

MARCH 2024

Samuel Capoti
Stefan Ion Ungureanu
Alessio Golfetto
Kristiano Kamenica
Pierre Lobrillo
Michelle Cusinatti

BUILDING

WEEK III



Funzioni Main

Per l'analisi del malware in questione, abbiamo impiegato IDA Pro, un software disassembler che esamina il malware e restituisce il suo codice in linguaggio assembly (nel nostro caso, assembly dell'architettura x86). Il codice disassemblato viene visualizzato tramite una barra colorata; ciò che ci interessa è il colore blu, che rappresenta il codice del malware. Una volta completate le operazioni da parte di IDA, sullo schermo comparirà il punto di ingresso dell'applicazione, cioè il punto in cui il programma inizia ad eseguire il suo codice. Questo può essere visualizzato attraverso lo screenshot qui di seguito:

```
; int __cdecl main(int a
main proc near

hModule= dword ptr -11Ch
Data= byte ptr -118h
var_117= byte ptr -117h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
```

PARAMETRI E VARIABILI

La funzione main funge da punto di ingresso e, attraverso un'analisi, possiamo identificare i parametri della funzione e le variabili dichiarate al suo interno. Di seguito, le abbiamo elencate:

Parametri:

Abbiamo individuato i seguenti parametri:
argc, argv, envp.

Variabili:

Le variabili identificate sono: hModule, Data,
var_8, var_4.



DAY 1

Sezioni

Attraverso l'ausilio di CFF Explorer, abbiamo individuato quattro sezioni:

Malware_Build_Week_U3.exe									
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
000001D8	000001E0	000001E4	000001E8	000001EC	000001F0	000001F4	000001F8	000001FA	000001FC
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	0000B000	00000000	00000000	0000	0000	40000040

.text:

Questa sezione contiene il codice eseguibile del programma, ovvero le istruzioni in Assembly che vengono eseguite dal processore durante l'esecuzione del programma.

.rdata:

In questa sezione sono presenti dati di sola lettura (read-only data) che rimangono costanti e non subiscono modifiche durante l'esecuzione del programma.

.data:

A differenza della sezione precedente, questa area consente la modifica dei dati durante l'esecuzione. Spesso, include variabili globali e altri valori che possono essere modificati dinamicamente.

.rsrc:

Quest'ultima sezione ospita immagini, icone, stringhe e altri dati relativi all'interfaccia utente e alle funzionalità del programma.



DAY 1

LIBRERIE

Attraverso l'analisi con il tool CFF Explorer, abbiamo identificato due librerie:

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

KERNEL32.dll:

Questa libreria comprende le funzioni fondamentali per interagire con il sistema operativo, come la gestione della memoria e la manipolazione dei file.

ADVAPI32.dll:

Questa libreria contiene le funzioni necessarie per l'interazione con il registro di sistema del sistema operativo.



