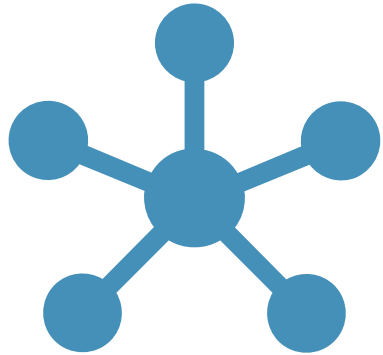


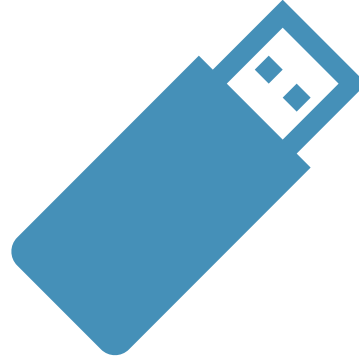


MALWARE ANALYSIS: AZIENDA THETA

ANALISI COMPLETA DEL MALWARE « MALWARE_BUILD_WEEK_U3 »



ISOLARE LA
MACCHINA



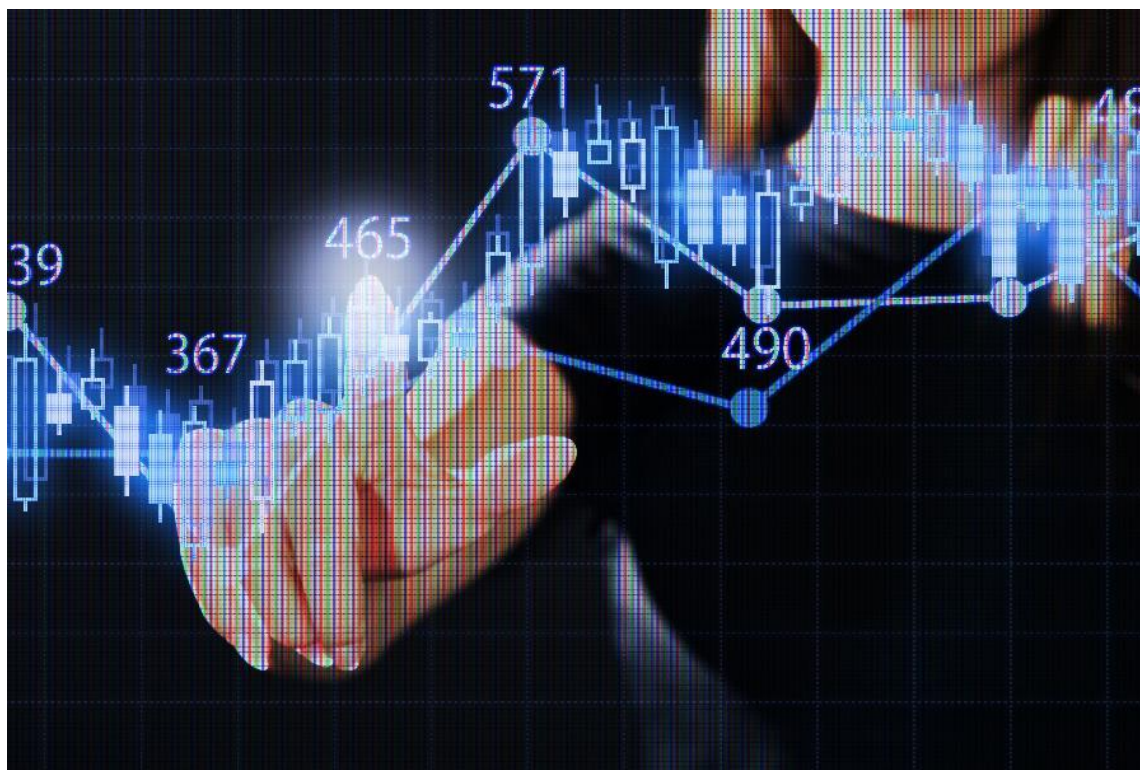
DISATTIVARE
CONTROLLER USB



EVITARE CARTELLE
CONDIVISE

REQUISITI TECNICI

PANORAMA GENERALE



E' stata eseguita un ' analisi completa del malware « Malware_Build_Week_u3» , ovvero sia non eseguendo il malware tramite l'analisi statica attraverso tool come IDA, CFF explorer; sia tramite l'esecuzione del malware ovvero con l'analisi dinamica utilizzando strumenti come Process Monitor, OllyDBG. Una volta terminate, sono state riportate le conclusioni con le relative evidenze

TEAM :

Andrea Bonarrigo, Marta Masoni, Giovanni Cossu, Pasquale Iannella, Giuseppe Prezzo, Marco Levi, Gabriele Pagana, Dehans Gjerka

SOMMARIO

Giorno 1: Funzione «main» , sezioni e librerie malware

Giorno 2: Analisi da indirizzo di memoria 00401017 ad indirizzo 00401047:

Giorno 3: Analisi tra indirizzo di memoria 00401080 ad indirizzo 00401128

Giorno 4: Analisi dinamica tramite ProcessMonitor

Giorno 5: Conclusioni

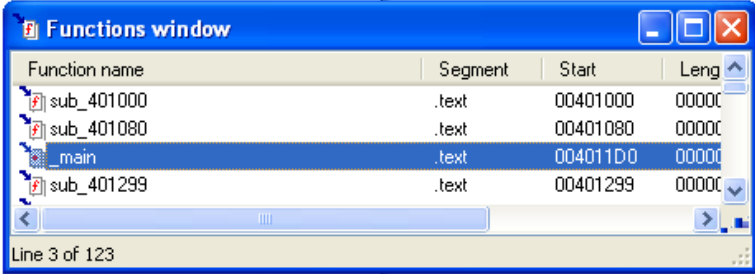
GIORNO 1: FUNZIONE MAIN, SEZIONI E LIBRERIE MALWARE.

