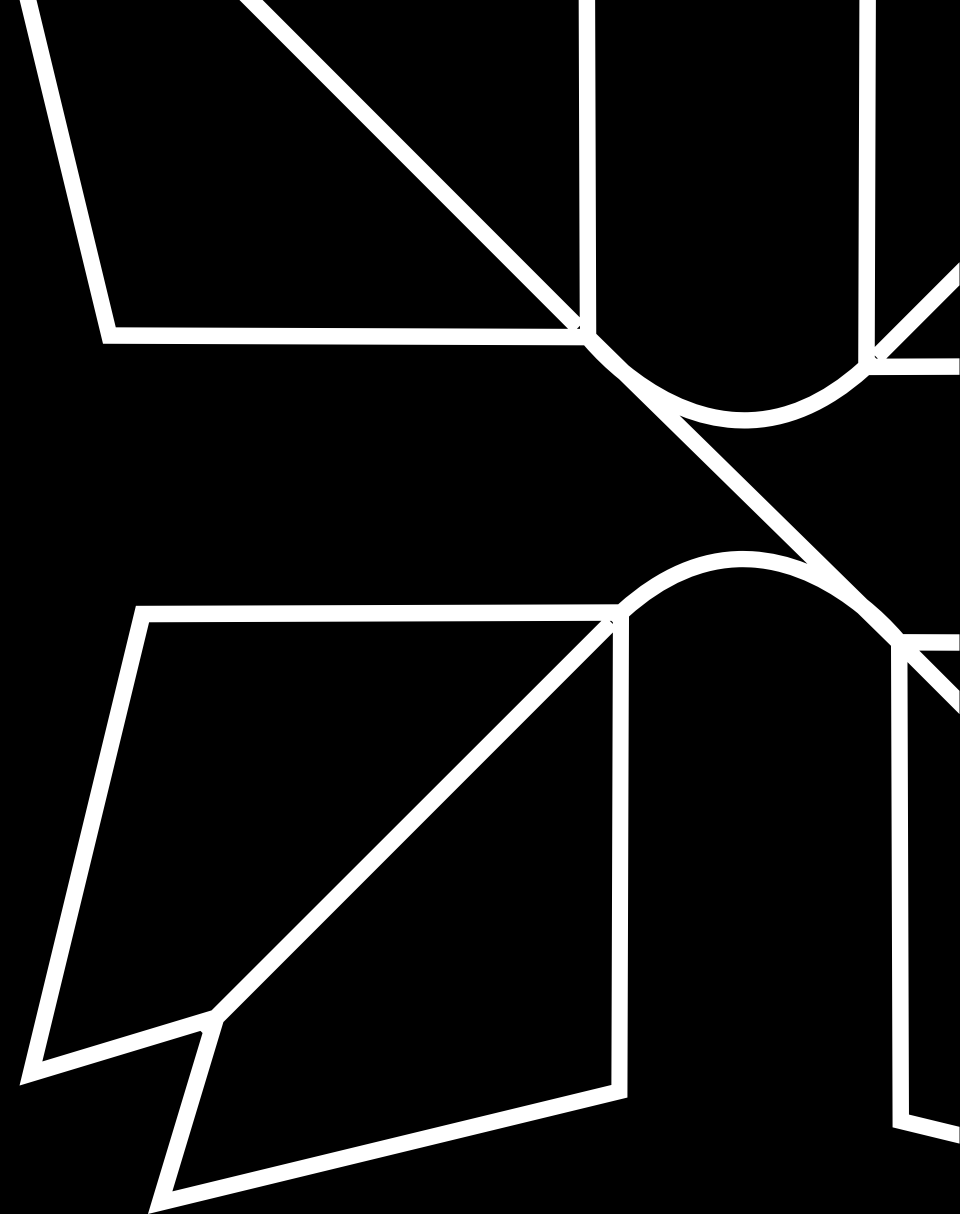
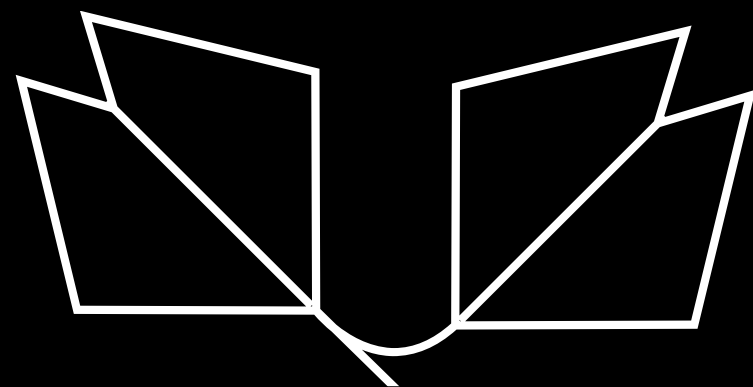