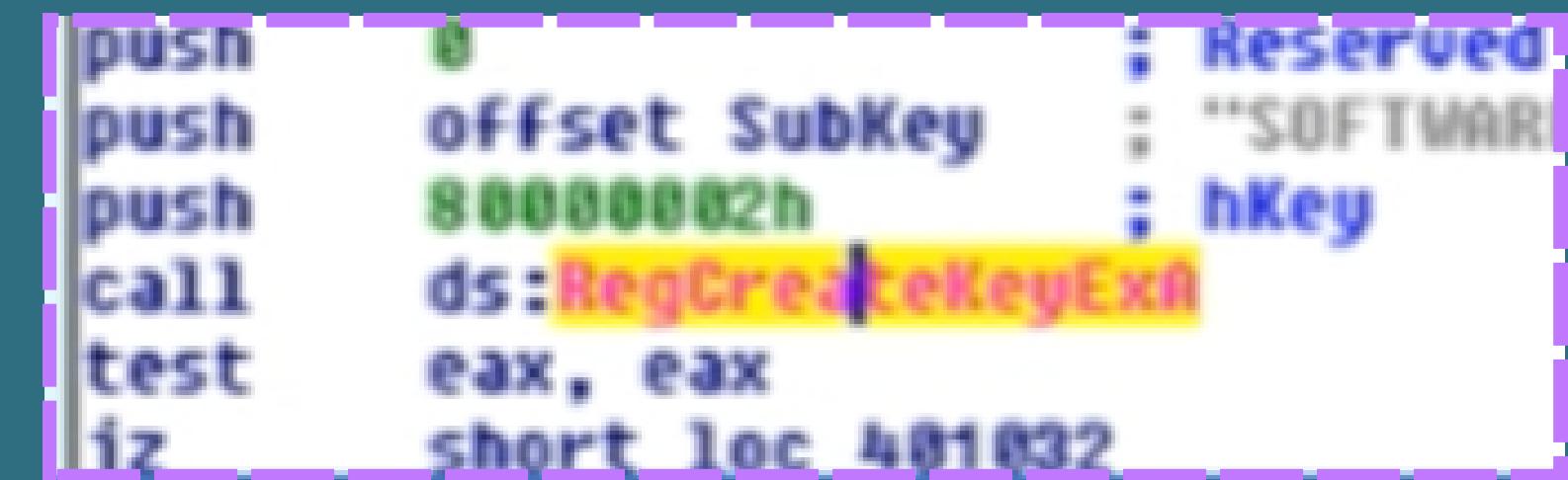
FUNZIONI DELLE LIBRERIE

- Tra le funzioni utilizzate dalle librerie, quelle degne di nota per KERNEL32.dll sono visibili nello screenshot sopra. È possibile ipotizzare che il malware agisca come un Dropper, che carica una risorsa malevola una volta presente sul computer target.
- Per la libreria ADVAPI32.dll, abbiamo individuato due funzioni che consentono la creazione e la configurazione di una chiave nel registro di sistema. È probabile che ciò sia finalizzato a ottenere la persistenza necessaria per il malware.

OFTs	FTs (IAT)	Hint	Name
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

SPIEGAZIONE FUNZIONE

All'indirizzo di memoria 00401021 è presente la funzione RegCreateKeyExA. Questa funzione ha il compito di creare una specifica chiave di registro; se già presente, viene aperta. I parametri vengono passati attraverso una serie di istruzioni push.



```
push    0          ; Reserved
push    offset SubKey ; hKey
push    $0000002h   ; hKey
call    ds:RegCreateKeyExA
test   eax, eax
jz     short loc_401032
```



SCOPO DELLA FUNZIONE

L'oggetto rappresentato dal parametro all'indirizzo di memoria 00401017 è la stringa "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon", che corrisponde al percorso della chiave di registro di sistema che il programma sta cercando di creare utilizzando la funzione "RegCreateKeyExA". Il codice coinvolge la chiamata alla funzione "RegCreateKeyExA", una funzione dell'API di Windows utilizzata per creare o aprire una chiave di registro di sistema. Quindi, questa stringa rappresenta il nome della chiave di registro stesso.



PARAMETRI

Le istruzioni comprese tra 00401027 e 00401029 fanno riferimento ai codici mnemonici test e jZ.

TEST:

Questa istruzione viene utilizzata per eseguire un'operazione logica tra due operandi, tramite un confronto bit a bit. Il registro EAX viene confrontato con se stesso attraverso un'operazione di AND. Il risultato non viene memorizzato, ma gli effetti dell'operazione sono riscontrabili sui flag dell'architettura (EFLAGS). Il flag ZF verrà impostato a 1 se il risultato della comparazione restituirà zero; altrimenti, ZF avrà valore 0 se il risultato della comparazione sarà diverso da zero.

JZ:

L'istruzione jZ fa parte della famiglia dei jmp e in questo caso assume il valore di un jump condizionale. La condizione necessaria per attuare il salto è rappresentata dal valore della ZF ("Jump if Zero"); il salto verrà effettuato se la ZF restituirà 1 nella comparazione precedente.