1) PARAMETRI E VARIABILI DELLA FUNZIONE MAIN

```
; Attributes: bp-based frame
; 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

push    ebp
mov     ebp, esp
sub     esp, 11Ch
push    ebx
push    esi
push    edi
mov     [ebp+var_4], 0
push    0 ; lpModuleName
```



Function name	Segment	Start	Length
sub_401000	.text	00401000	00000000
sub_401080	.text	00401080	00000000
_main	.text	004011D0	00000000
sub_401299	.text	00401299	00000000

È stato aperto il malware tramite il tool IDA che ci permette di avere un'analisi statica grafica e testuale del codice malevolo. Utilizziamo la finestra «functions» per trovare l'indirizzo di memoria associato alla funzione Main(), mostrato qui di seguito:

GIORNO 1: FUNZIONE MAIN, SEZIONI E LIBRERIE MALWARE.

Possiamo individuare i **parametri** passati alla funzione e le **variabili** dichiarate nella stessa funzione:

```
hModule= dword ptr -11Ch  
Data= byte ptr -118h
```

```
-----  
var_8= dword ptr -8  
var_4= dword ptr -4
```



Le **variabili** sono state individuate poiché presentano offset negativo rispetto al puntatore EBP; in totale sono 4:

- **hModule;**
- **Data;**
- **Var 8;**
- **Var 4**

```
argv= dword ptr 8  
argv= dword ptr 0Ch  
envp= dword ptr 10h
```



I **parametri** individuati sono in totale 3 ed a differenza delle variabili, essi presentano offset positivo rispetto al puntatore EBP.

GIORNO 1: FUNZIONE MAIN, SEZIONI E LIBRERIE MALWARE.

L'header del file eseguibile ci fornisce molte informazioni in merito al malware, tra cui le sezioni di cui si compone il software, le librerie e le funzioni importate ed esportate dal malware.

Nello specifico, **le sezioni** di cui si compone il malware da noi analizzato sono le seguenti:

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	00003EA8	00008000	00003000	00008000	00000000	00000000	0000	0000	C0000040
.rsrc	00001A70	0000C000	00002000	00008000	00000000	00000000	0000	0000	40000040

- SEZIONE.TEXT: contiene le istruzioni che la CPU eseguirà una volta che il software viene avviato.;
- SEZIONE .RDATA: include le informazioni sulle librerie e le funzioni importate ed esportate dall'eseguibile;
- SEZIONE .DATA: contiene i dati e le variabili globali del programma eseguibile, quindi accessibili da qualsiasi funzione;
- SEZIONE .RSRC: include le risorse utilizzate dall'eseguibile, come icone, immagini,, ecc che non sono parte dell'eseguibile.

GIORNO 1: FUNZIONE MAIN, SEZIONI E LIBRERIE MALWARE.

Le librerie importate dal malware sono invece le seguenti:

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



CONSIDERAZIONI GENERALI

RegSetValueExA
RegCreateKeyExA
...

ADVAPI32
ADVAPI32
KERNEL32

KERNEL32.dll: contiene le funzioni principali per interagire con il sistema operativo (manipolazione file, gestione memoria , ecc);
ADVAPI32.dll: contiene le funzioni per interagire con i servizi ed i registri del sistema operativo Microsoft.

La libreria ADVAPI32.dll importa in totale 2 funzioni che vengono utilizzate sia per scrivere o modificare il valore di una chiave del registro di sistema esistente; sia per per creare una nuova sottochiave del registro di sistema o per aprire una sottochiave esistente

GIORNO I: FUNZIONE MAIN, SEZIONI E LIBRERIE MALWARE.

Module Name	Imports
0000769E	N/A
szAnsi	(nFunctions)
KERNEL32.dll	51

OFTs	FTs (IAT)	Hint	Name
00007538	00007010	00007644	00007646
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
00007604	00007604	001B	CloseHandle

La libreria KERNEL32.dll importa in totale 51 funzioni che in generale gestiscono i processi ,la memoria , eventuali errori, i file ed informazioni (come credenziali) del sistema attaccato

```
@!@.8!@.TGAD....  
BINARY..RI..Gina  
DLL.SWFTWARE\Mic  
rosoft\Windows.N  
T\CurrentVersion  
\Winlogon...DR..  
msgina32.dll....  
wb..msgina32.dl  
l.....  
!*@.!.Xq@.Hq@.  
@@.....@@.!!..  
.....!.....
```

Dall'analisi statica effettuata, possiamo dedurre in generale che il malware modifica/crea una chiave di registro Windows in modo da ottenere eventualmente la persistenza o per eseguire altre azioni malevole; inoltre tramite la libreria KERNEL32.dll utilizza varie funzioni per manipolare i processi e la memoria. Analizzando le varie sezioni, interessante è la sezione «.data» (mostrato a lato) : si nota che il malware va presumibilmente a modificare la chiave di registro `\HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon` e «msgina32.» ; di conseguenza, possiamo ipotizzare che il malware modifichi l'interfaccia grafica che viene visualizzata all'utente durante il processo di autenticazione e quindi accesso al sistema

GIORNO 2: ANALISI DA INDIRIZZO DI MEMORIA 00401017 AD INDIRIZZO 00401047

Facendo riferimento all' **indirizzo di memoria 00401021**, mostrato qui di seguito:

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

Notiamo la chiamata di funzione «**RegCreateKeyExA**» che abbiamo già visto fa parte della libreria ADVAPI32.dll. Ha come scopo la creazione di una nuova chiave di registro nei sistemi Windows oppure ne consente l'apertura se la chiave è già esistente.

