

Malware Analysis

Alessandro Morabito--Antonio Spirin--Daniele Morabito--Ernesto Robles--Stefano Castiglioni-- Giorgio Trovesi--Giuseppe Pariota--Giacomo Caregnato

Nel seguente report, effettueremo l’analisi di un malware, denominato Malware_Build_Week_U3, individuandone gli elementi costituenti e le possibili funzioni, giustificando quest’ultime con gli elementi evidenziati nell’analisi stessa.

DAY1

Analisi strutturale del malware.

Innanzitutto, esaminiamo la struttura del malware per ricavarne le **sezioni** di cui è composto, come mostrate nell’immagine seguente.

property	value	value	value	value	
name	.text	.rdata	.data	.rsrc	
md5	6BB361AB84E6EA32F545B128...	23FDE5162E5B17A6E440A46...	E433B4C400EFC11A593220E...	9D561586EEB5ECDA6C3214C...	
entropy	6.225	3.770	0.601	4.154	
file-ratio (92.31%)	46.15 %	7.69 %	23.08 %	15.38 %	
raw-address	0x00001000	0x00007000	0x00008000	0x0000B000	
raw-size (49152 bytes)	0x00006000 (24576 bytes)	0x00001000 (4096 bytes)	0x00003000 (12288 bytes)	0x00002000 (8192 bytes)	
virtual-address	0x00401000	0x00407000	0x00408000	0x0040C000	
virtual-size (47372 bytes)	0x00005646 (22086 bytes)	0x000009AE (2478 bytes)	0x00003EA8 (16040 bytes)	0x00001A70 (6768 bytes)	
entry-point	0x00001487	-	-	-	
characteristics	0x60000020	0x40000040	0xC0000040	0x40000040	
writable	-	-	x	-	
executable	x	-	-	-	
shareable	-	-	-	-	
discardable	-	-	-	-	
initialized-data	-	x	x	x	
uninitialized-data	-	-	-	-	
unreadable	-	-	-	-	
self-modifying	-	-	-	-	
virtualized	-	-	-	-	
file	executable, offset: 0x00003249	n/a	n/a	executable, offset: 0x0000B070	

Di seguito una rapida spiegazione delle sezioni scoperte:

- **.text**: contiene il codice eseguibile del programma.
- **.data**: contiene le variabili globali del programma, definite dal programmatore. Questa sezione può essere modificata dal programma stesso durante l'esecuzione.
- **.rdata**: sezione di sola lettura, contiene le costanti e le stringhe a cui fa riferimento il programma.
- **.rsrc**: contiene le risorse utilizzate dal programma, quali ad esempio icone e ulteriori dati.

Esaminiamo poi le **librerie** importate dal malware, fornendone una rapida overview. L’analisi delle librerie risulta fondamentale in fase di indagine in quanto può fornire importanti indizi riguardo il funzionamento del malware stesso, indicando in particolare le funzioni che questo utilizza e le parti del sistema che va a modificare.

Malware_Build_Week_U3.exe						
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	File
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	0C

- **kernel32.dll**:è una libreria che contiene le funzioni principali per l’interazione con il sistema operativo come la gestione della memoria o la manipolazione dei file.
- **advapi32.dll**:questa libreria contiene le risorse per interagire con i servizi ed i registri del sistema operativo.

Per quanto riguarda le **funzioni** richiamate dal software malevolo, di seguito le più interessanti al fine di capire il suo funzionamento.

Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007664	00007664	00A3	FindResourceA

Al netto di librerie e funzioni prese in esame, possiamo immaginare che il malware si comporti come un **dropper**, caricando ed eventualmente eseguendo un ulteriore software contenuto al suo interno: sarà possibile confermare tale ipotesi solo dopo una più approfondita analisi statica e dinamica avanzata.

Parametri e variabili.

Attraverso l’analisi del codice assembly ricavato dal malware evidenziamo quali paramenti e variabili vengano rispettivamente passate e dichiarate all’interno della **funzione main()**, come mostrate nella prossima immagine.

```

; int __cdecl main(int argc,const char **argv,const char *envp)
_main proc near

hModule= dword ptr -11Ch
Data= byte ptr -118h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h

```

Notiamo che sono presenti **quattro variabili** e **tre parametri**, rispettivamente con offset degli indirizzi di memoria **negativo** e **positivo** rispetto quello del registro **ebp**.

DAY2

Analisi del malware tramite codice assembly

Esamineremo di seguito una specifica funzione presente nel codice del malware, visibile nell'immagine seguente espressa in formato assembly.

```
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
```

La funzione interessata è **RegCreateKeyExA** all'indirizzo di memoria 00401021 ed richiamata nello stack dal **call**; la funzione ha lo scopo di creare (o aprire, se già presente) la chiave di registro specificata dal parametro corrispondente della funzione stessa. I parametri che accetta le vengono infatti passati attraverso i **push** alle righe superiori. In particolare, quella che le viene passata col push all'indirizzo di memoria 00401017 è la **subkey**, in questo caso il valore della chiave che viene passata alla funzione stessa.

Un altro interessante costrutto è presente agli indirizzi 00401027 e 00401029; abbiamo infatti un **test** tra due argomenti uguali, **eax**: essendo **test** equiparabile all'operazione booleana **AND**, il risultato di un'operazione avente gli operandi uguali sarà uguale all'operando stesso. Inoltre, è importante ricordare che l'operazione **test** modifica solo il **registro EFLAG**. Guardando alla riga successiva, il salto condizionale **jz** è effettuato solo se il test precedente imposta la Zero Flag (**ZF = 1**), ciò a sua volta avviene solo, in questo caso, se **eax** è uguale a 0.

