseguibile stesso.

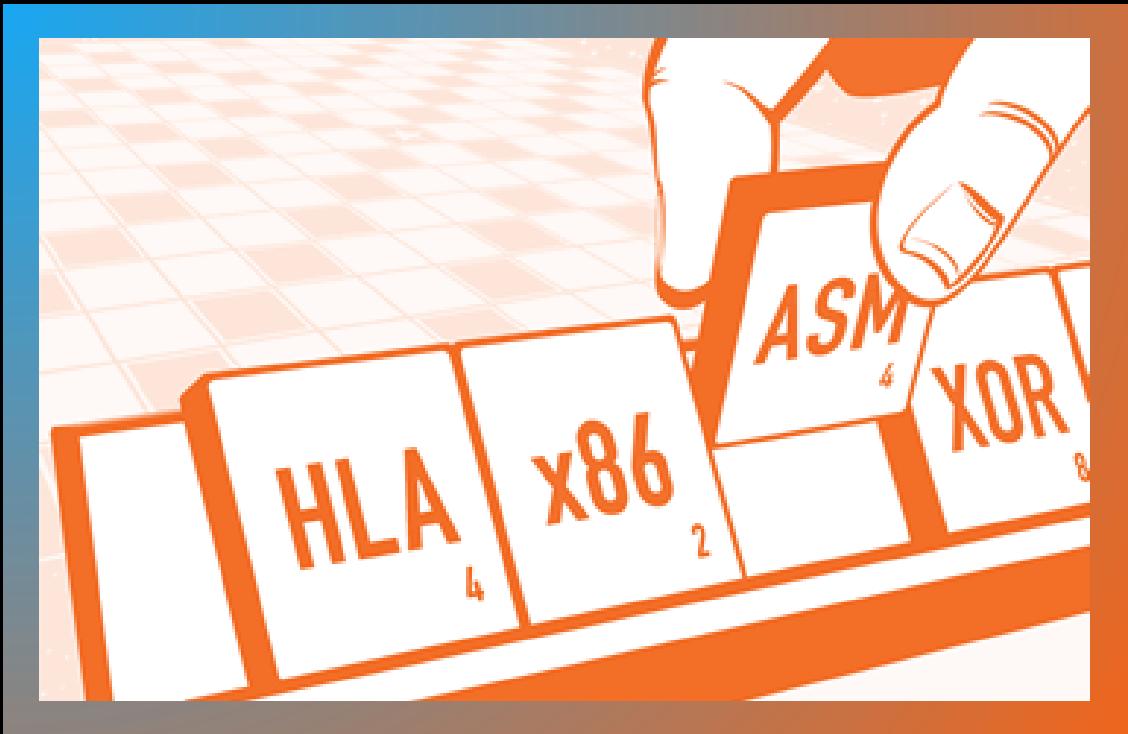
```
C:\Users\user\Desktop\Software Malware analysis\md5deep-4.3\md5deep-4.3>md5deep
C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\Malware_Build_Week_U3.exe
md5deep: WARNING: You are running a 32-bit program on a 64-bit system.
md5deep: You probably want to use the 64-bit version of this program.
a9c55bb87a7c5c3c923c4fa12940e719  C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\Malware_Build_Week_U3.exe
```

Vendor	Detection
AhnLab-V3	Trojan/Win32.Agent.C39204
AllCloud	Backdoor
Anti-AVI	Trojan/Win32.Agent
Avast	Win32:Trojan-gen
Avira (no cloud)	TR/Agent.L3248.465
BitDefenderTheta	Gen:NN.Zedaf.F.36802.Aq4@a0c1r0b
ClamAV	Win.Trojan.Agent.595082
CyberReason	Malicious.B7a7c5
Cynet	Malicious (score: 99)
DWeb	BackDoor.Sigged2.1689
Emsisoft	Gen:Variant.Dolna.65814 [B]
ESET-NOD32	Win32/Agent.WQJ
GData	Gen:Variant.Dolna.65814
Kingssoft	Win32.inf.undefined
Malwarebytes	Backdoor.Agent.MSGN
MaxSecure	Trojan.Malware.4145763.sungen
Microsoft	Trojan/Win32/Totbrick.MTB
Rising	Trojan.Agent.B8.E (TFE:5MaGuq4FVE)
Skyhigh (SWG)	GenericRCO-GT169C5B887A7C
Ikarus	Trojan-Dropper.Agent
Ionic	Trojan/Win32.Totbrick.41c
MAX	Malware (ai Score:94)
McAfee	GenericRCO-GT169C5B887A7C
NANO-Antivirus	Trojan/Win32.Agent.doujh
Sangfor Engine Zero	Trojan/Win32.Totbrick.Hv9
Sohos	Mal/Generic-S

Il valore ottenuto è **a9c55bb87a7c5c3c923c4fa12940e719**.

Caricandolo su Virus Total si ottengono le classificazioni precedentemente riportate dalle fonti disponibili.

GIORNO 2



TRACCIA

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**. (se serve, valutate anche un'altra o altre due righe assembly)
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**»?
7. Nel complesso delle due funzionalità appena viste, spiegate quale funzionalità sta implementando il Malware in questa sezione.

2.1 SCOPO FUNZIONE

```

    .text:00401017      push    offset SubKey ; "SOFTWARE\\Micros
    .text:0040101C      push    8000002h ; 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_401078

```

La funzione chiamata alla locazione di memoria **00401021** è **RegCreateKeyExA**: un metodo della libreria di **Windows API** utilizzato per creare una nuova chiave o aprirne una esistente nel Registro di sistema di Windows.

Questa funzione viene spesso utilizzata per **manipolare** le chiavi nel Registro di sistema da un'applicazione Windows o da un eventuale malware per ottenere la persistenza sul sistema.

```

LSTATUS RegCreateKeyExA(
    [in]          HKEY           hKey,
    [in]          LPCSTR         lpSubKey,
    [in]          DWORD          Reserved,
    [in, optional] LPSTR          lpClass,
    [in]          DWORD          dwOptions,
    [in]          REGSAM         samDesired,
    [in, optional] const LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    [out]         PHKEY          phkResult,
    [out, optional] LPDWORD        lpdwDisposition
);