I **parametri della funzione** appena descritta vengono passati attraverso un push sullo stack di funzione. I parametri sono hKey, lpSubKey, Reserved, lpClass, dwOptions, samDesired, lpSecurityAttributes, phkResult, lpdwDisposition. La stringa «SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon» viene passata come SubKey e 80000002h viene utilizzato come valore della chiave di handle predefinita (hKey)

```
push     eax                ; phkResult
push     0                  ; lpSecurityAttributes
push     0F003Fh            ; samDesired
push     0                  ; dwOptions
push     0                  ; lpClass
push     0                  ; Reserved
push     offset SubKey      ; "SOFTWARE\Microsoft\Windows
push     80000002h          ; hKey
```

GIORNO 2: ANALISI DA INDIRIZZO DI MEMORIA 00401017 AD INDIRIZZO 00401047

Alla **locazione di memoria 00401017** viene passato il parametro **SubKey**. Il parametro è costituito da un puntatore ad una stringa, che rappresenta la registry subkey da creare o aprire all'interno della key specificata dall'handle hkey. Nel nostro caso la SubKey è data dalla stringa «SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon».

```
* .text:00401015      push    0           ; Reserved
* .text:00401017      push    offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
* .text:0040101F      push    80000000h    ; hKey
```

Considerando le istruzioni comprese tra gli **indirizzi di memoria 00401027 e 00401029**, possiamo notare che esse rappresentano un salto condizionale. Infatti, l'**istruzione test** effettua un confronto bitwise tra gli operatori, senza modificarne i valori. Quindi vengono modificate solo le flag a seguito del confronto. L'**istruzione successiva (jz)** salta all'indirizzo 00401032 se la Zero Flag è attiva, quindi se gli operandi sono uguali a seguito del test.

```
test    eax, eax
jz      short loc_401032
```

GIORNO 2: ANALISI DA INDIRIZZO DI MEMORIA 00401017 AD INDIRIZZO 00401047

Facendo riferimento al codice in assembly compreso tra l'indirizzo di memoria 00401027 e 00401029 , viene riportata di seguito la traduzione in codice C :

CODICE IN ASSEMBLY

```
push    0                ; Reserved
push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
push    80000002h        ; hKey
call    ds:RegCreateKeyExA
test    eax, eax
jz      short loc_401032
```



```
loc_401032:
mov     ecx, [ebp+cbData]
push    ecx              ; cbData
mov     edx, [ebp+lpData]
push    edx              ; lpData
push    1                ; dwType
push    0                ; Reserved
push    offset ValueName ; "GinaDLI
mov     eax, [ebp+hObject]
push    eax              ; hKey
call    ds:RegSetValueExA
test    eax, eax
jz      short loc_401062
```



TERMINA
PROGRAMMNA

CODICE IN C

```
#include <windows.h>

...

void subroutine() {

    DWORD codiceErrore;
    HKEY nuovaKey;

    codiceErrore = RegCreateKeyExA(HKEY_LOCAL_MACHINE, SOFTWARE\\Microsoft\\Windows NT\\
    CurrentVersion\\Winlogon, 0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, &
    nuovaKey, NULL);

    if (codiceErrore == ERROR_SUCCESS) {

        RegSetValueExA();
    }
    else {
        return ;
    }
}
```

GIORNO 2: ANALISI DA INDIRIZZO DI MEMORIA 00401017 AD INDIRIZZO 00401047

Questa subroutine controlla il messaggio che restituisce la funzione RegCreate. Infatti se esso assume il valore uguale a 0 allora chiama la funzione RegSetValue; in caso contrario termina il programma.



```
#include <windows.h>

...

void subroutine() {

    DWORD codiceErrore;
    HKEY nuovaKey;

    codiceErrore = RegCreateKeyExA(HKEY_LOCAL_MACHINE, SOFTWARE\\Microsoft\\Windows NT\\
    CurrentVersion\\Winlogon, 0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, &
    nuovaKey, NULL);

    if (codiceErrore == ERROR_SUCCESS) {

        RegSetValueExA();
    }
    else {
        return ;
    }
}
```


GIORNO 2: ANALISI DA INDIRIZZO DI MEMORIA 00401017 AD INDIRIZZO 00401047

All'indirizzo **00401047** viene chiamata la funzione
RegSetValueExA

La funzione si occupa di impostare il valore per una determinata registry subkey ed il valore del parametro **ValueName** che è **GinaDLL**.

→ **call ds:RegSetValueExA**

→ **push offset ValueName ; "GinaDLL"**


CONCLUSIONI

In questa sezione il malware tenta di creare/aprire una chiave di registro: la key "**HKEY_LOCAL_MACHINE**", la subkey "**SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon**" ed la libreria "GinaDLL". Si può dedurre quindi che il malware sfrutta la funzionalità **winlogon** per eseguire **GinaDLL** per recuperare le credenziali dell'utente ed ottenere accesso al sistema. Gina è l'acronimo di «Graphical Identification and Authentication» ed è un componente del sistema operativo Microsoft Windows che gestisce la funzionalità di login degli utenti. Quando un utente accede al sistema, il processo di login utilizza la Gina DLL per presentare l'interfaccia grafica di autenticazione all'utente, come la finestra di login con il nome utente e la password. La Gina DLL è anche responsabile della verifica delle credenziali di accesso dell'utente e dell'autorizzazione dell'accesso al sistema.