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

Di seguito, a scopo esemplificativo, il codice assembly esaminato viene "tradotto" in C.

```
If (eax == 0) { jump to short_loc 00401032 }
```

Tornando all'analisi del codice assembly, la successiva funzione chiamata sullo stack di particolare interesse è **RegSetValueExA**, mostrata nella seguente immagine.

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

Tra i vari parametri importati, possiamo notare **ValueName**, a cui è assegnato il valore **GinaDLL**; questa è la chiave di registro su cui lavorerà la funzione.

Considerando gli elementi analizzati finora, possiamo supporre che con questi ultimi passaggi il malware stia cercando di modificare le chiavi di registro per ottenere **persistenza** sul sistema o per "intaccare" il processo di **autenticazione** del sistema stesso.

DAY3

Analisi routine del malware

Procedendo con l’analisi, utilizziamo anche il software **OlllyDBG**, di cui è mostrato un estratto nella prossima immagine, per esaminare il malware in un ambiente controllato in cui è possibile interrompere il flusso del codice a piacimento.

004010BD	. 50	PUSH EAX	[ResourceType => "BINARY" Malware_.00408038 ResourceName => "TGAD" hModule FindResourceA
004010BE	. 8B0D 34804000	MOV ECX,DWORD PTR DS:[408034]	
004010C4	. 51	PUSH ECX	
004010C5	. 8B55 08	MOV 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	. 837D EC 00	CMP DWORD PTR SS:[EBP-14],0	

Tra gli elementi della routine presa in esame notiamo la funzione **FindResourceA** ed in particolare il parametro necessario **ResourceName**, a cui è assegnato il valore **TGAD**: questa è la porzione del malware a cui il codice accede per trovare un secondo codice malevolo da eseguire successivamente, probabilmente il **GinaDLL** visto in precedenza; considerando ciò e il continuo susseguirsi di chiamate di funzione **FindResource()**, **LoadResource()** e **SizeOfResource()** abbiamo una ulteriore conferma che l’eseguibile esaminato sia un **dropper**.

Tali informazioni, seppur non disponibili con analisi statica base in quanto questa non ci permette di indagare la sezione **.rsrc**, erano in parte visibili dal codice assembly di **IDA**, come mostrato nella seguente immagine, dove **lpName** prende come valore la stessa porzione **TGAD**.

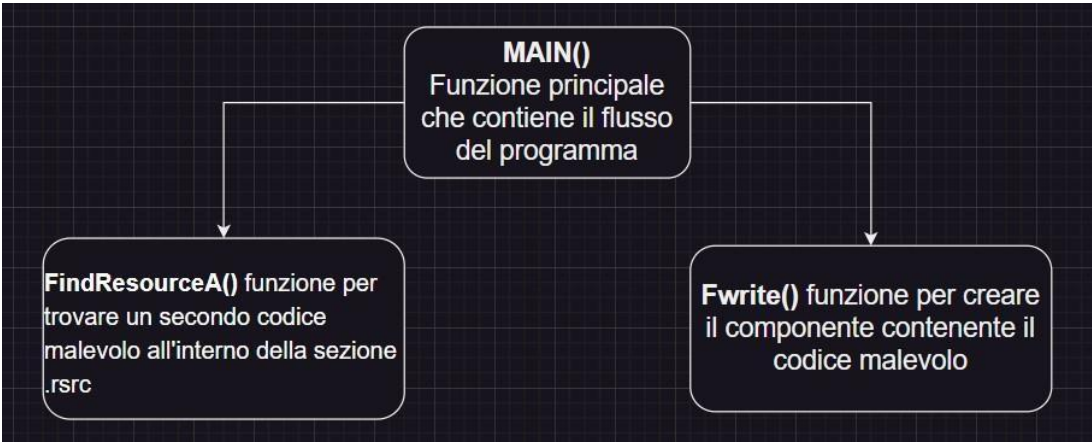
text:004010C5	mov	edx, [ebp+hModule]	
text:004010C8	push	edx	; hMod; LPCSTR lpName
text:004010C9	call	ds:FindResourceA	lpName dd offset aTgad ; DATA XREF: sub_401080+3ETr
text:004010CF	mov	[ebp+hResInfo], eax	; "TGAD"

Eseguendo il malware viene creato con la funzione **Fwrite()** nella sua stessa cartella il file **msgina32.dll**, sul quale si può effettuare una analisi statica.

```
push    offset aMsgina32_dll_0 ; "msgina32.dll"
call    _fopen

push    eax
call    _fwrite
```

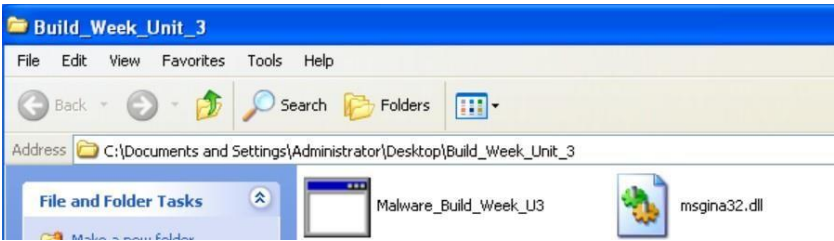
Ancora una volta, possiamo ipotizzare che il malware interagisca con il processo di **autenticazione** dell’utente, ma non è possibile giungere ad alcuna conclusione certa sul suo scopo limitandoci ad analizzare le informazioni disponibili sulle funzioni esportate e librerie importate. A scopo esemplificativo, di seguito è presente un diagramma di flusso che mostra i processi del malware fino al punto esaminato.



DAY4

Esame dei processi del malware

Come visto in precedenza, in seguito all’esecuzione del malware viene creato un file msgina32.dll, il quale è probabilmente una versione alterata della libreria presa di mira.