```

2.2 PASSAGGIO PARAMETRI

```

    .text:00401000      push   ebp
    .text:00401001      mov    ebp, esp
    .text:00401003      push   ecx
    .text:00401004      push   0          ; lpdwDisposition
    .text:00401004      lea    eax, [ebp+hObject]
    .text:00401005      push   eax        ; phkResult
    .text:00401006      push   0          ; lpSecurityAttributes
    .text:00401007      push   0F003Fh   ; samDesired
    .text:00401008      push   0          ; dwOptions
    .text:00401009      push   0          ; lpClass
    .text:0040100A      push   0          ; Reserved
    .text:0040100B      push   offset SubKey ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
    .text:0040100C      push   8000002h   ; hKey
    .text:00401021      call   ds:RegCreateKeyExA

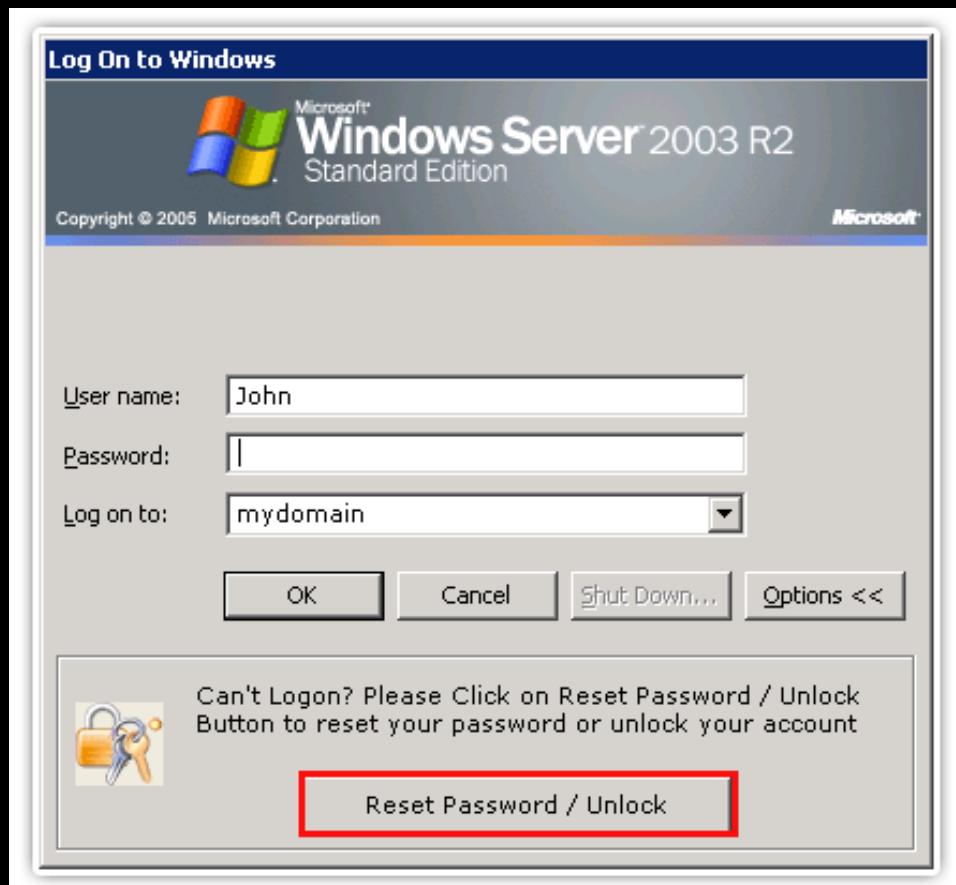
```

I parametri della funzione vengono passati tramite **push** sullo stack con successiva chiamata alla funzione tramite **call**.

2.3 LOCAZIONE OGGETTO

Alla locazione **00401017** viene pushato nello stack l'oggetto **lpSubKey**; esso rappresenta il nome di una sottochiave aperta o creata dalla funzione. La sottochiave specificata deve essere identificata dal parametro **hKey**, il quale non può essere **NUL**. Nel caso del nostro malware la chiave del registro da modificare si trova al percorso "**SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon**"; essa rappresenta una parte specifica del Registro di sistema di Windows che contiene le impostazioni relative all'autenticazione nello stesso, noto come **WINLOGON**. Questa chiave contiene i valori e le impostazioni riguardanti il comportamento e l'interfaccia grafica associata alla finestra di autenticazione a Windows; in particolare alcuni dei valori che si trovano all'interno di questa chiave riguardano **GINA** (Graphical Identification and Authentication), ossia una **dll (Dynamic-link library)** implementata nel sistema operativo Windows che fornisce procedure personalizzabili di identificazione e autenticazione degli utenti.

GINA è anche responsabile per la presentazione dell'interfaccia grafica di accesso a Windows, fornendo il form nel quale gli utenti inseriscono nome utente e password per accedere al sistema.



2.4 SIGNIFICATO ISTRUZIONI

• .text:00401027	test	eax, eax
• .text:00401029	jz	short loc_401032
• .text:0040102B	mov	eax, 1
• .text:00401030	jmp	short loc_40107B

In **Assembly**, l'istruzione **test** viene utilizzata per eseguire un'operazione logica di **AND** tra due operandi senza memorizzare il risultato, ma impostando solo i flag della **CPU** in base al risultato dell'operazione.

Nel caso in esame viene eseguito il **test eax, eax**: questa operazione permette di verificare quale sia il valore del registro.

Se il risultato è zero, allora tutti i bit del registro eax sono impostati a zero. Viceversa, se il risultato non fosse zero, almeno uno dei bit nel registro eax è impostato su uno.

L'operando impone di conseguenza i flag del registro EFLAGS nella CPU (in particolare lo Zero Flag).

Se eax è zero, **ZF** viene impostato a 1; se eax non è zero, ZF viene impostato a 0.

Proseguendo nella lettura del codice, l'istruzione **jz (jump if zero)** indica un salto condizionale all'indirizzo specificato, che viene eseguito solo se ZF è impostato a 1.

Nel caso eax non sia uguale a zero, l'esecuzione del codice procede con le istruzioni **mov eax,1** (che assegna al registro eax il valore 1) e con un successivo salto incondizionato (**jmp**) all'indirizzo di memoria **loc_40107B**.

Si può dedurre che l'eseguibile stia implementando un **controllo** sul flusso d'esecuzione; inserendo la valutazione in un contesto più generale, notiamo che questo controllo avviene dopo la chiamata di funzione **RegCreateKeyExA()**, che si occupa di creare o modificare una chiave di registro.

L'ipotesi più accreditata è che in eax venga scritto il valore di phkResult, restituito in output dalla funzione stessa, e che il malware salti alla modifica della chiave se essa è stata creata o aperta con successo.

Il malware chiuderà invece l'esecuzione se si sono verificati errori,

2.5 RAPPRESENTAZIONE IN C

Le precedenti righe di codice analizzate rappresentano in C il costrutto di selezione **If**.