CODICE IN C

```
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 ; "GinaDLL"  
mov    eax, [ebp+hObject]  
push   eax          ; hKey  
call   ds:RegSetValueExA  
test   eax, eax  
jz    short loc_401062
```

Sì, puoi tradurre le istruzioni 00401027 e 00401029 in linguaggio C come segue:

```
if (eax == 0) {  
    // Salto alla locazione 401032  
} else {  
    // Proseguimento del codice  
}
```

Questa traduzione riflette il comportamento delle istruzioni test e jZ. Se il registro EAX è uguale a zero (come verificato dall'istruzione test), viene effettuato un salto alla locazione 401032 (come specificato dall'istruzione jZ). Altrimenti, il codice continua nell'istruzione successiva.



FUNZIONALITÀ MALWARE

```
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  ; "GinaDLL"  
mov     eax, [ebp+hObject]  
push    eax                 ; hKey  
call    ds:RegSetValueExA  
test   eax, eax  
short loc_401062
```

Analizzando attentamente le due chiamate di funzione precedentemente esaminate, emerge chiaramente che il malware è coinvolto nella creazione di una nuova chiave di registro, attribuendole il valore "GinaDLL". Approfondendo la ricerca online, è emerso che questa specifica chiave di registro è progettata per gestire o alterare il processo di accesso standard di Windows.

Questo porta a concludere che il malware potrebbe essere stato progettato con l'intenzione di modificare il normale flusso di accesso a Windows. In altre parole, sembra che il software dannoso stia intervenendo nel processo di login, suggerendo un potenziale tentativo di compromettere o manipolare il sistema di autenticazione di Windows a fini malevoli. La chiave di registro "GinaDLL" potrebbe essere coinvolta in tale manipolazione, rappresentando un elemento cruciale nel tentativo del malware di influenzare il comportamento del sistema operativo durante le fasi di login.



DAY 3

VALORE PARAMETRO

The screenshot shows the assembly view of OllyDbg. The assembly code is as follows:

```
00401086: 56          PUSH ESI
00401087: 57          PUSH EDI
00401088: C745 EC 000000 MOV [LOCAL.5],0
0040108F: C745 E8 000000 MOV [LOCAL.6],0
00401096: C745 F8 000000 MOV [LOCAL.2],0
0040109D: C745 F0 000000 MOV [LOCAL.4],0
004010A4: C745 F4 000000 MOV [LOCAL.3],0
004010AB: 837D 08 00    CMP [ARG.1],0
004010AF: v75 07       JNZ SHORT Malware_.004010B8
004010B1: 33C0         XOR EAX,EAX
004010B3: vE9 07010000  JMP Malware_.004011BF
004010B8: > A1 30804000 MOV EAX,DWORD PTR DS:[408030]
004010BD: 50          PUSH EAX
004010BE: 8B0D 34804000 MOV ECX,DWORD PTR DS:[408034]
004010C4: 51          PUSH ECX
004010C5: 8B55 08       MOV EDX,[ARG.1]
004010C8: 52          PUSH EDX
004010C9: FF15 28704000 CALL DWORD PTR DS:[&KERNEL32.FindResourceA]
004010CF: 8945 EC       MOV [LOCAL.5],EAX
004010D2: 837D EC 00    CMP [LOCAL.5],0
004010D6: v75 07       JNZ SHORT Malware_.004010DF
004010D8: 33C0         XOR EAX,EAX
004010DA: vE9 E00000000 JMP Malware_.004011BF
004010DF: > 8B45 EC     MOV EAX,[LOCAL.5]
004010E2: 50          PUSH EAX
004010E3: 8B4D 08       MOV ECX,[ARG.1]
004010E6: 51          PUSH ECX
004010E7: FF15 14294000 CALL DWORD PTR DS:[&KERNEL32.LoadResourceA]
```