GIORNO 3: ANALISI ROUTINE TRA INDIRIZZO DI MEMORIA 00401080 AD INDIRIZZO 00401128

Tramite CFF explorer ricerchiamo il valore del parametro
“**ResourceName**” passato alla funzione FindResourceA() , ovvero
«**TGAD**».

E' possibile identificarlo anche tramite il tool IDA sostituendo il
valore di “ResourceName” con “lpName”



```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | @!@.8!@.TGAD....  
00 54 47 41 44 00 00 00 00 00 00 00 00 00 00 00 | BINARY..RI..Gina  
00 52 49 0A 00 47 69 6E 61 00 00 00 00 00 00 00 | DLL.SFTWARE\Mic  
54 57 41 52 45 5C 4D 69 63 00 00 00 00 00 00 00 | rosoft\Windows.N  
57 69 6E 64 6F 77 73 20 4E 00 00 00 00 00 00 00 | T\CurrentVersion  
6E 74 56 65 72 73 69 6F 6E 00 00 00 00 00 00 00 |  
  
* .data:00408034 ; LPCSTR lpName  
* .data:00408034 lpName dd offset aTgad  
* .data:00408034 aTgad db 'TGAD',0
```

CODICE ASSEMBLY : DA «00401080» A «00401128»

Facendo riferimento al codice illustrato nella prossima slide, è determinante evidenziare le chiamate di funzione che
effettua il Malware in questa sezione di codice perché sono caratteristiche distintive dei virus **DROPPER**

E' un programma malevolo che contiene al suo interno un malware che viene salvato sul
disco del pc attaccato una volta che il programma viene avviato.

Detto ciò, caratteristiche strutturali importanti del dropper sono 2:

**MALWARE NELLA SEZIONE
«.RSS/ RSRC»**



APIs UTILIZZATE

GIORNO 3: ANALISI ROUTINE TRA INDIRIZZO DI MEMORIA 00401080 AD INDIRIZZO 00401128

APIs UTILIZZATE

Queste chiamate API sono utilizzate per accedere alle risorse all'interno di un file eseguibile. Si potrebbero utilizzare queste chiamate per caricare ed eseguire codice malevolo o per nascondere. Si utilizza la chiamata "VirtualAlloc" per allocare dello spazio di memoria per eseguire il proprio codice in modalità stealth. Il malware utilizza la chiamata "LockResource" per bloccare le risorse in memoria in modo che non possano essere modificate o rilevate dai programmi antivirus.

Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	0288	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA

```
loc_401088:      mov     eax, lpType      ; CODE XREF loc_4010DF:
push     eax
mov     ecx, lpName
push     ecx
mov     edx, [ebp+hModule]
push     edx
call    ds:FindResourceA
mov     [ebp+hResInfo], eax
cmp     [ebp+hResInfo], 0
jnz     short loc_4010DF
xor     eax, eax
jmp     loc_4011BF

loc_4010DF:      mov     eax, [ebp+hResInfo]
push     eax
mov     ecx, [ebp+hModule]
push     ecx
call    ds:LoadResource
mov     [ebp+hResData], eax
cmp     [ebp+hResData], 0
jnz     short loc_4010FB
jmp     loc_4011A5

loc_4010FB:      mov     edx, [ebp+hResData]
push     edx
call    ds:LockResource
mov     [ebp+var_8], eax
cmp     [ebp+var_8], 0
jnz     short loc_401113
jmp     loc_4011A5

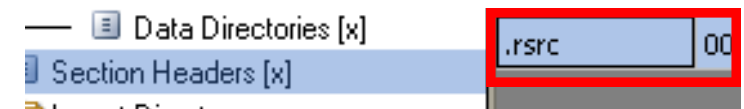
loc_401113:      mov     eax, [ebp+hResInfo]
push     eax
mov     ecx, [ebp+hModule]
push     ecx
call    ds:SizeofResource
mov     [ebp+dwSize], eax
cmp     [ebp+dwSize], 0
ja      short loc_40112C
jmp     short loc_4011A5
```

GIORNO 3: ANALISI ROUTINE TRA INDIRIZZO DI MEMORIA 00401080 AD INDIRIZZO 00401128

MALWARE NELLA SEZIONE «.RSS/ RSRC»

Il malware incluso nel dropper è contenuto nella **sezione : «.rsrc» dell'header**

	ASCII
%d...\M.S.G.i.n.	WlxWkstaLockedSA
a...ShellShutdow	S...G.i.n.a.D.L.
nDialog.WlxActiv	L...S.o.f.t.w.a.
ateUserShell...	r.e.\.M.i.c.r.o.
WlxDisconnectNot	s.o.f.t.\.W.i.n.
ify.WlxDisplayLo	d.o.w.s...N.T.\.
ckedNotice..WlxD	C.u.r.r.e.n.t.V.
isplaySASNotice.	e.r.s.i.o.n.\.W.
WlxDisplayStatus	i.n.l.o.g.o.n...
Message.WlxGetCo	M.S.G.i.n.a...d.
nsoleSwitchCrede	l.l....U.N...%
ntials..WlxGetSt	s...D.M...%s...
atusMessage.WlxI	P.W...%s...O.L.
nitialize.█.WlxI	D...%s...WlxL
sLockOk.WlxIsLog	oggedOutSAS....
offOk...WlxLogge	E.r.r.o.r.C.o.d.
dOnSAS..WlxLogof	e...%d...E.r.r.
f...WlxNegotiate	o.r.M.e.s.s.a.g.
...WlxNetworkPr	e...%s.....
oviderLoad..WlxR	%s...%s...-...
econnectNotify..	%s....m.s.u.t.
WlxRemoveStatusM	i.l.3.2...s.y.s.
essage..WlxScreeI.....
nSaverNotify...	a.....
WlxShutdown.WlxS
tartApplication.
WlxWkstaLockedSA



Come si legge, il malware richiama diverse funzioni che sono contenute all'interno dell'header «winwix.h». Esso è utilizzato nella programmazione di applicazioni Windows e contiene definizioni di funzioni, costanti e strutture dati utilizzate dal servizio di autenticazione di Window, **Winlogon**. Il file winwix.h è utilizzato principalmente **per scrivere un provider di autenticazione personalizzato** per Windows consentendo quindi di personalizzare il processo di autenticazione ad esempio aggiungendo nuove modalità di autenticazione o di verifica delle credenziali.