Di seguito la traduzione del codice assembly in C:

```
If (x==0) {
    goto loc_401032;
}
```

2.6 VALORE DI “ValueName”

• .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	

Il valore del parametro **ValueName** passato alla funzione **RegSetValueExA** è **“GinaDLL”**; lo vediamo anche analizzando la porzione di codice in cui viene definito il suo valore tramite l'istruzione db “GinaDLL”, 0 che indica la definizione di una variabile stringa che contiene i caratteri “GinaDLL”.

* .data:004008048 aRi	db	'RI', 0Ah, 0	; DATA XREF: sub_401000:loc_401062↑o
* .data:00400804C ; char ValueName[]			
.data:00400804C ValueName	db	'GinaDLL', 0	; DATA XREF: sub_401000+3E↑o
* .data:004008054 ; char SubKey[]			
.data:004008054 SubKey	db	'SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon', 0	; DATA XREF: sub_401000+17↑o
.data:004008054			

2.7 FUNZIONALITA' IMPLEMENTATE DAL MALWARE

Da una ricerca più approfondita emerge come la modifica della sottochiave di registro

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL

sia un comportamento tipicamente associabile ad un tentativo di sostituzione dell'interfaccia di login autentica con un'altra **GUI** malevola, probabilmente volta a sottrarre i dati inseriti dagli utenti.

Quest'operazione è indicativa del tentativo di ottenimento di **persistenza** da parte del malware.

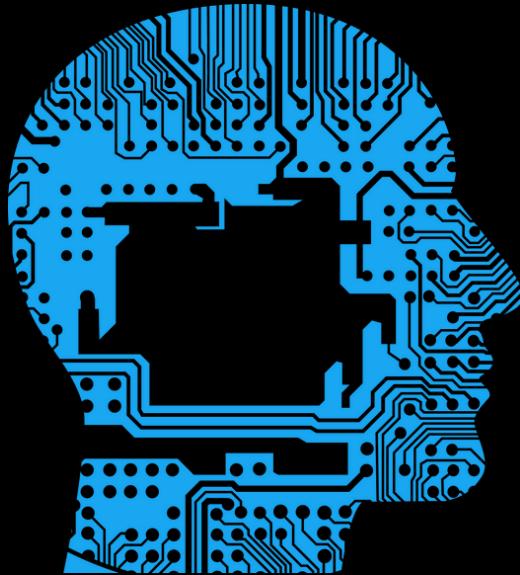
Loading and Running a GINA DLL

Windows loads and executes the standard Microsoft GINA DLL (MSGina.dll). To load a different [GINA](#), you must alter the following registry key value:

```
HKEY_LOCAL_MACHINE
  Software
    Microsoft
      Windows NT
        CurrentVersion
          Winlogon
            GinaDLL<dl>
<dt>
  Data type
</dt>
<dd>      REG_SZ</dd>
</dl>
```

If the GinaDLL key value is present, it must contain the name of a GINA DLL, which [Winlogon](#) will load and use.

GIORNO 3



Riprendete l'analisi del codice, **analizzando** le routine tra le locazioni di memoria **00401080** e **00401128**:

1. Qual è il valore del parametro «**ResourceName**» passato alla funzione **FindResourceA()**;
2. Il susseguirsi delle chiamate di funzione che effettua il Malware in questa sezione di codice l'abbiamo visto durante le lezioni teoriche. Che **funzionalità** sta implementando il Malware?
3. È possibile identificare questa funzionalità utilizzando l'**analisi statica basica**? (dal giorno 1 in pratica)
4. In caso di risposta affermativa, **elencare le evidenze a supporto**. Entrambe le funzionalità principali del Malware viste finora sono richiamate all'interno della funzione **Main()**.
5. **Disegnare un diagramma di flusso** (inserite all'interno dei box solo le informazioni circa le funzionalità principali) che comprenda le 3 funzioni.

3.1 PARAMETRO RESOURCE NAME

```

.text:00401088
.text:00401088 loc_401088:
→ .text:00401088        mov    eax, lpType           ; CODE XREF: sub_401088+2F↑j
* .text:00401088        push   eax                ; lpType
* .text:0040108D        mov    ecx, lpName          ; lpName
* .text:0040108E        push   ecx                ; lpName
* .text:004010C4        push   edx, [ebp+hModule]      ; hModule
* .text:004010C5        push   edx                ; hModule
* .text:004010C8        call   ds:FindResourceA
* .text:004010C9        mov    [ebp+hResInfo], eax
* .text:004010CF        mov    [ebp+hResInfo], 0
* .text:004010D2        cmp    [ebp+hResInfo], 0
* .text:004010D6        jnz    short loc_4010DF
* .text:004010D8        xor    eax, eax
* .text:004010DA        jmp    loc_4011BF
* .text:004010DF        :

```

La funzione **FindResourceA()** è una funzione fornita dalle **API** di Windows.

```

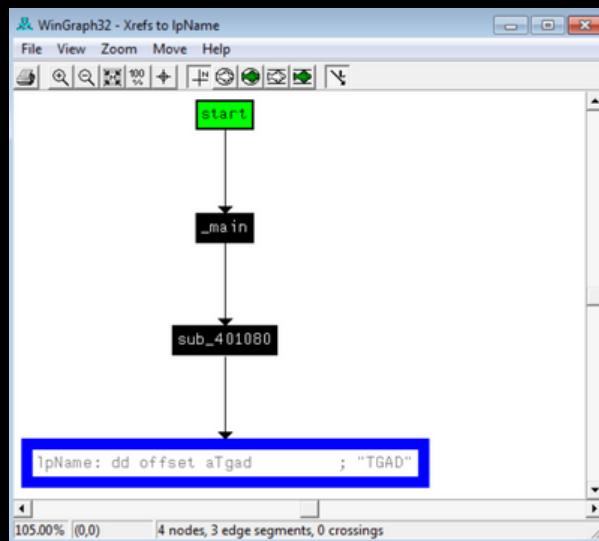
HRSRC FindResourceA(
    [in, optional] HMODULE hModule,
    [in]           LPCSTR  lpName,
    [in]           LPCSTR  lpType
);

```

Il parametro **ResourceName** di nostro interesse è rappresentato dalla variabile **lpName**, un tipo stringa (**LPCSTR**).

Abbiamo due modi per ottenere il suo valore: attraverso l'analisi statica o dinamica.

Nel caso dell'analisi statica utilizziamo **IDA**; una volta trovata la variabile **lpName** nel codice facciamo clic con il tasto destro e selezioniamo **Chart of xrefs to** (la funzione che mostra tutte le referenze a un determinato indirizzo o valore nel codice assembly; in pratica permette di vedere tutte le istruzioni o i luoghi nel codice in cui un particolare indirizzo o valore è utilizzato o referenziato).