Registers (Registers window):

EIP	00401487	Malware_.<ModuleEntryPoint>
C	0	ES 002B 32bit 0(FFFFFFFF)
P	1	CS 0023 32bit 0(FFFFFFFF)
A	0	SS 002B 32bit 0(FFFFFFFF)
Z	1	DS 002B 32bit 0(FFFFFFFF)
S	0	FS 0053 32bit 7EFDD000(FFF)
T	0	GS 002B 32bit 0(FFFFFFFF)
D	0	
O	0	LastErr ERROR_SUCCESS (00000000)
EFL	00000246	(NO,NB,E,BE,NS,PE,GE,LE)
ST0	empty	0.0
ST1	empty	0.0
ST2	empty	0.0
ST3	empty	0.0
ST4	empty	0.0
ST5	empty	0.0
ST6	empty	0.0
ST7	empty	0.0
FST	0000	Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0
FCW	027F	Prec NEAR,53 Mask 1 1 1 1 1 1 1 1

Call stack (Registers window):

- ResourceType => "BINARY"
- Malware_.00408038
- ResourceName => "TGAD"
- hModule FindResourceA
- hResource
- hModule LoadResourceA

Dall' analisi in OllyDbg possiamo dedurre che il parametro passato dalla funzione `FindResource()` è "TGAD".
Ciò è stato possibile dopo aver analizzato ed eseguito il codice tra gli indirizzi 00401080 e 004010C9.



DAY 3

FUNZIONALITÀ DEL MALWARE

L'andamento concatenato delle chiamate alle funzioni `FindResource()`, `LoadResource()`, e `SizeOfResource()` suggerisce una potenziale attività maligna associata a un dropper. In altre parole, all'avvio del programma, sembra che venga estratto un componente dalla sezione delle risorse, con `GINA.dll` indicato come un possibile candidato. Questa supposizione è stata formulata sulla base di un'analisi statica, supportata dalle tracce evidenziate dalle chiamate di funzione e dalla presenza della sezione `.rsrc`.

```
004010BD . 50          PUSH EAX
004010BE . 8B0D 34804000 MOV ECX,DWORD PTR DS:[408034]
004010C4 . 51          PUSH ECX
004010C5 . 8B55 08      MOV EDX,[ARG.1]
004010C8 . 52          PUSH EDX
004010C9 . FF15 28704000 CALL DWORD PTR DS:[<&KERNEL32.FindResourceA>]
004010CF . 8945 EC      MOV [LOCAL.5],EAX
004010D2 . 837D EC 00    CMP [LOCAL.5],0
004010D6 . ^75 07       JNZ SHORT Malware_.004010DF
004010D8 . 33C0          XOR EAX,EAX
004010DA . ^E9 E00000000 JMP Malware_.004011BF
004010DF > 8B45 EC      MOV EAX,[LOCAL.5]
004010E2 . 50          PUSH EAX
004010E3 . 8B4D 08      MOV ECX,[ARG.1]
004010E6 . 51          PUSH ECX
004010E7 . FF15 14704000 CALL DWORD PTR DS:[<&KERNEL32.LoadResource>]
004010ED . 8945 E8      MOV [LOCAL.6],EAX
004010F0 . 837D E8 00    CMP [LOCAL.6],0
004010F4 . ^75 05       JNZ SHORT Malware_.004010FB
004010F6 . ^E9 AA000000 JMP Malware_.004011A5
004010FB > 8B55 E8      MOV EDX,[LOCAL.6]
004010FE . 52          PUSH EDX
004010FF . FF15 10704000 CALL DWORD PTR DS:[<&KERNEL32.LockResource>]
00401105 . 8945 F8      MOV [LOCAL.2],EAX
00401108 . 837D F8 00    CMP [LOCAL.2],0
0040110C . ^75 05       JNZ SHORT Malware_.00401113
0040110E . ^E9 92000000 JMP Malware_.004011A5
00401113 > 8B45 EC      MOV EAX,[LOCAL.5]
00401116 . 50          PUSH EAX
00401117 . 8B4D 08      MOV ECX,[ARG.1]
0040111A . 51          PUSH ECX
0040111B . FF15 0C704000 CALL DWORD PTR DS:[<&KERNEL32.SizeofResource>]
```

```
ResourceType => "BINARY"
Malware_.00408038
ResourceName => "TGAD"

hModule FindResourceA
hResource LoadResource
hResource LockResource
hResource SizeofResource
```