**FUNZIONI UTILIZZATE
DAL MALWARE
DELL'HEADER
«WINWLH.H»**



GIORNO 3: ANALISI ROUTINE TRA INDIRIZZO DI MEMORIA 00401080 AD INDIRIZZO 00401128

- **WlxActivateUserShell** : Attiva il programma della shell utente
- **WlxDisconnectNotify**: Winlogon chiama questa funzione quando una sessione di rete di Servizi terminal è disconnessa.
- **WlxDisplayLockedNotice**: Consente all'GINA di visualizzare informazioni sul blocco,
- **WlxDisplaySASNotice**:
- **WlxDisplayStatusMessage**
- **WlxGetConsoleSwitchCredentials**: Winlogon chiama questa funzione per leggere le credenziali dell'utente attualmente connesso per trasferirle in modo trasparente a una sessione di destinazione.
- **WlxGetStatusMessage**
- **WlxInitialize** : inizializza il provider di autenticazione personalizzato
- Winlogon
- **WlxIsLockOk**: determina se il blocco dello schermo è consentito per la sessione dell'utente.
- **WlxIsLogoffOk**: determina se il logoff dell'utente è consentito per la sessione dell'utente
- **WlxLoggedOnSAS**: Winlogon chiama questa funzione quando riceve un evento sas mentre l'utente è connesso e la workstation non è bloccata.
- **WlxLoggedOutSAS**: gestisce gli eventi Secure Attention Sequence (SAS), come l'accesso remoto e la disconnessione dell'utente.
- **WlxLogoff**:
- **WlxNegotiate**: negozia le opzioni di autenticazione e le credenziali dell'utente
- **WlxNetworkProviderLoad**: per raccogliere informazioni di identificazione e autenticazione valide.
- **WlxReconnectNotify**
- **WlxRemoveStatusMessage**
- **WlxScreenSaverNotify**
- **WlxShutdown**: subito prima dell'arresto, consentendo all'GINA di eseguire qualsiasi attività di arresto
- **WlxStartApplication**: chiamata quando il sistema richiede l'avvio di un'applicazione nel contesto dell'utente.
- **WlxWkstaLockedSAS**: funzione chiamata quando riceve una sequenza di attenzione sicura e la workstation è bloccata.

GIORNO 3: ANALISI ROUTINE TRA INDIRIZZO DI MEMORIA 00401080 AD INDIRIZZO 00401128

Queste funzioni sono utilizzate per implementare un provider di autenticazione personalizzato per Windows, che può personalizzare il processo di autenticazione e di gestione della sessione utente.



Vengono sfruttate dal dropper tramite l'installazione ed esecuzione di un software/provider di autenticazione malevolo per registrare le credenziali dell'utente o per inviarle a un server esterno.

```
%s...%s...-...  
%s.....msut.  
i.l.3.2...s.y.s.  
.....  
a.....  
.....  
.....
```

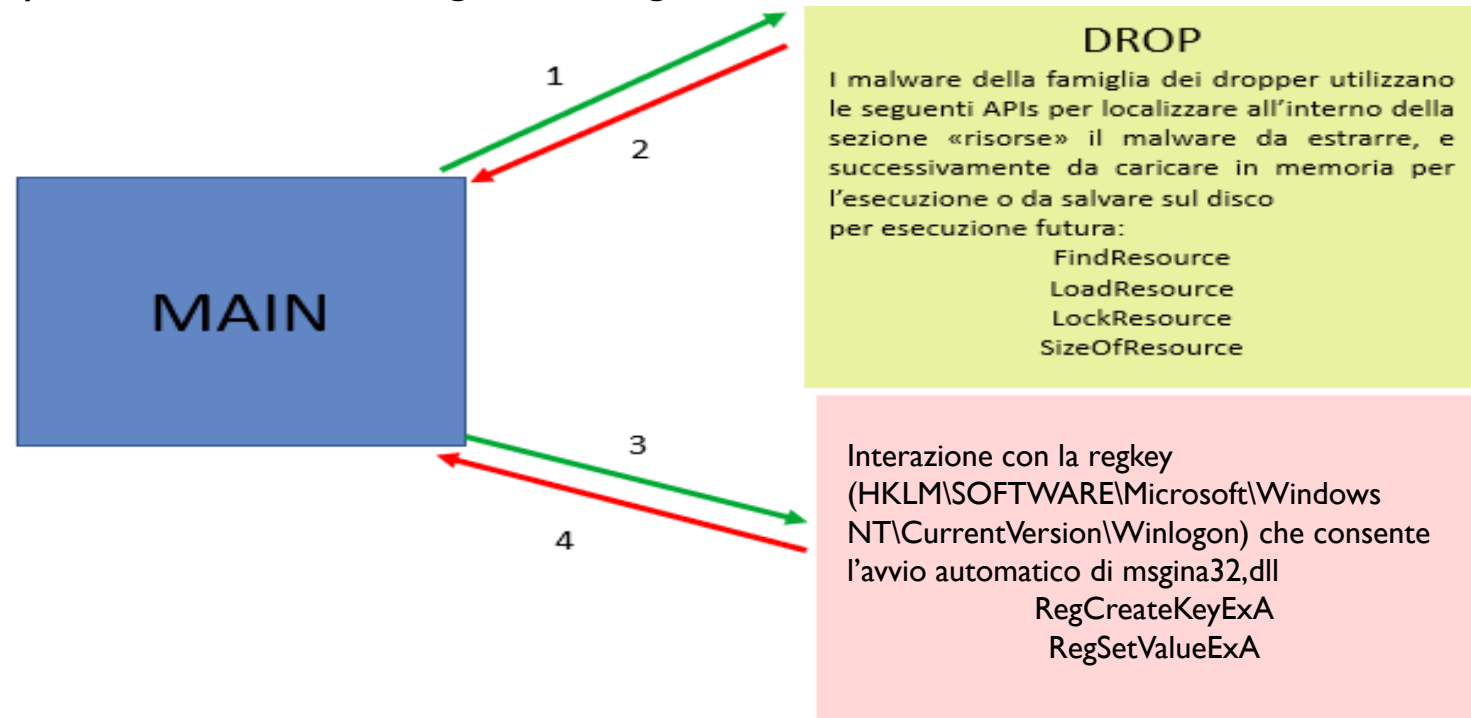
Nel nostro caso specifico, vengono salvate nel file «msutil32.sys», che analizzeremo in seguito