BUILD WEEK 3

NETRAIDERS

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:

- **QUANTI PARAMETRI SONO PASSATI ALLA FUNZIONE MAIN()?**
- **QUANTE VARIABILI SONO DICHIARATE ALL'INTERNO DELLA FUNZIONE MAIN()?**
- **QUALI SEZIONI SONO PRESENTI ALL'INTERNO DEL FILE ESEGUIBILE? DESCRIVETE BREVEMENTE ALMENO 2 DI QUELLE IDENTIFICATE**
- **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.**



GIORNO 1

```
.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
```

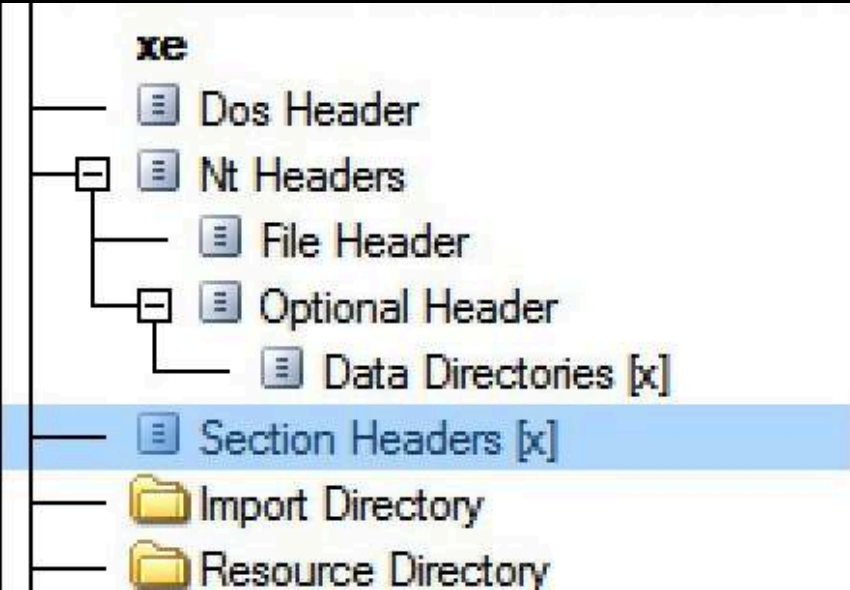
I **Parametri** che vengono passati sono 3 e sono evidenziati dal riquadro rosso.

Essi possono essere utilizzati dalla funzione per eseguire calcoli, prendere decisioni o manipolare dati

Le **Variabili** dichiarate sono 5 e sono evidenziate verde.

Queste solitamente contengono dati da utilizzare durante l'esecuzione del programma, ad esempio, numeri, caratteri, etc.

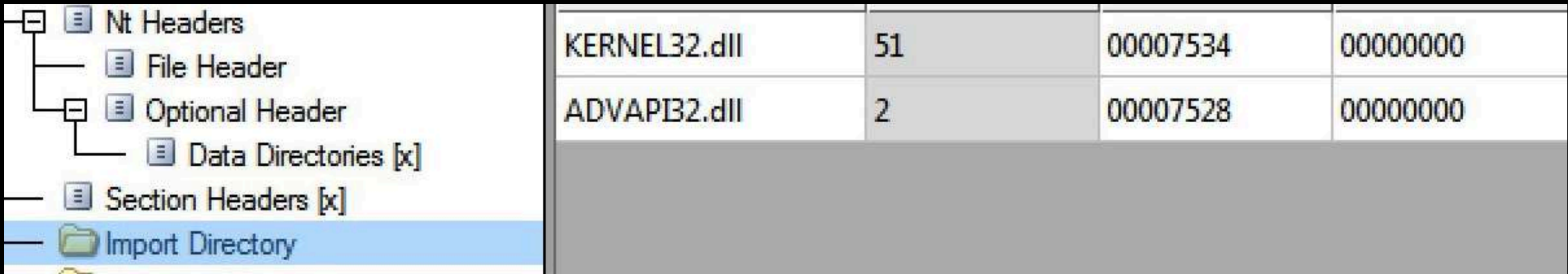
La principale differenza tra i due riguarda l'offset, negativo per le variabili e positivo per i parametri.