Per esaminare questo processo e, in generale, i processi generati dal malware, utilizziamo

Procmon. Notiamo immediatamente il processo con cui viene creata la nuova chiave di registro, ossia **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\Currentversion\Winlogon**, e il successivo, evidenziato nella seguente immagine, in cui le viene assegnato il valore **GinaDLL**.

1940	RegOpenKey	HKLM		SUCCESS	Desired Access: Maximum Allowed
1940	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Diagnostics		NAME NO...	Desired Access: Read
1940	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ntdll.dll		NAME NO...	Desired Access: Read
1940	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\kernel32.dll		NAME NO...	Desired Access: Read
1940	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon		SUCCESS	Desired Access: All Access
1940	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL		SUCCESS	Type: REG_SZ, Length: 520, Data:
1940	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon		SUCCESS	

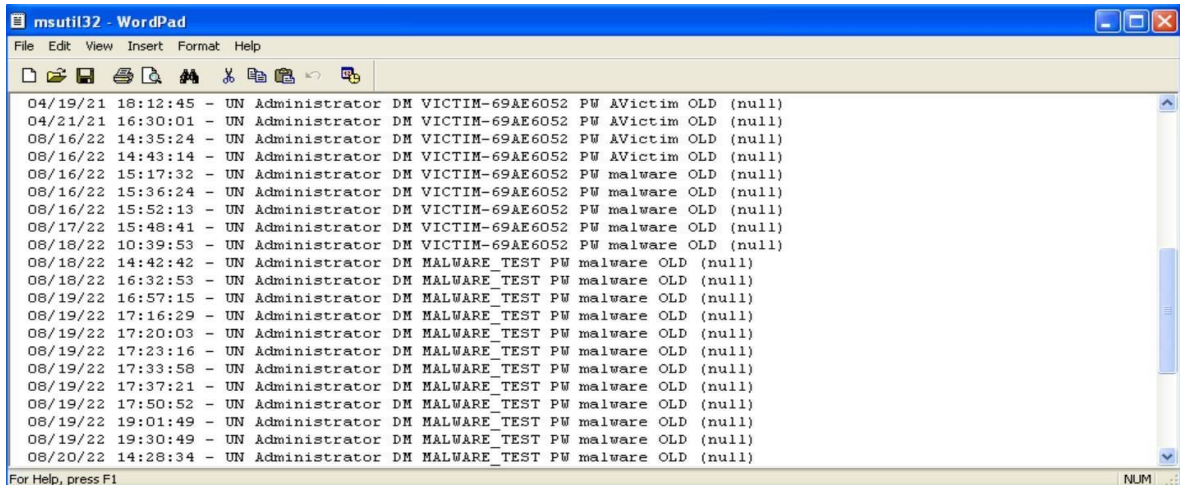
Vediamo infine la chiamata di sistema che ha creato il nuovo file nella cartella del malware.

Malware_Build_W...	1940	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS
Malware_Build_W...	1940	FileSystemControl	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS
Malware_Build_W...	1940	QueryOpen	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe.Local	NAME NO...
Malware_Build_W...	1940	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS
Malware_Build_W...	1940	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS
Malware_Build_W...	1940	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS
Malware_Build_W...	1940	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS
Malware_Build_W...	1940	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS
Malware_Build_W...	1940	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS
Malware_Build_W...	1940	SetEndOfFileInfo...	C:\WINDOWS\system32\config\software.LOG	SUCCESS

DAY5

Conclusioni

Al netto delle azioni del malware, deduciamo che in fase di autenticazione il sistema userà la versione **alterata** della **libreria gina.dll**, andando probabilmente a **rubare le credenziali** inserite dall'utente. Il malware, inoltre, crea un **file testuale** chiamato **msutil.sys** sul quale registra gli input dell'utente stesso: nella seguente immagine, "Administrator" è il nome dell'utente connessosi mentre "malware" è la sua password.



```
msutil32 - WordPad
File Edit View Insert Format Help

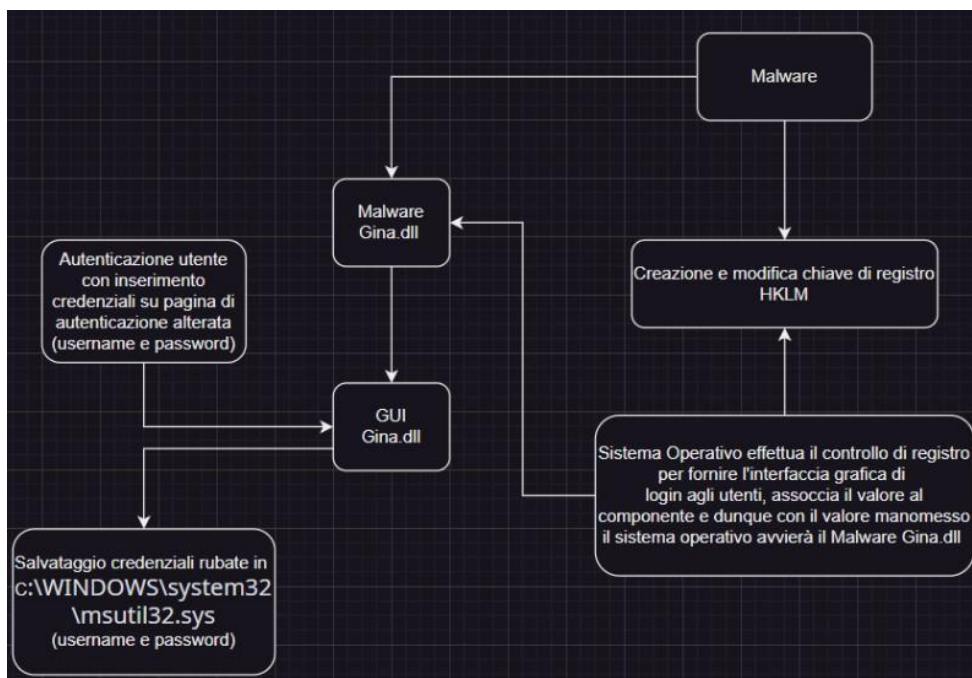
04/19/21 18:12:45 - UN Administrator DM VICTIM-69AE6052 PW AVictim OLD (null)
04/21/21 16:30:01 - UN Administrator DM VICTIM-69AE6052 PW AVictim OLD (null)
08/16/22 14:35:24 - UN Administrator DM VICTIM-69AE6052 PW AVictim OLD (null)
08/16/22 14:43:14 - UN Administrator DM VICTIM-69AE6052 PW AVictim 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)
08/19/22 17:20:03 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 17:23:16 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 17:33:58 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 17:37:21 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 17:50:52 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 19:01:49 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/19/22 19:30:49 - UN Administrator DM MALWARE_TEST PW malware OLD (null)
08/20/22 14:28:34 - UN Administrator DM MALWARE_TEST PW malware OLD (null)

For Help, press F1
```