Osservando che lpName è correlata alla variabile aTgad, ci spostiamo nella parte del codice che riguarda aTgad.

```

.lp32
.data:004008030 ; LPCSTR lpType
.data:004008030 lpType dd offset aBinary ; DATA XREF: sub_401080:loc_4010B8Tr
; "BINARY"
.data:004008034 ; LPCSTR lpName
.data:004008034 lpName dd offset aTgad ; DATA XREF: sub_401080+3ETr
; "TGAD"
.data:004008038 aTgad db 'TGAD',0 ; DATA XREF: .data:lpNameTo
.align 10h
.data:004008040 aBinary db 'BINARY',0 ; DATA XREF: .data:lpTypeTo
.align 4
.data:004008047

```

L'istruzione **lpName dd offset aTgad** indica che lpName è un puntatore all'offset della variabile aTgad; quindi lpName contiene l'offset della variabile aTgad e può essere utilizzato per accedere al valore di aTgad nel codice, poiché punta all'indirizzo di memoria dove si trova la variabile aTgad.

L'istruzione aTgad db "TGAD", 0 indica la definizione di una variabile stringa di nome aTgad che contiene i caratteri "TGAD". Pertanto deduciamo che il valore di lpName (ResourceName) è "TGAD".

Nel caso dell'analisi dinamica lanciamo [OllyDbg](#) e impostiamo un **breakpoint** alla chiamata della funzione **FindResourceA()** e facciamo partire il debugger: interromperà l'esecuzione del codice automaticamente una volta arrivati al breakpoint settato, da qui possiamo vedere i valori dei parametri passati alla funzione. Troviamo che il valore di lpName (ResourceName) è "TGAD".

<pre> 004010A4 .C745 F4 00000 MOV DWORD PTR SS:[EBP-C],0 004010AB .837D 08 00 CMP DWORD PTR SS:[EBP+8],0 004010AF .75 07 JNZ SHORT Malware_.004010B8 004010B1 .33C0 XOR EAX,EAX 004010B3 .E9 07010000 JMP Malware_.004011BF 004010B8 > A1 30804000 MOU EAX,DWORD PTR DS:[408030] 004010BD .50 PUSH EAX 004010BE .8B0D 34804000 MOU ECX,DWORD PTR DS:[408034] 004010C4 .51 PUSH ECX 004010C5 .8B55 08 MOU EDX,DWORD PTR SS:[EBP+8] 004010C8 .52 PUSH EDX 004010C9 .FF15 28704000 CALL DWORD PTR DS:[<&KERNEL32.FindResou 004010CF .8945 EC MOV DWORD PTR SS:[EBP-14],EAX 004010D2 .897D EC 00 CMP DWORD PTR SS:[EBP-14],0 004010D6 .75 07 JNZ SHORT Malware_.004010DF 004010D8 .33C0 XOR EAX,EAX </pre>	ResourceType => "BINARY" Malware_.00408038 ResourceName => "TGAD" hModule FindResourceA
--	--

3.2 FUNZIONALITA' IMPLEMENTATE

Il susseguirsi delle chiamate di funzione **FindResourceA**, **LoadResource**, **LockResource** e **SizeofResource** rappresenta il comportamento tipico di un malware di tipo **dropper**.

Un dropper è un programma malevolo che contiene al suo interno un malware. Nel momento in cui viene eseguito, inizia la sua esecuzione ed estrae il malware contenuto per salvarlo sul disco.

Generalmente, il malware incluso nel dropper è contenuto nella sezione **.rss** dell'eseguibile, ovvero nella sezione risorse (talvolta anche identificata con **.rsc**).

I dropper hanno delle caratteristiche distintive piuttosto singolari; per estrarre il malware contenuto nella sezione delle risorse utilizzano delle API come: **FindResourceA()**, **LoadResource()**, **LockResource()** e **SizeOfResource()**, esattamente quelle contenute nel file analizzato. Queste API permettono di localizzare all'interno della sezione risorse il malware da estrarre, e successivamente da caricare in memoria per l'esecuzione o da salvare sul disco per esecuzione futura.