GIORNO 3: ANALISI ROUTINE TRA INDIRIZZO DI MEMORIA 00401080 AD INDIRIZZO 00401128

CONCLUSIONI

E' stata utilizzata **l'analisi statica basica** del malware, **senza quindi eseguirlo**, per avere un'idea quasi chiara del suo comportamento. Essa si può riassumere nel seguente diagramma:



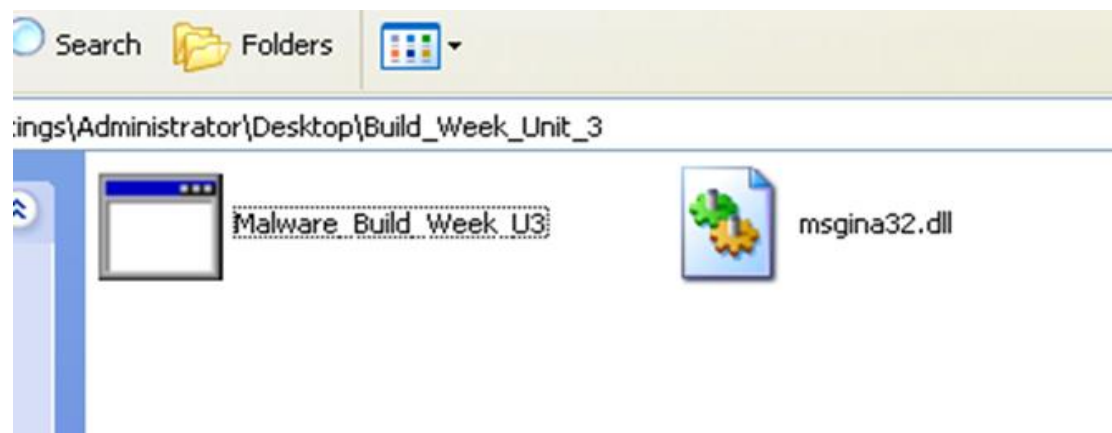
GIORNO 4: ANALISI DINAMICA TRAMITE PROCESS MONITOR

E' stato avviato Process Monitor ed una volta resettati i filtri preimpostati all'avvio del programma è stato eseguito il malware.

Come si vede dalla figura a destra, il malware utilizza la chiamata di sistema «**CreateFile**» per modificare il contenuto della cartella dove è presente l'eseguibile, creando un file chiamato «**msgina32.dll**»

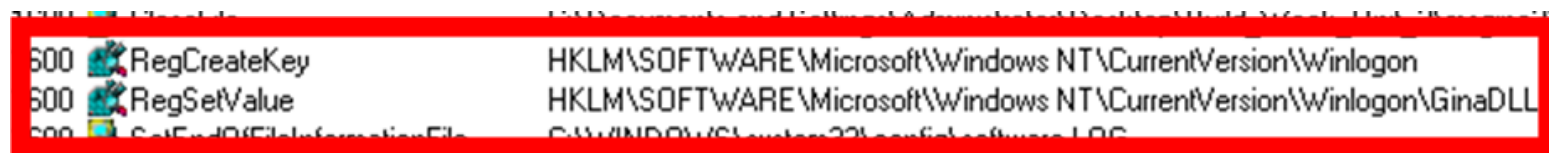
1600 CreateFile
1600 CreateFile
1600 CreateFile

C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3



GIORNO 4: ANALISI DINAMICA TRAMITE PROCESS MONITOR

Continuando ad analizzare i risultati di Process Monitor, si nota che il malware crea una chiave di registro alla quale viene associata il valore di GinaDLL.