Di seguito il frammento di codice assembly in cui si nota la creazione del documento di testo.

```
.text:1000158E      call     _vsnwprintf
.text:10001593      push    offset word_10003320 ; wchar_t *
.text:10001598      push    offset aMsutil32_sys ; "msutil32.sys"
.text:1000159D      call    _wfopen
.text:100015A2      mov     esi, eax
.text:100015A4      add     esp, 18h
.text:100015A7      test    esi, esi
```

In conclusione, mostriamo il **completo funzionamento del malware** in un diagramma di flusso di alto livello, rendendo evidenti tutti i passaggi che l'eseguibile effettua una volta avviato.



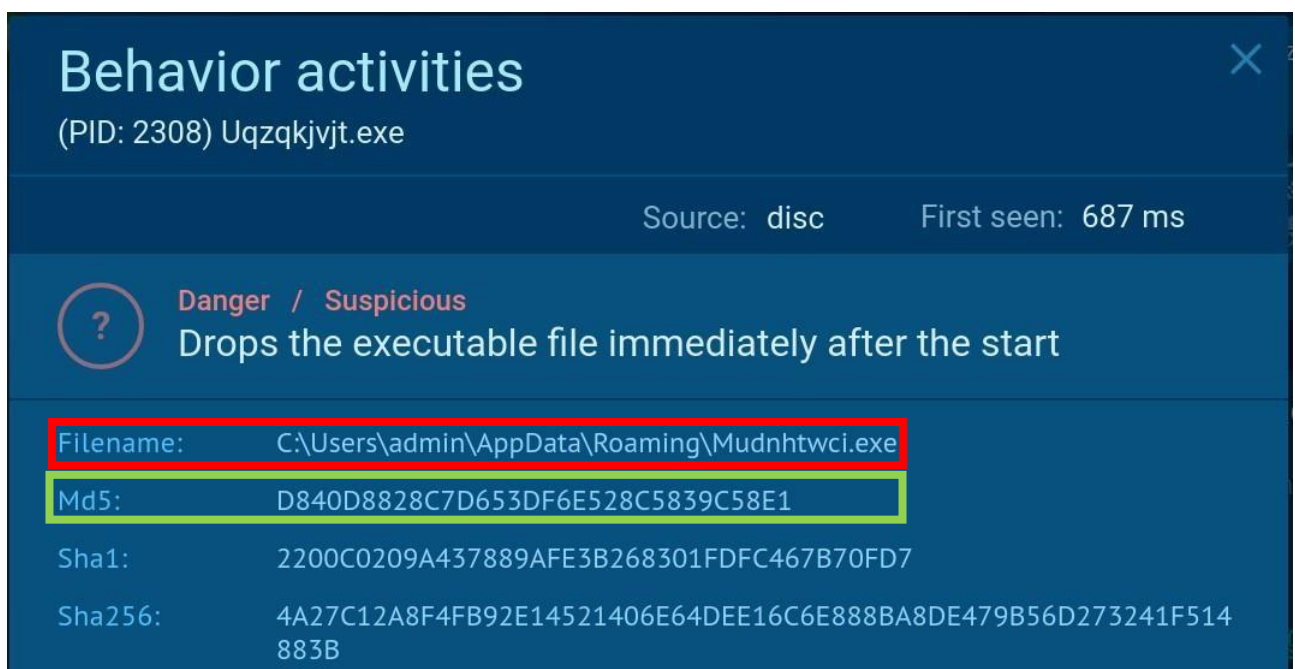
Bonus 1 – Agenttesla

Anyrun è un software di rilevamento, ricerca e monitoraggio per minacce informatiche, dotato inoltre di un sandbox online col quale effettuare ricerca su eventuali malware.

Nel seguente report analizzeremo un estratto dal programma Anyrun, nel quale vedremo l’alert riguardo al malware riconosciuto come **Agenttesla**.

Analisi

Anyrun riconosce tra i file scaricati sulla macchina che il file **uqzqkqvjt.exe** si comporta come un **dropper**, andando a creare un ulteriore eseguibile al path evidenziato in **rosso** nella seguente figura.



Il file creato è in realtà una copia di uqzqkqvjt.exe, in quanto i due hanno lo stesso **hash MD5**; nella precedente immagine evidenziato in **verde** l’hash dell’eseguibile creato.



Successivamente, un eseguibile **RegAsm.exe** viene riconosciuto da Anyrun come malevolo, in particolare viene associato allo **spyware AGENTTESLA**. Quest’ultimo è in grado di funzionare da keylogger, effettuare screenshot, rubare password ed inviare i dati così estratti ad un server CnC esterno. Nel caso qui esaminato, vediamo nel seguente screenshot alcuni esempi dei dati estratti da una serie di web browser., seguito da immagini che mostrano il percorso dei file estratti.