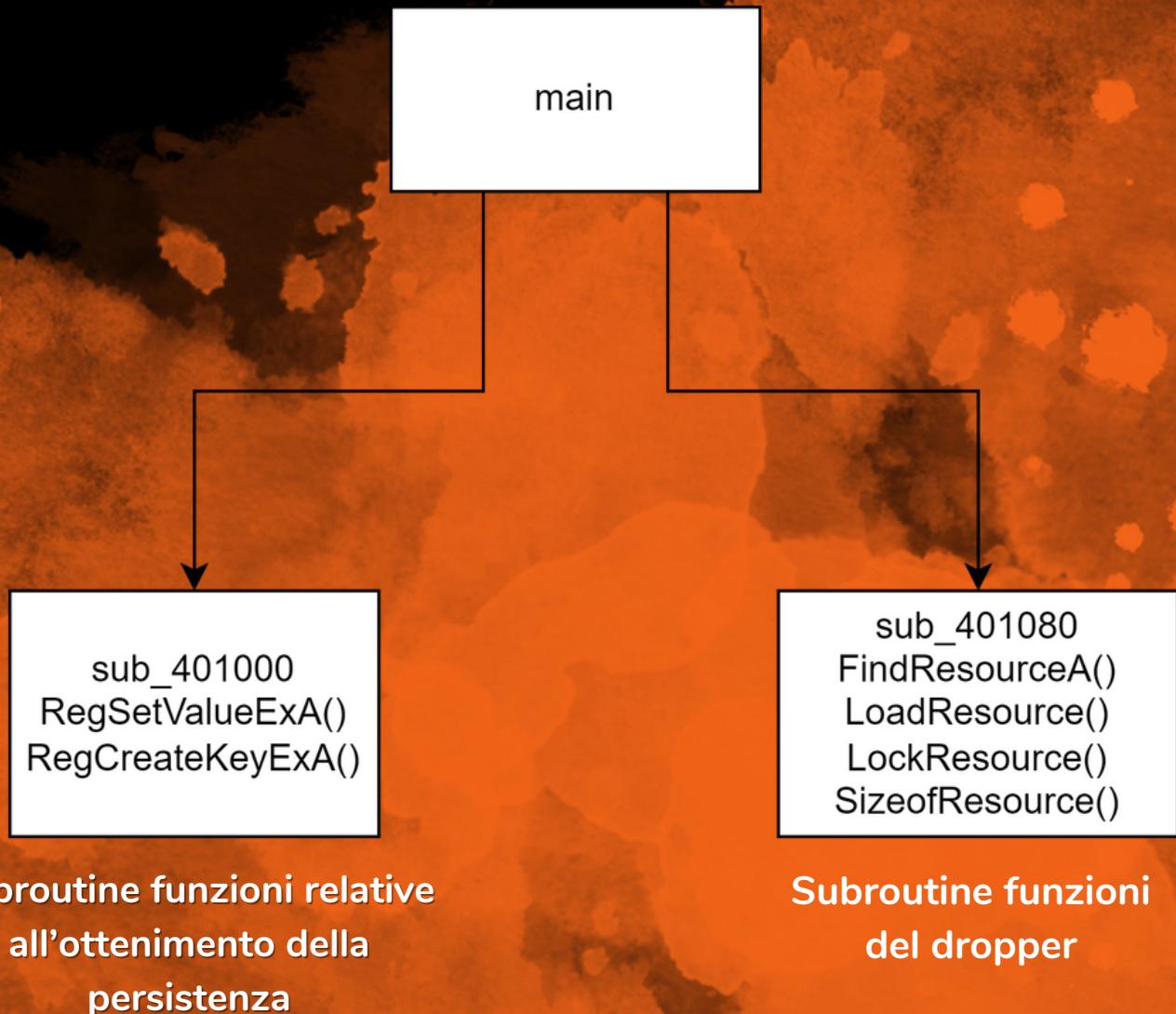
3.3 ANALISI STATICÀ BASICA

Si può effettivamente riuscire a capire la ricerca dell'ottenimento della persistenza come fatto nel giorno 1, ovvero analizzando la presenza delle funzioni ad essa tipicamente relative.

3.4 EVIDENZE A SUPPORTO

Le evidenze a supporto di questa tesi sono numerose; innanzitutto la presenza di contenuto all'interno della sezione risorse (.rsrc), ma anche il tentativo di nascondere alcune elementi tramite il caricamento delle librerie in runtime e il codice hash univoco possono essere d'aiuto.

3.5 DIAGRAMMA DI FLUSSO



GIORNO 4

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 .

1. Quale chiave di registro **viene creata**?
2. Quale valore **viene associato** alla chiave di registro creata?
3. 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**.

ANALISI DINAMICA BASICA

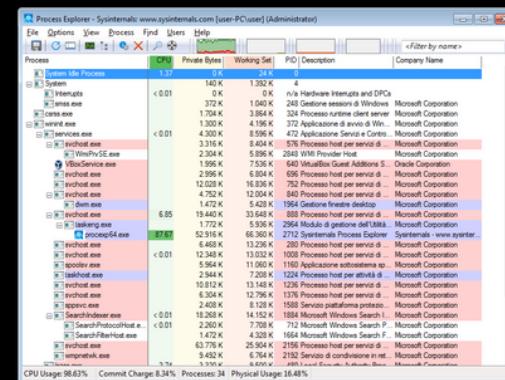
Per eseguire il malware in maniera sicura, va configurato correttamente un **ambiente virtuale** nel quale operare.

È inoltre buona norma eseguire **un'istantanea** (ovvero un punto di ritorno alla prima esecuzione del malware) a cui tornare successivamente all'esecuzione.

L'**analisi dinamica basica** segue una procedura di svolgimento abbastanza stagna, che è la seguente:

- avviare **Process Explorer**;
- avvio e setup **ApateDNS**;
- primo snapshot con **RegShot**;
- avvio **Process Monitor**;
- avvio cattura traffico di rete con **Wireshark**;
- esecuzione del malware;
- stop cattura **Wireshark** e **Process Monitor**;
- secondo snapshot **RegShot**;
- stop **ApateDNS**;
- stop **Process Explorer**.

Di seguito i vari passaggi dettagliati con i risultati ottenuti dalla seguente analisi dettagliata.



1. Apertura Process Explorer

Aprendo Process Explorer si procede alla verifica dello stato dei processi.

2. Modifica DNS per ApateDNS

Con ApateDNS è possibile monitorare lo stato di risoluzione dei nomi, impostando un filtro sull'indirizzo del gateway.

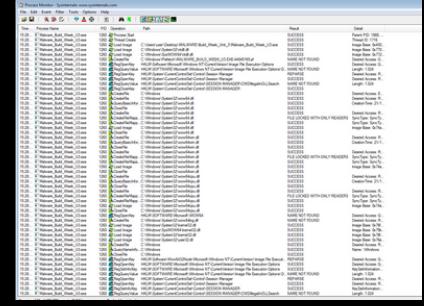
3. Prima istantanea RegShot

La prima istantanea eseguita con RegShot permette di capire lo stato delle chiavi di registro prima dell'esecuzione del malware.



4. Avvio ProcMon

Viene avviato Process Monitor al fine di verificare quali processi verranno associati all'esecuzione del malware.



6. Wireshark

Wireshark viene utilizzato per monitorare i traffici di rete.

Nel malware in analisi, non sono presenti librerie che lasciano intendere un tentativo di connessione e Wireshark conferma questa ipotesi.

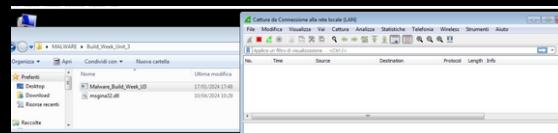
7. Avvio del Malware

Si esegue il malware
Malware_Build_Week_U3.exe



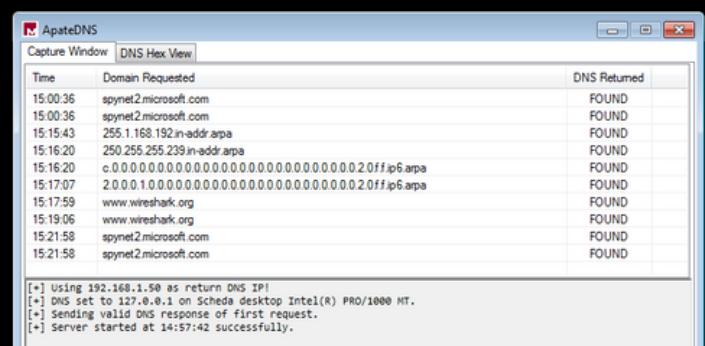
9. Risultato ApateDNS

ApateDNS analizza i nomi di dominio risolti tramite richiesta ad un server DNS dal malware.



8. Seconda istantanea RegShot

La seconda istantanea eseguita con RegShot permette di capire lo stato delle chiavi di registro dopo l'esecuzione del malware.



RISPOSTE

1. Modifiche malware sul registro

Nell'interfaccia di ProcMon, si può impostare un filtro che permetta la visione delle sole modifiche effettuate sulla chiave di registro, visibile nella figura successiva.



Viene richiesto l'accesso alla sottochiave di registro “**HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\WINLOGON**”.