Riflessioni analisi statica

The screenshot shows a debugger interface with two main panes. On the left, a tree view lists various functions: sub_401000, sub_401080, _main, sub_401299, _fclose, _fwrite, _fopen, _fopen, _strchr, start, _amsq_exit, _fast_error_exit, and stbuf. The _main function is currently selected. On the right, the assembly code for the _main function is displayed:

```
.text:004010B3          jmp    loc_4011BF
.text:004010B8 ; -----
.text:004010B8 loc_4010B8:
.text:004010B8          mov    eax, lpType
.text:004010BD          push   eax           ; lpType
.text:004010BE          mov    ecx, lpName
.text:004010C4          push   ecx           ; lpName
.text:004010C5          mov    edx, [ebp+Module1]
.text:004010C8          push   edx           ; LPCSTR lpName
.text:004010C9          push   [lpName]        dd offset aTGad
.text:004010CF          mov    [ebp+messaggio], eax
.text:004010D2          cmp    [ebp+hResInfo], 0
.text:004010D6          jnz   short loc_4010DF
.text:004010D8          xor    eax, eax
.text:004010DA          jmp    loc_4011BF
.text:004010DF ; -----
```

The assembly code is annotated with comments: 'CODE XREF: sub_401080+2F†j' for the first mov instruction, and 'DATA XREF: sub_401080+3E†r; "TGAD"' for the dd offset instruction. The variable 'lpName' is highlighted in yellow in several places. The memory address 000010C4 is also highlighted.

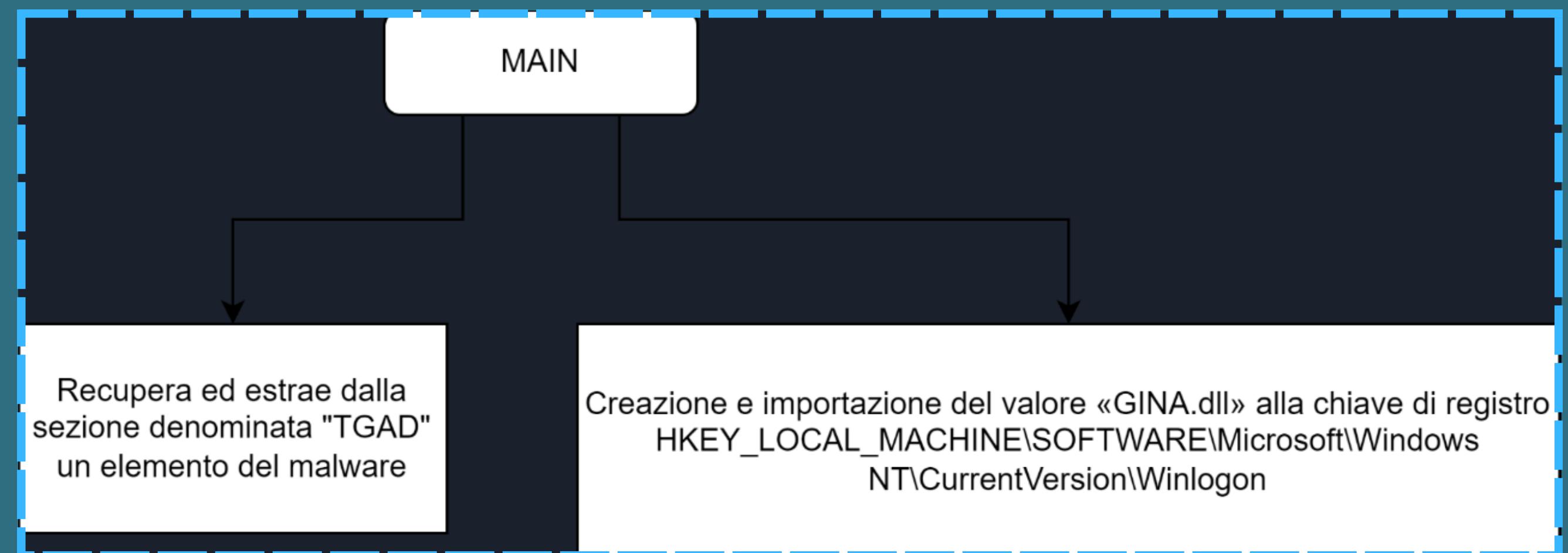
Ciò che non era noto inizialmente è la presenza all'interno della sezione .rsrc di una sotto-sezione denominata "TGAD>>". Questa sotto-sezione è stata identificata come il parametro utilizzato da FindResourceA() per estrarre il componente dannoso. L'inserimento di questa informazione aggiuntiva ha contribuito a delineare più chiaramente il processo mediante il quale il malware agisce durante l'estrazione del componente nocivo, confermando le ipotesi basate sull'analisi statica originale.