Danger 6

AGENTTESLA has been detected (YARA)

T1071 Application Layer Protocol (1)

Connects to the CnC server

T1555.003 Credentials from Web Browsers (1)

Steals credentials from Web Browsers

T1552.001 Credentials In Files (2)


Steals credentials from Web Browsers

Actions looks like stealing of personal data

T1518 Software Discovery (1)

Actions looks like stealing of personal data

Drops the executable file immediately after the start

 **Danger / Stealing**

Actions looks like stealing of personal data

T1552.001 Credentials In Files

T1518 Software Discovery

Operation:

CREATE

Device:

DISK_FILE_SYSTEM

Object:

UNKNOWN TYPE

Name:

C:\Users\admin\AppData\Local\Elements Browser\User Data

Status:


0xC000003A

Created:

SUPERSEDED

Access:

FILE_READ_ATTRIBUTES

 **Danger / Stealing**

Steals credentials from Web Browsers

T1555.003 Credentials from Web Browsers

T1552.001 Credentials In Files

Operation:

CREATE

Device:

DISK_FILE_SYSTEM

Object:

FILE

Name:

C:\Users\admin\AppData\Roaming\Mozilla\SeaMonkey\profiles.ini

Status:

0xC000003A

Created:

SUPERSEDED

Access:

READ_CONTROL, SYNCHRONIZE, FILE_READ_DATA, FILE_READ_EA, FILE_READ_ATTRIBUTES

Le informazioni rubate vengono quindi inviate ad un server localizzato in **Vietman**.

HTTP Requests		Connections		DNS Requests		Threats		Filter by PID, domain, name or ip			PCAP
Timeshift	Protocol	Rep	PID	Process name	CN	IP	Port	Domain	ASN	Traffic	
BEFORE	UDP	✓	4	System	?	192.168.100.255	137	–	–	↑ 994 b ↓	–
BEFORE	UDP	?	–	–	?	224.0.0.252	5355	–	–	↑ 48 b ↓	–
864 ms	UDP	✓	1956	svchost.exe	?	239.255.255.250	1900	–	–	↑ 399 b ↓	–
868 ms	UDP	✓	4	System	?	192.168.100.255	138	–	–	↑ 2.04 Kb ↓	–
1869 ms	UDP	?	324	svchost.exe	?	224.0.0.252	5355	–	–	↑ 48 b ↓	–
2871 ms	TCP	✗	792	RegAsm.exe	🇺🇸	64.185.227.156	443	api.ipify.org	WEBNX	↑ 173 b ↓	7 b
4975 ms	TCP	🔥	792	RegAsm.exe	🇻🇳	112.213.92.100	587	mail.asiaparadisehotel.com	SUPERDATA	↑ 897 b ↓	523 b

Per inviare le informazioni, il malware utilizza il protocollo **smtp**, in particolare l’indirizzo email utilizzato per l’invio e le sue credenziali sono i seguenti.

AgentTesla

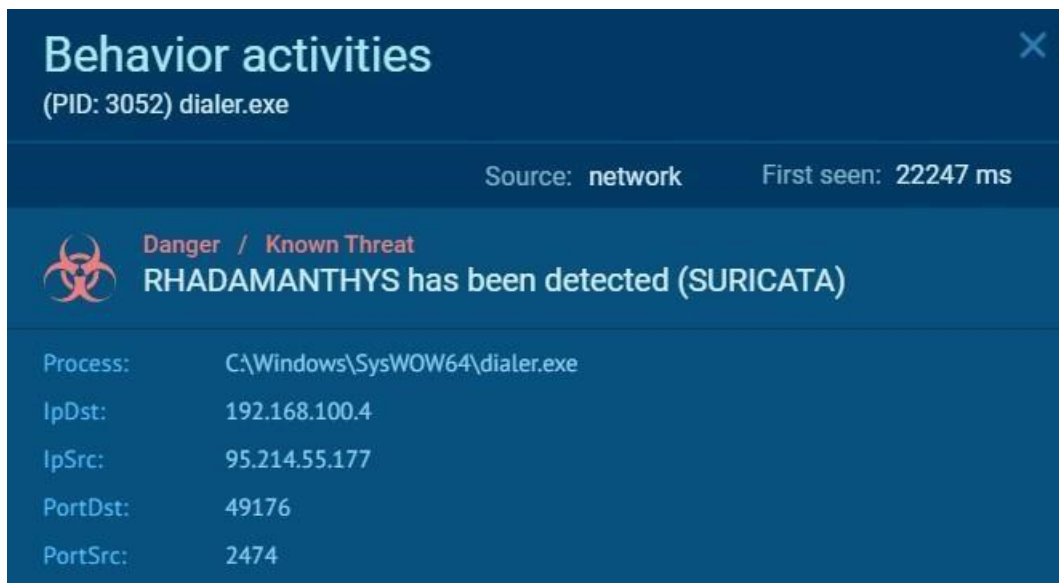
(PID) Process	(792) RegAsm.exe
Protocol	smtp
Host	mail.asiaparadisehotel.com
Port	587
Username	asia@asiaparadisehotel.com
Password	*b2ycDldex\$@