2. Valore della chiave di registro

Si può osservare che, a causa dell'aggiornamento di sicurezza relativo al sistema operativo Windows 7 (l'eseguibile è infatti ideato per Windows XP) il malware non ottiene il permesso per accedere alla chiave di registro necessaria all'assegnazione del valore “**GinaDLL**”.

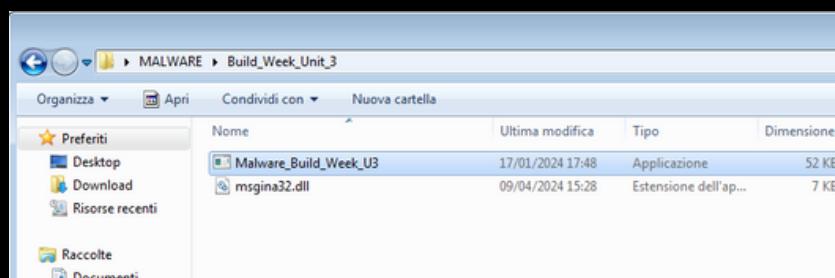
Time	Process	Action	Target Path	Result	Details
15:20...	Malware_Build_Week_U3.exe	RegCreateKey	HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	
15:20...	Malware_Build_Week_U3.exe	RegSetValue	HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	ACCESS DENIED	Desired Access: All Access, Disposition: REG_OPENED_E... KeySetInformationClass: KeySetHandleTagsInformation, Le... Query: HandleTags, HandleTags: 0x400 Type: REG_SZ, Length: 320, Data: C:\Users\user\Desktop\...
15:20...	Malware_Build_Week_U3.exe	RegCloseKey	HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	
15:20...	Malware_Build_Week_U3.exe	RegSetValue	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options	SUCCESS	
15:20...	Malware_Build_Week_U3.exe	RegCloseKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options	SUCCESS	
15:20...	Malware_Build_Week_U3.exe	RegSetValue	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Nls\Setting\Versions	SUCCESS	
15:20...	Malware_Build_Week_U3.exe	RegCloseKey	HKEY_LOCAL_MACHINE	SUCCESS	

3. Modifica cartella malware

Il malware ha creato all'interno della sua cartella una nuova interfaccia GINA, chiamata “**msgina32.dll**”, alterata in modo tale da intercettare i dati inseriti dagli utenti.

Time	Process	Action	Target Path	Result
15:20...	Malware_Build_Week_U3.exe	CreateFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS
15:20...	Malware_Build_Week_U3.exe	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
15:20...	Malware_Build_Week_U3.exe	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
15:20...	Malware_Build_Week_U3.exe	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
15:20...	Malware_Build_Week_U3.exe	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
15:20...	Malware_Build_Week_U3.exe	QueryNameInfo	C:\Windows\System32\apisetschema.dll	SUCCESS
15:20...	Malware_Build_Week_U3.exe	QueryNameInfo	C:\Windows\System32\Malware_Build_Week_U3.exe	SUCCESS
15:20...	Malware_Build_Week_U3.exe	QueryNameInfo	C:\Windows\System32\wow64win.dll	SUCCESS
15:20...	Malware_Build_Week_U3.exe	QueryNameInfo	C:\Windows\System32\wow64.dll	SUCCESS
15:20...	Malware_Build_Week_U3.exe	QueryNameInfo	C:\Windows\System32\wow64cpu.dll	SUCCESS
15:20...	Malware_Build_Week_U3.exe	QueryNameInfo	C:\Windows\SysWOW64\cryptbase.dll	SUCCESS

La chiamata di sistema che si occupa di creare il file è mostrata in figura.



GIORNO 5

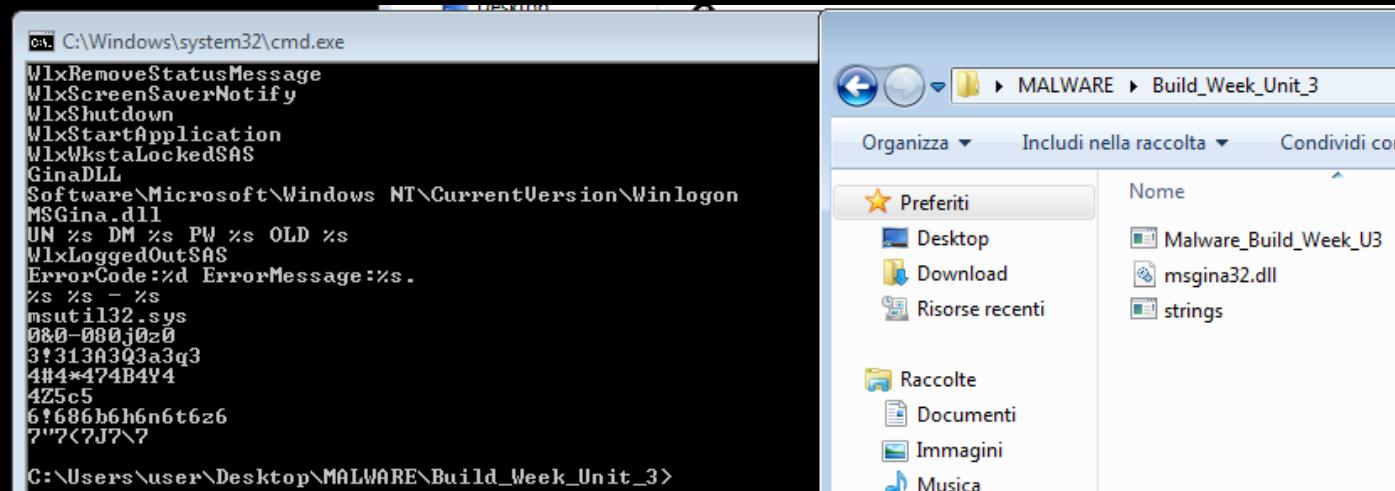
GINA (Graphical identification and authentication) è un componente lecito di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica - ovvero permette agli utenti di inserire username e password nel classico riquadro Windows, come quello in figura a destra che usate anche voi per accedere alla macchina virtuale.

- Cosa può succedere se il file .dll lecito **viene sostituito** con un file .dll malevolo, che intercetta i dati inseriti?

Sulla base della risposta sopra, **delineate il profilo del Malware** e delle sue funzionalità.