DAY 3

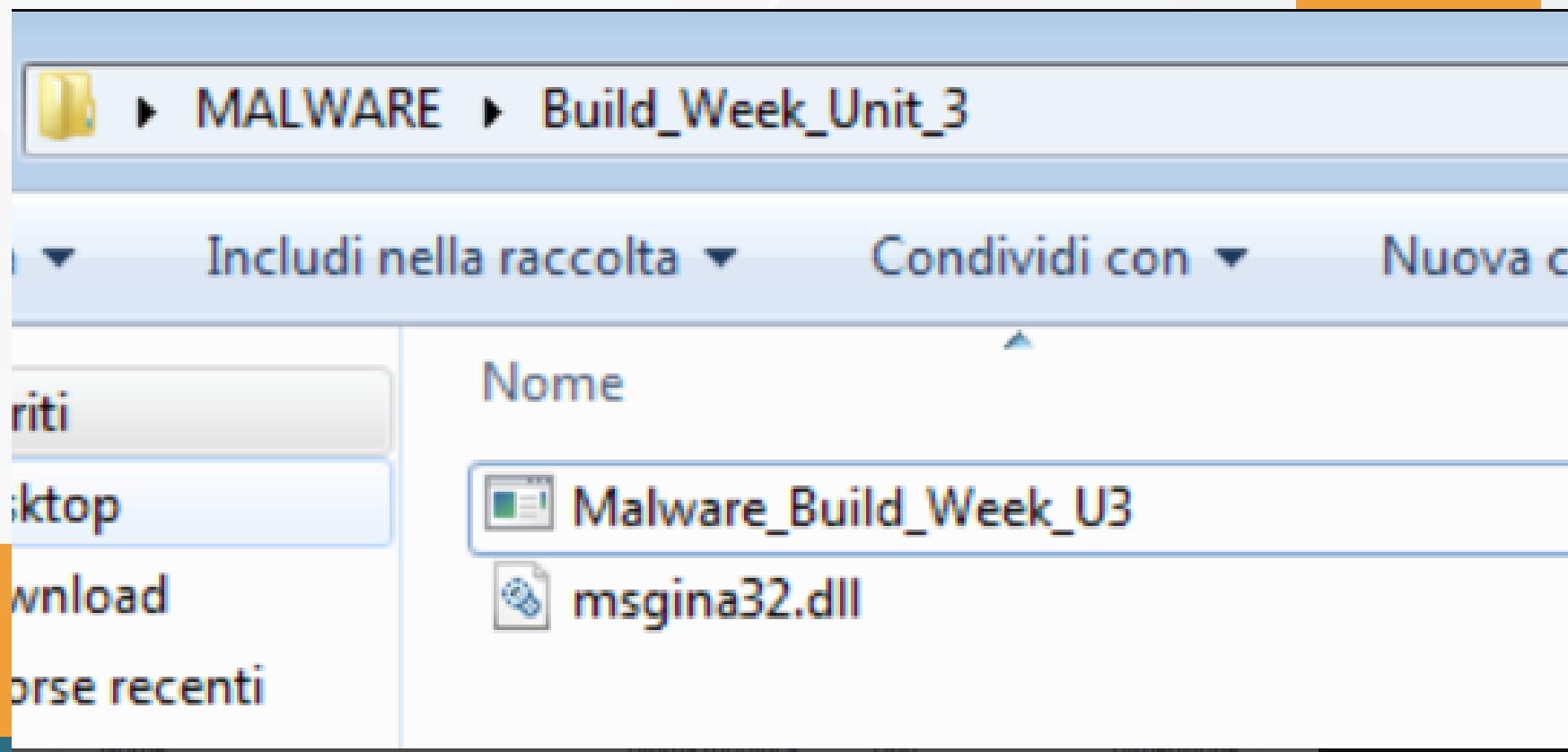
DIAGRAMMA DI FLUSSO

Ecco un diagramma di flusso che rappresenta le funzioni principali:





DAY 4



Una volta eseguito il malware, potrebbe comparire nella stessa cartella dell'eseguibile un file DLL denominato msgina32.dll. Questo file potrebbe rappresentare una versione dannosa del file GINA.dll, un componente legittimo di Windows.



DAY 4

CHIAVE DI REGISTRO

20:55:...	Malware_Build_...	1032	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
20:55:...	Malware_Build_...	1032	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
20:55:...	Malware_Build_...	1032	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
20:55:...	Malware_Build_...	1032	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
20:55:...	Malware_Build_...	1032	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon

*Esaminando tramite l'utilizzo di Procmon notiamo che viene creata la chiave di registro:
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon.
Il valore assegnato alla chiave è GinaDLL.*



La chiamate alla funzione CreateFile e la responsabile della creazione di msgina32.dll

20:55:...	Malware_Build_...	1032	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3
20:55:...	Malware_Build_...	1032	CreateFile	C:\Windows\SysWOW64\sechost.dll
20:55:...	Malware_Build_...	1032	QueryBasicInfor...	C:\Windows\SysWOW64\sechost.dll
20:55:...	Malware_Build_...	1032	CloseFile	C:\Windows\SysWOW64\sechost.dll
20:55:...	Malware_Build_...	1032	CreateFile	C:\Windows\SysWOW64\sechost.dll
20:55:...	Malware_Build_...	1032	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll
20:55:...	Malware_Build_...	1032	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll
20:55:...	Malware_Build_...	1032	CloseFile	C:\Windows\SysWOW64\sechost.dll
20:55:...	Malware_Build_...	1032	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
20:55:...	Malware_Build_...	1032	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
20:55:...	Malware_Build_...	1032	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
20:55:...	Malware_Build_...	1032	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll

Attraverso l'applicazione di tecniche di analisi statica e dinamica, abbiamo compreso il funzionamento del malware. Dopo essere stato installato sulla macchina della vittima e eseguito dall'utente, il malware opererà a livello software per creare una chiave di registro e impostare il suo valore come GINA.DLL. Quest'ultimo verrà quindi sostituito al componente GINA originale.

Diagramma di flusso Malware

Sostituire un file DLL legittimo con uno malevolo, il quale intercetta i dati inseriti, costituisce un'azione che consente all'attaccante di recuperare le credenziali di accesso dell'utente.

È rilevante sottolineare che tali informazioni verosimilmente vengono salvate in un file di input utente oppure trasmesse direttamente all'attaccante.





GRAZIE!