Bonus 2 – Rhadamanthys

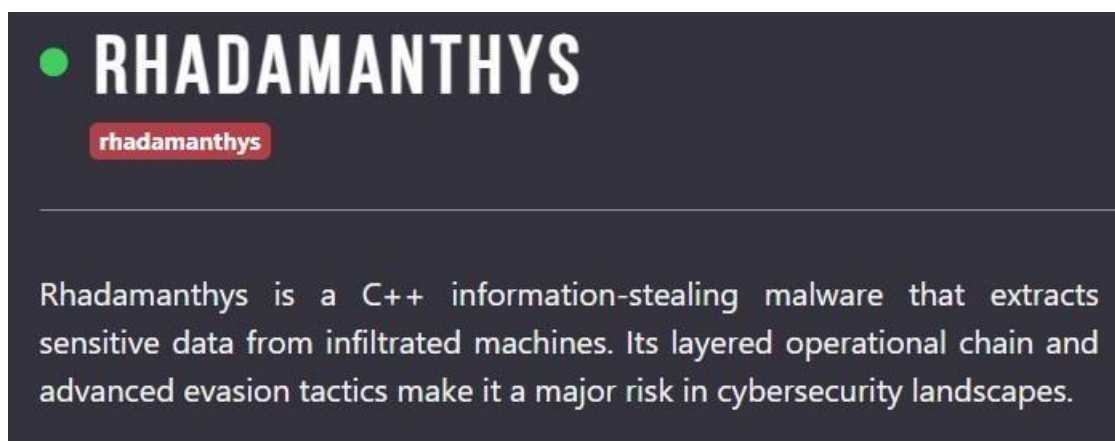
Nel seguente report, analizzeremo alcune schermate di Anyrun al fine di comprendere e spiegare quale possibile minaccia sia stata individuata dal software.

Analisi

Notiamo nella sequenza di schermate presentate da Anyrun che l'utente scarica da un repository Github un file **loader.exe**, il quale si comporta come un **downloader** scaricando a sua volta l'eseguibile **dialer.exe**. Quest'ultimo è un **trojan**, in quanto al suo interno contiene il file che Anyrun identifica come malware effettivo: **Rhadamanthys**.



Rhadamanthys è un malware **info-stealer** che prende di mira principalmente credenziali e dati finanziari salvati nel browser, account email, portafogli di criptovaluta oltre a hardware, sistema e software installati e IP delle vittime. Inoltre è organizzato secondo il modello **MaaS** (malware as a service), per cui i suoi autori lo “pubblicizzano” e ne mettono in vendita i servizi. (fonte: <https://any.run/malware-trends/rhadamanthys>)



Esaminando nel dettaglio l’alert, vediamo che Rhadamanthys agisce a livello delle connessioni internet, in particolare si collega all’IP **95.214.55.177:2474** dalla porta locale **49176**, tramite protocollo **TCP**; tale IP risulta essere situato in **Polonia**. Analizzando inoltre le richieste DNS, notiamo l’assenza della risoluzione dell’indirizzo IP di quest’ultima connessione: questa quindi non necessita di un dominio associato.

Behavior activities

(PID: 3052) dialer.exe

Source: networkFirst seen: 22442 ms

?

Warning / Unusual Activities
Connects to unusual port
T1571 Non-Standard Port

Process:

C:\Windows\SysWOW64\dialer.exe

IpDst:

95.214.55.177

PortDst:

2474

PortSrc:

49176

Protocol:

TCP

Nella seguente immagine, come evidenziato nel paragrafo precedente, l’ultima connessione in uscita dalla macchina vittima **non presenta la risoluzione dell’IP tramite DNS** al contrario delle altre. Notiamo inoltre che tale connessione viene fatta partire da **dialer.exe**.

Filter by PID, domain, name or ip											PCAP
HTTP Requests	0	Connections	15	DNS Requests	12	Threats	3				
Timeshift	Protocol	Rep	PID	Process name	CN	IP	Port	Domain	ASN	Traffic	
1315 ms	UDP	✓	1950	svchost.exe	?	239.255.255.250	1900	-	-	↑ 399 b ↓ -	
1353 ms	TCP	✓	2852	chrome.exe	🇺🇸	140.82.121.3	443	github.com	GITHUB	↑ 1.18 Kb ↓ 5.92 Kb	
1371 ms	TCP	✓	2852	chrome.exe	🇺🇸	185.199.109.133	443	raw.githubusercontent.com	FASTLY	↑ 1.19 Kb ↓ 454 Kb	
1700 ms	UDP	?	2820	chrome.exe	?	224.0.0.251	5353	-	-	↑ 120 b ↓ -	
2705 ms	UDP	?	324	svchost.exe	?	224.0.0.252	5355	-	-	↑ 48 b ↓ -	
2706 ms	TCP	✓	2852	chrome.exe	🇺🇸	142.250.186.78	443	sb-ssl.google.com	GOOGLE	↑ 2.32 Kb ↓ 8.65 Kb	
11909 ms	TCP	✓	2852	chrome.exe	🇺🇸	142.250.185.78	443	safebrowsing.google.com	GOOGLE	↑ 1.62 Kb ↓ 8.07 Kb	
11912 ms	TCP	✓	2852	chrome.exe	🇺🇸	142.250.186.170	443	optimizationguide-pa.googleapis.com	GOOGLE	↑ 1.57 Kb ↓ 7.04 Kb	
11922 ms	TCP	✓	2852	chrome.exe	🇺🇸	142.250.186.170	443	optimizationguide-pa.googleapis.com	GOOGLE	↑ 1.86 Kb ↓ 8.49 Mb	
22116 ms	TCP	🔥	3052	dialer.exe	🇵🇱	95.214.55.177	2474	-	Meverywhere sp. z o.o.	↑ 992 b ↓ 2.22 Mb	

BONUS 3 – CALCOLATRICE INNOVATIVA

ANALISI STATICA BASICA:

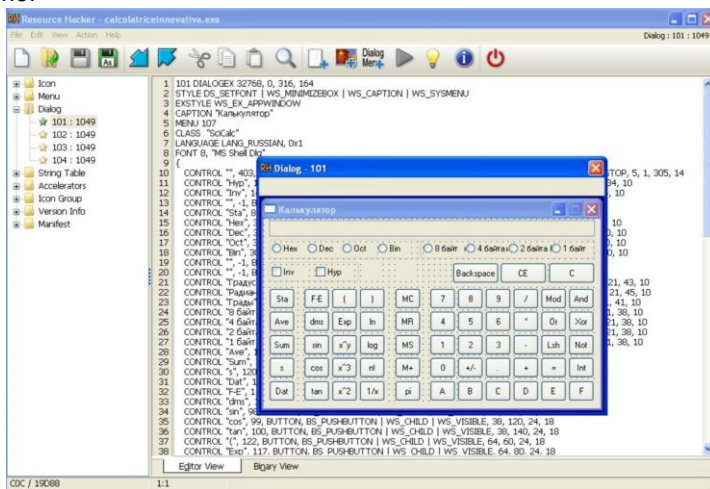
Analizzando l'eseguibile con *CFFExplorer* possiamo vedere che le librerie importate comprendono quelle relative alla GUI, modifiche chiavi di registro e operazioni di base del sistema operativo.

calcolatriceinnovativa.exe						
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
SHELL32.dll	1	00012CA8	FFFFFFFF	FFFFFFFF	00012E42	0000109C
msvcrt.dll	26	00012DC8	FFFFFFFF	FFFFFFFF	00012F60	000011BC
ADVAPI32.dll	3	00012C0C	FFFFFFFF	FFFFFFFF	00012FFC	00001000
KERNEL32.dll	30	00012C2C	FFFFFFFF	FFFFFFFF	000131D4	00001020
GDI32.dll	3	00012C1C	FFFFFFFF	FFFFFFFF	0001320C	00001010
USER32.dll	69	00012CB0	FFFFFFFF	FFFFFFFF	000136A4	000010A4

Importando il file su *Resource Hacker* si trovano sia immagini della calcolatrice, che ci mostrano menu con lettere dell'alfabeto cirillico, così come icone e altri file di contorno.

Utilizzando *PEStudio* possiamo identificare la lingua utilizzata come Russo, il che va in conflitto con il nome originale del file "CALC.exe". Possiamo inoltre individuare il nome del produttore che sembra essere Microsoft. Tuttavia, provando ad avviare il presunto malware, vediamo che Windows non riconosce il pubblicatore, e chiede conferma per l'esecuzione.

property	value
md5	D729FA69235A30CF3D60DCE1C6855
sha1	20C07443F9F5F3606B02C783FE9EE1C464A
sha256	20C0759513A63EC8102FA60E9714056734F4891C30D0CE43756205CB6201
file-type	executable
date	imgp1...
language	Russian
code-page	Unicode UTF-16, little endian
CompanyName	Корпорация Майкрософт
FileDescription	Калькулятор для Windows
FileVersion	5.1.2600.0 (xpsp1.010817-1148)
InternalName	CALC
LegalCopyright	© Корпорация Майкрософт. Все права защищены.
OriginalFilename	CALC.EXE
ProductName	Операционная система Microsoft® Windows®
ProductVersion	5.1.2600.0



ANALISI DINAMICA BASICA:

Utilizzano *ProcMon* per controllare le azioni eseguite dal Malware ci accorgiamo che questo termina con codice di errore 1073741819 (codice error 0xC0000005, segmentation fault).

2019-12-18 21:26	calcolatorecinnova.exe	252	Process Start	SUCCESS	Parent PID: 1231, Command Line: "C:\Documents and Settings\Administrator\My Documents\Download&Executed\calcolatorecinnova.exe"
2019-12-18 21:26	calcolatorecinnova.exe	252	Thread Create	SUCCESS	Thread ID: 2040
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x10000000, Image Size: 0x1000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x16000000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x18000000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x19000000, Image Size: 0x17000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x17400000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x17600000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x17800000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x17900000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x17A00000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x17B00000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x17C00000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x17D00000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x17E00000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Load Image	SUCCESS	Image Base: 0x17F00000, Image Size: 0x4000
2019-12-18 21:26	calcolatorecinnova.exe	252	Thread Exit	SUCCESS	Thread ID: 2040, User Name: 0x00000000, Kernel Time: 0x002500

Analizzando modifiche al filesystem, modifiche ai registri, connessioni di rete non vediamo niente di rilevante. Non è possibile quindi effettuare una analisi dinamica del malware fornito.

È stato quindi eseguito il file su un sistema con Windows 7 a 64-bit si è riusciti ad effettuare una analisi dinamica basica.

Si notano le connessioni effettuate verso un dispositivo sulla stessa rete (192.168.1.80:4444).

Time ...	Process Name	PID	Operation	Path	Result	Detail
9.46:3...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.46:4...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.46:5...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.47:0...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.47:1...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.47:2...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.47:3...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.47:4...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.47:5...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.48:0...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.48:2...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...
9.48:2...	calcolatriceinn...	1420	TCP Reconnect	John-PC:49160 -> 192.168.1.80:4444	SUCCESS	Length: 0; sequen...

ANALISI STATICA AVANZATA:

Aperto il file con *IDA Pro* dopo aver verificato con *exeinfope* e *DiE* che esso non era compresso, possiamo vedere come la maggior parte delle chiamate di funzioni avvengano sulla base dei valori salvati in registro. Per questo siamo impossibilitati dall'individuare quali sono quelle effettivamente chiamate costruendone una sequenza. Anche analizzando le singole funzioni non riusciamo a intuire il funzionamento del malware. Il problema più grande però probabilmente può essere ricondotto al fatto che l'eseguibile è destinato a un'architettura a 64-bit, il che richiede un'analisi con un disassemblatore dedicato.