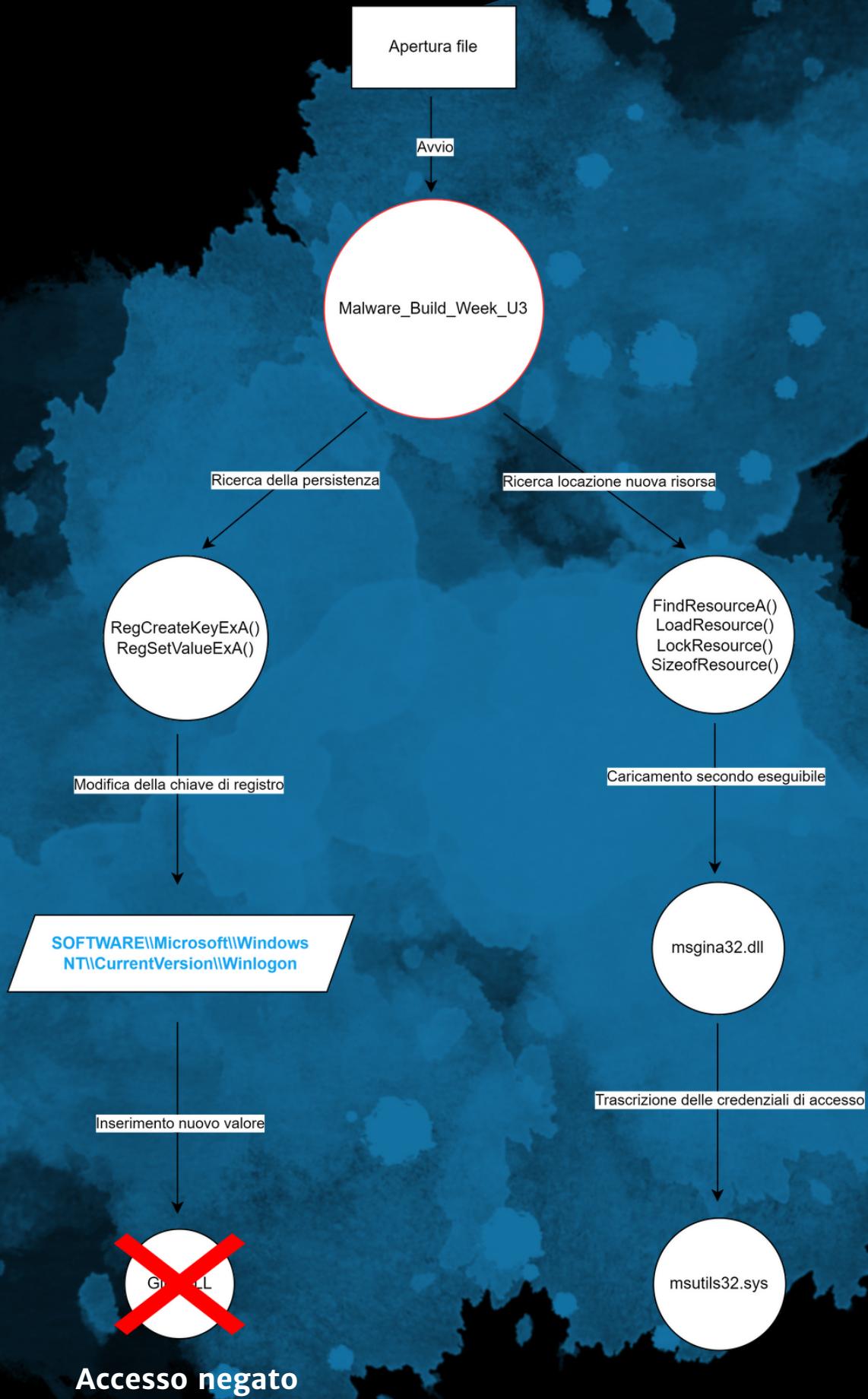
Unite tutti i punti per **creare un grafico** che ne rappresenti lo scopo ad alto livello.

Questo malware è un dropper che installa all'intero del sistema operativo un altro malware.



I dati ottenuti applicando il tool **Strings** sul secondo eseguibile (msgina32.dll) confermano ulteriormente gli indizi che suggerivano che esso sia classificabile come **credential stealer**, ossia un software malevolo progettato per rubare le credenziali di accesso degli utenti attraverso diverse metodiche.

Nel caso di questo in esame, il programma acquisisce le credenziali sostituendo la graphic interface originale con una personalizzata in modo tale che trascriva su un altro file (msutil32.sys) le credenziali inserite.



GLOSSARIO

ANALISI DINAMICA

• Avanzata

Viene eseguita l'analisi dinamica del malware attraverso l'utilizzo di un debugger, ovvero uno strumento di sviluppo molto specializzato che si collega all'app in esecuzione e consente di controllare il codice.

• Basica

Prevede l'esecuzione sicura in ambiente di laboratorio virtuale del malware, al fine di analizzarne il comportamento in runtime.

ANALISI STATICÀ

• Avanzata

Prevede l'utilizzo di un disassembler, ovvero un software in grado di fornire le istruzioni Assembly che compongono un eseguibile.

• Basica

Analisi di un malware a livello superficiale; non prevede l'avvio dell'eseguibile, ma si limita ad analizzarne l'header per vedere le sezioni che lo compongono e le librerie che utilizza.

APATE DNS:

Tool che funge da server DNS utilizzato per intercettare tutte le richieste generate da un malware verso i domini Internet

API

Le API (Application Programming Interface) sono delle funzioni specifiche che permettono la comunicazione fra un eseguibile ed il sistema operativo.

È particolarmente importante conoscerne le funzionalità nella fase di comprensione del comportamento di un malware perché le API ci permettono di capire nel dettaglio come il programma si interfacci con il sistema operativo sul quale opera.

BREAKPOINT

Un breakpoint è una funzionalità di debugging che permette di arrestare l'esecuzione di un programma in un dato istante di tempo.

Ne esistono di 3 tipi:

- software breakpoint: ferma il programma all'esecuzione di una data istruzione;
- hardware breakpoint: stoppa l'esecuzione ad un indirizzo di memoria specifico;
- breakpoint condizionale: arresta il flusso di esecuzione al verificarsi di una condizione.

CFF EXPLORER:

CFF Explorer è un software utilizzato per l'analisi dei file eseguibili; Il termine "CFF" sta per "Cavalry File Format", che si riferisce alla struttura dei file eseguibili di Windows.

CHIAVI DI REGISTRO WINDOWS

Il registro di Windows si suddivide in cinque sottocartelle, dette root key.

- **HKEY_LOCAL_MACHINE**, contiene i record e le configurazioni della macchina;
- **HKEY_CURRENT_USER**, contiene i record e le configurazioni dell'utente corrente;
- **HKEY_CLASSES_ROOT**, contiene le informazioni su apertura file ed estensioni;
- **HKEY_CURRENT_CONFIG**, contiene impostazione e configurazioni hardware;
- **HKEY_USERS**, configurazioni comuni per gli utenti di sistema.