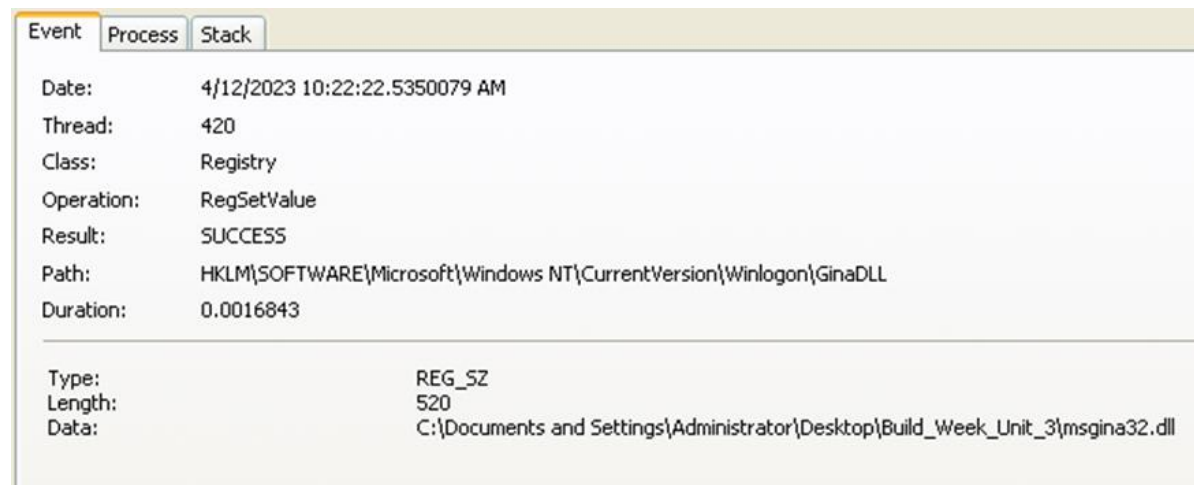


600	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
600	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
600	SetEndOfFile	C:\Windows\Globalization\Sorting\SortDefault.nls

CONCLUSIONI

Dalle informazioni che abbiamo raccolto dall'analisi statica e dinamica, possiamo dire che il malware in questione è un dropper.

Molto importante viene fuori che il malware dropa msgina32.dll, ed ottiene la persistenza creando la chiave di registro. Infatti come abbiamo già visto, l'eseguibile originale contiene la DLL nella sezione delle risorse che rilasciata nella directory di lavoro come msgina32.dll.



Event	Process	Stack
Date:	4/12/2023 10:22:22.5350079 AM	
Thread:	420	
Class:	Registry	
Operation:	RegSetValue	
Result:	SUCCESS	
Path:	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	
Duration:	0.0016843	
Type:	REG_SZ	
Length:	520	
Data:	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	

GIORNO 5: CONCLUSIONI

In conclusione a tutte le analisi e considerazioni fatte, possiamo affermare che il malware analizzato è un DROPPER che tenta di sostituire la libreria GINA.dll con una creata ad hoc , in modo da poter intercettare le credenziali d'accesso al pc attaccato.

I PUNTI SALIENTI

1. La libreria "GINA.DLL" (Graphical Identification and Authentication Dynamic Link Library) è un componente di sistema del sistema operativo Windows che gestisce l'interfaccia di login.



Essa viene sostituita con una versione compromessa chiamata msgina32.dll

Operation:	RegSetValue
Result:	SUCCESS
Path:	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
Duration:	0.0016843

Type:	REG_SZ
Length:	520
Data:	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll

GIORNO 5: CONCLUSIONI

2 . La libreria modificata esporta diverse funzioni tutte con prefisso **Wlx** che come abbiamo visto viene utilizzato per identificare le funzioni, le costanti e le strutture di dati specifiche dell'API di WinLogon.

Come il malware quindi ruba le credenziali?

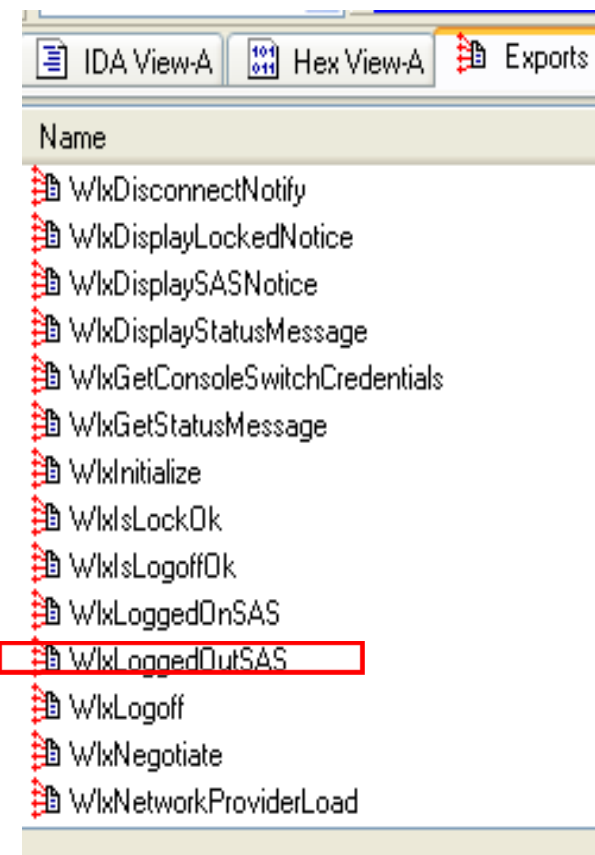


Implementa **WlxLoggedOutSAS**, che copia il nome utente, le password vecchie e nuove e il dominio

Tutte le altre funzioni Wlx vengono passate senza essere modificate.



```
; void __stdcall WlxShutdown(PVOID pWlxContext,DWORD ShutdownType
public WlxShutdown
WlxShutdown proc near
push    offset aWlxshutdown ; "WlxShutdown"
call    sub_10001000
jmp     eax
WlxShutdown endp
```