VIRUSTOTAL:

Caricando il malware su *virustotal* viene generalmente riconosciuto come Trojan, mentre *ESET* lo identifica come backdoor. Questo tipo di malware richiede una connessione alla rete, ma tra le librerie importate in maniera dinamica non troviamo niente di relativo. Notiamo la presenza della funzione *LoadLibrary* all'interno della libreria *KERNEL32.dll* che permette l'importazione in runtime di librerie. Analizzando il processo con *ProcMon* vediamo che fa riferimento a *mswsock.dll*.

Module	Address	Size	Path
calcolatriceinnov...	0x1000000	0x1f000	C:\Users\John\Desktop
mswsock.dll	0x74140000	0x3c000	C:\Windows\SysWOW64

BONUS 4 – AmicoNerd ANALISI

STATICA BASICA:

Analizzando inizialmente il file “amiconerd.exe” con *exeinfope* si vede che questo è estraibile utilizzando il software *de4dot*.

Utilizzando quindi CFFExplorer per analizzare il file spaccettato si nota che la funzione *_CorExeMain* della libreria *mscorlib.dll* sia l'unica ad essere importata in maniera dinamica.

szAnsi	(nFunctions)
mscorlib.dll	1

szAnsi
_CorExeMain

Questo permette all'eseguibile di fare riferimento a librerie .NET. Possiamo quindi supporre che sia stato scritto con un linguaggio come *C#, F#* o *Visual Basic*, cosa verificata con *PEStudio*.

signature [Microsoft Visual C# v7.0 / Basic .NET](#)

Con lo stesso software possiamo inoltre trovare il nome originale del file, "AutoPico.exe". Una veloce ricerca ci mostra che questo è un software che permette l'attivazione illegale di software Microsoft sfruttando il *Key Management System (KMS)* dello stesso Microsoft e simulandone il server relativo.

language	neutral
code-page	Unicode UTF-16, little endian
Comments	Portable
CompanyName	@ByELDI
FileDescription	AutoPico
FileVersion	15.0.0.7
InternalName	AutoPico.exe
LegalCopyright	n/a
LegalTrademarks	n/a
OriginalFilename	AutoPico.exe
ProductName	AutoPico
ProductVersion	15.0.0.7
Assembly Version	15.0.0.7

ANALISI DINAMICA BASICA:

L'analisi dinamica basica ci mostra che viene creato un secondo processo *DW20.exe*, ma non riusciamo a individuare nessuna connessione verso siti potenzialmente malevoli, facendo riferimento sia alle richieste *DNS* sia a indirizzi *IP* a cui ci si vuole connettere, aiutandoci rispettivamente con *ApateDNS* e *Wireshark*.

Time	Domain Requested	DNS Return...
14:10:47	yutao318525.3322.org	FOUND
14:11:00	www.google.com	FOUND
14:11:00	3.pool.ntp.org	FOUND
14:11:28	yutao318525.3322.org	FOUND

ANALISI STATICA AVANZATA:

Non avendo visto nessun comportando malevolo si è proceduto con l'analisi statica avanzata per vedere se è possibile identificare delle funzioni potenzialmente malevole, che potrebbero non essere state eseguite per via della configurazione della macchina virtuale.

Sono state così trovate delle funzioni facenti riferimento al *Key Management System*, così come funzioni proprie di *KMSPico*.

```
aKmsclient      db 'KMSClient',0
aKmsclientprodu db 'KMSClientProduct',0
aKmsserver      db 'KMSServer',0
aKmsnetworkclie db 'KMSNetworkClient',0
aKmsserversetti db 'KMSServerSettings',0
aIkmsserversett db 'IKMSServerSettings',0
aAutopico_kms   db 'AutoPico.KMS',0
```


VIRUSTOTAL:

Abbiamo infine caricato il file su *virustotal* per avere una conferma dei risultati. L'esito è stato che *ESET-NOD32* individua l'eseguibile come *HackTool*, ovvero software che permette l'attivazione illegale programmi e sistemi operativi Microsoft. Per fare questo viene effettuata una connessione verso un server *KMS* che può essere potenzialmente malevolo.

KMS activates computers on a local network, eliminating the need for individual computers to connect to Microsoft. To do this, KMS uses a client-server topology. KMS client computers can locate KMS host computers by using Domain Name System (DNS) or a static configuration. KMS clients contact the KMS host by using remote procedure call (RPC). KMS can be hosted on computers that are running the Windows Vista, Windows 7, Windows Server 2003, Windows Server 2008, or Windows Server 2008 R2 operating systems.

AhnLab-V3	⚠ HackTool/Win.AutoKMS.C948312	ALYac	⚠ Application.Hacktool.KMSActivator.AQ
Antiy-AVL	⚠ RiskWare[NetTool]/Win64.RPCHook	Arcabit	⚠ Application.Hacktool.KMSActivator.AQ
Avast	⚠ Win32:MiscX-gen [PUP]	AVG	⚠ Win32:MiscX-gen [PUP]
BitDefender	⚠ Application.Hacktool.KMSActivator.AQ	BitDefenderTheta	⚠ Gen:NN.ZemsiIF.36792.Tm1@a8vJERd
ClamAV	⚠ Win.Tool.Kmsactivator-9811695-0	CrowdStrike Falcon	⚠ Win/grayware_confidence_100% (W)
Cybereason	⚠ Malicious.9168f0	Cylance	⚠ Unsafe
Cynet	⚠ Malicious (score: 100)	DeepInstinct	⚠ MALICIOUS
Elastic	⚠ Malicious (high Confidence)	Emsisoft	⚠ Application.HackTool (A)
eScan	⚠ Application.Hacktool.KMSActivator.AQ	ESET-NOD32	⚠ A Variant Of MSIL/HackTool.IdleKMS.E Po...
Fortinet	⚠ Riskware/RPCHook	GData	⚠ MSIL.Application.HackKMS.X