COSTRUTTI

I costrutti indicano all'esecutore l'ordine in cui le operazioni devono essere eseguite. Essi devono essere scelti in modo da rispecchiare quanto più possibile i meccanismi che naturalmente vengono usati quando si descrive (ad esempio in italiano) una qualsiasi successione di operazioni. Si dividono in costrutti di sequenza, selezione (if, if-else ecc..) ed iterazione (cicli).

GLOSSARIO

DIAGRAMMA DI FLUSSO

Un diagramma di flusso è una rappresentazione grafica di un processo o di un algoritmo. È composto da una serie di simboli e frecce che indicano la sequenza di passaggi o decisioni all'interno del processo.
nella malware analysis vengono utilizzati per descrivere il comportamento ad alto livello delle applicazioni malevoli.

FUNZIONI

Una funzione è un'unità di organizzazione del codice che permette di raggruppare una sequenza di istruzioni in un unico blocco, caratterizzato da un nome, dei parametri in ingresso e uno o più dati restituiti in uscita.

GESTIONE MEMORIA

La memoria RAM (Random Access Memory) viene allocata in diverse porzioni (in base al numero di applicazioni in funzione in un dato istante).

Ogni porzione viene ulteriormente divisa in segmenti:

- segmento **Code/Text**: contiene le istruzioni da eseguire
- segmento **Data**: contiene variabili globali
- segmento **Heap**: un segmento allocato dinamicamente e accessibile dall'utente che contiene dati permanenti a meno che non venga terminata l'esecuzione del programma o gli stessi dati vengano eliminati.
- segmento **Stack**: contiene i parametri e le variabili locali utili all'esecuzione di una porzione di codice specifica. Lo stack è una pila, ovvero una struttura dati di tipo LIFO (last in first out) nel quale è possibile inserire (PUSH) o togliere (POP) dati solamente dalla testa della pila.

IDA PRO

L'Interactive Disassembler (IDA), è un disassembler comunemente utilizzato per il reverse engineering, ovvero l'applicazione dell'analisi statica dinamica al fine di comprendere dalle singole istruzioni Assembly il funzionamento generale di un eseguibile.

Supporta numerosi formati di file eseguibili per diversi processori e sistemi operativi.

LIBRERIE

Le librerie sono insiemi di funzioni o API predefinite.

Sono solitamente predisposte per essere riutilizzate da altri programmi software attraverso un'opportuna procedura di collegamento.

MALWARE (MALICIOUS SOFTWARE)

termine generico che indica software dannosi progettati per compromettere o sfruttare qualsiasi tipo di dispositivo, servizio o rete programmabile.

MD5DEEP:

MD5Deep è uno strumento software progettato per calcolare e confrontare le somme di controllo MD5 (Message Digest Algorithm 5) di file e directory. L'MD5 è un algoritmo di hash crittografico ampiamente utilizzato per generare una rappresentazione univoca (hash) di dati, spesso utilizzato per verificare l'integrità dei file.

OGGETTO

Con il termine oggetto, in informatica ed in particolar modo nell'ambito della programmazione, si intende nella maniera più generica una regione di memoria allocata.

OLLYDBG

OllyDbg è un debugger per sistemi Microsoft Windows, disponibile solo in versione a 32 bit. Traccia i registri, riconosce le funzioni e i parametri passati alle principali librerie standard, le chiamate alle API del sistema operativo, eventuali salti condizionali, tabelle, costanti, variabili e stringhe, esplicitando il tutto in istruzioni Assembly e permettendo l'inserimento di breakpoint al fine di bloccare il flusso d'esecuzione di uno script.

GLOSSARIO

PARAMETRI

I parametri sono gli identificatori associati internamente ai valori che vengono passati ad una funzione. Nel linguaggio Assembly x86 i parametri vengono scritti nella forma nomevariabile [ebp+offset].

PERSISTENZA

Il concetto di persistenza si riferisce alla caratteristica dei dati di un programma di sopravvivere all'esecuzione del programma stesso che li ha creati: senza questa capacità verrebbero salvati solo in memoria RAM venendo dunque persi allo spegnimento del computer.

Un malware dispone di tre possibili modalità per l'ottenimento della persistenza:

1 Modifica delle chiavi di registro

Il malware modifica una determinata chiave di registro nel registro Windows. In questo modo verrà eseguito all'avvio del sistema operativo.

2 Task pianificato

Il malware sfrutta la funzionalità dei task pianificati dal sistema operativo. Copiando il percorso del suo eseguibile verrà eseguito ad uno specificato istante e/o con una data frequenza.

3 Startup Folder

Il malware copia il suo eseguibile in una delle cartelle di startup. Questa può essere relativa al solo utente corrente o comune a tutti gli utenti.

PROCESS EXPLORER

Tool che permette di osservare tutti i processi in esecuzione su un sistema.

PROCMON (PROCESS MONITOR):

Tool avanzato di windows che permette di monitorare i thread attivi, le attività di rete, l'accesso ai file e le chiamate di sistema effettuate da un OS.

REGSHOT

Tool che permette di confrontare lo stato delle chiavi di registro di un sistema con le loro istantanee prima e dopo l'esecuzione di un malware.

SEZIONI

Le sezioni sono delle porzioni dell'eseguibile che accedono ad una determinata risorsa o ad una parte di essa. Servono a contenere tutti gli elementi necessari all'esecuzione.

Comunemente le sezioni si dividono in:

- a) **.text**: contiene le righe di codice che verranno eseguite;
- b) **.rdata**: contiene le librerie o funzioni che vengono importate ed esportate;
- c) **.data**: contiene le variabili globali, le quali devono essere accessibili da qualsiasi punto del codice;
- d) **.rsrc**: contiene le risorse utilizzate dall'eseguibile come ad esempio icone, immagini, menù e stringhe che non sono parte dell'eseguibile stesso;

VARIABILI LOCALI E GLOBALI

Una variabile, in informatica, è un contenitore di dati situato in una porzione di memoria destinata a contenere valori, suscettibili di modifica nel corso dell'esecuzione di un programma.

Può essere locale (se definita limitatamente ad una singola funzione o porzione di codice) o globale (se viene definita a monte dell'intero eseguibile e quindi utilizzabile da tutte le funzioni del programma).

WIRESHARK

Wireshark è un software di analisi del traffico di rete open-source. È ampiamente utilizzato per catturare, visualizzare e analizzare i dati che passano attraverso una rete informatica.

IL TEAM

LUIGI BENVENUTI

MARCO DE FALCO

DANIELE D'ESPOSITO

ALESSANDRO MARASCA

ANTHONY MIDEA



GRAZIE