GIORNO 5: CONCLUSIONI

WlxLoggedOutSAS inoltra le informazioni come le altre ma richiama la funzione «??@YAPAXI@Z» che copia l'username e le password vecchie nuove.



```
push esi
push edi
push offset aWlxloggedoutsa ; "WlxLoggedOutSAS"
call sub_10001000
push 64h
mov edi, eax
call ??2@YAPAXI@Z ; operator new(uint)
```

```
push eax
push offset aUnSDmSPwS0ldS ; "UN %s DM %s PW %s OLD %s"
push 0 ; dwMessageId
call sub_10001570
add esp, 18h

; CODE XREF: WlxLoggedOutSAS+47fj
; WlxLoggedOutSAS+4d1j

mov eax, edi
pop edi
pop esi
retn 20h
endp
```

DOVE VENGONO SALVATE LE CREDENZIALI?

```
push esi
push eax ; va_list
push ecx ; wchar_t *
push 800h ; size_t
push edx ; wchar_t *
call _vsnwprintf
push offset word_10003320 ; wchar_t *
push offset aMsutil32_sys ; "msutil32.sys"
call _wfopen
mov esi, eax
add esp, 18h
test esi, esi
jz loc_1000164F
```

Scrive l'utente, il dominio, la password e la vecchia password nel file di testo
C:\Windows\System32\msutil32.sys tramite la chiamata di funzione «_vsnwprintf»

è un file di testo normale senza codifica, che memorizza le credenziali con timestamp.



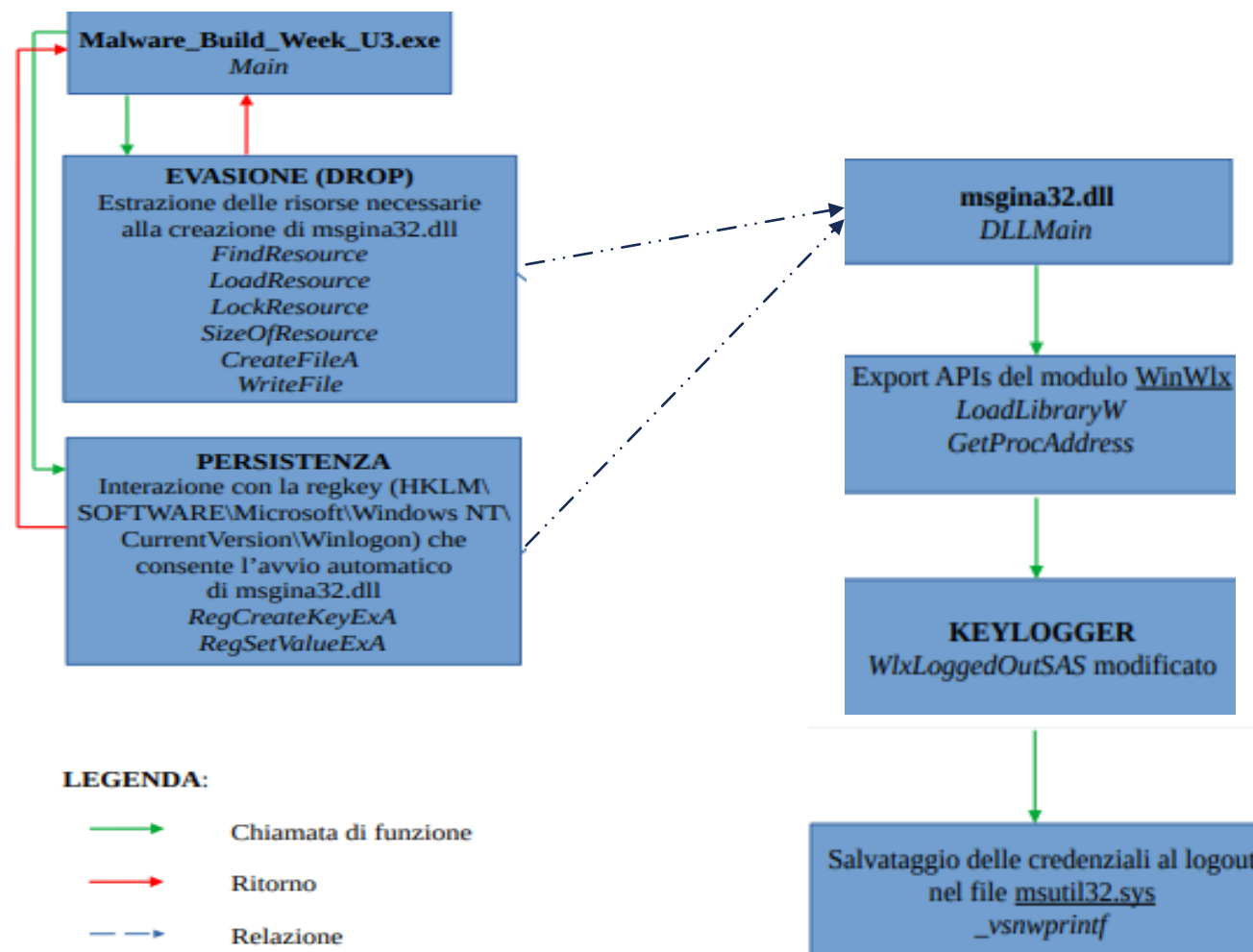
```
04/12/23 11:20:07 - UN Administrator DM MALWARE_TEST PW malware OLD {null}
```

GIORNO 5: CONCLUSIONI

Qui alla destra, si può vedere un grafico riassuntivo delle funzionalità del malware capendone quindi il suo scopo: **accedere potenzialmente alle informazioni e alle risorse dell'account attaccato senza il consenso o l'autorizzazione dell'utente**



L'attaccante in questo modo potrebbe essere in grado di acquisire il controllo completo del sistema e delle sue risorse, compromettendo così la sicurezza e la privacy dell'utente





GRAZIE