	Byte[8]	Dword	Dword
	.text	00005646	00001000
	.rdata	000009AE	00007000
	.data	00003EA8	00008000
	.rsrc	00001A70	0000C000

Le sezioni trovate sono 4:

- “.text”; contiene istruzioni che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l’unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.
- “.rdata”; include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall’eseguibile.
- “.data”; contiene tipicamente dati o variabili globali del programma eseguibile e devono essere disponibili da qualsiasi parte del programma. Una variabile è dichiarata globalmente, quindi non all’interno di una sola funzione, quando è disponibile ed accessibile da qualsiasi funzione del programma
- “.rsrc”; include le risorse utilizzate dall’eseguibile come icone, immagini, menu stringhe che non sono parte dell’eseguibile stesso.

GIORNO 1



KERNEL32.dll	51	00007534	00000000
ADVAPI32.dll	2	00007528	00000000

Le *librerie* importate sono due e sono le seguenti:

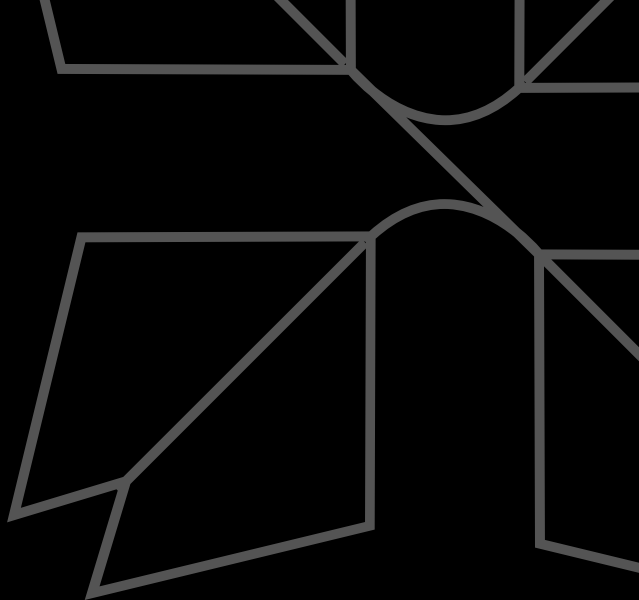
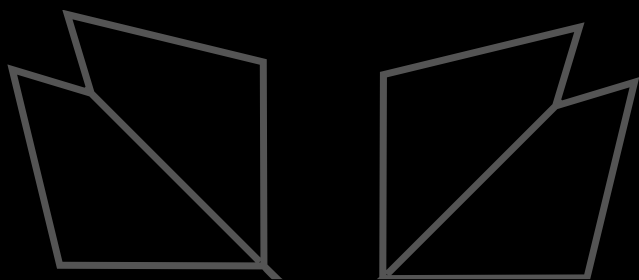
- “**KERNEL32.dll**”; contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione dei file, la gestione della memoria.
- “**ADVAPI32.dll**”; contiene le funzioni per interagire con i servizi ed i registri del sistema operativo.

Dall’analisi iniziale il programma potrebbe cercare la persistenza sfruttando la libreria *ADVAPI32* tramite le funzioni



00407000	RegSetValueExA	ADVAPI32
00407004	RegCreateKeyExA	ADVAPI32

Mentre dalla libreria *KERNEL32* notiamo le funzioni per prendere il controllo dei processi come ci fa intuire la funzione GetProcAddress.



GIORNO 1

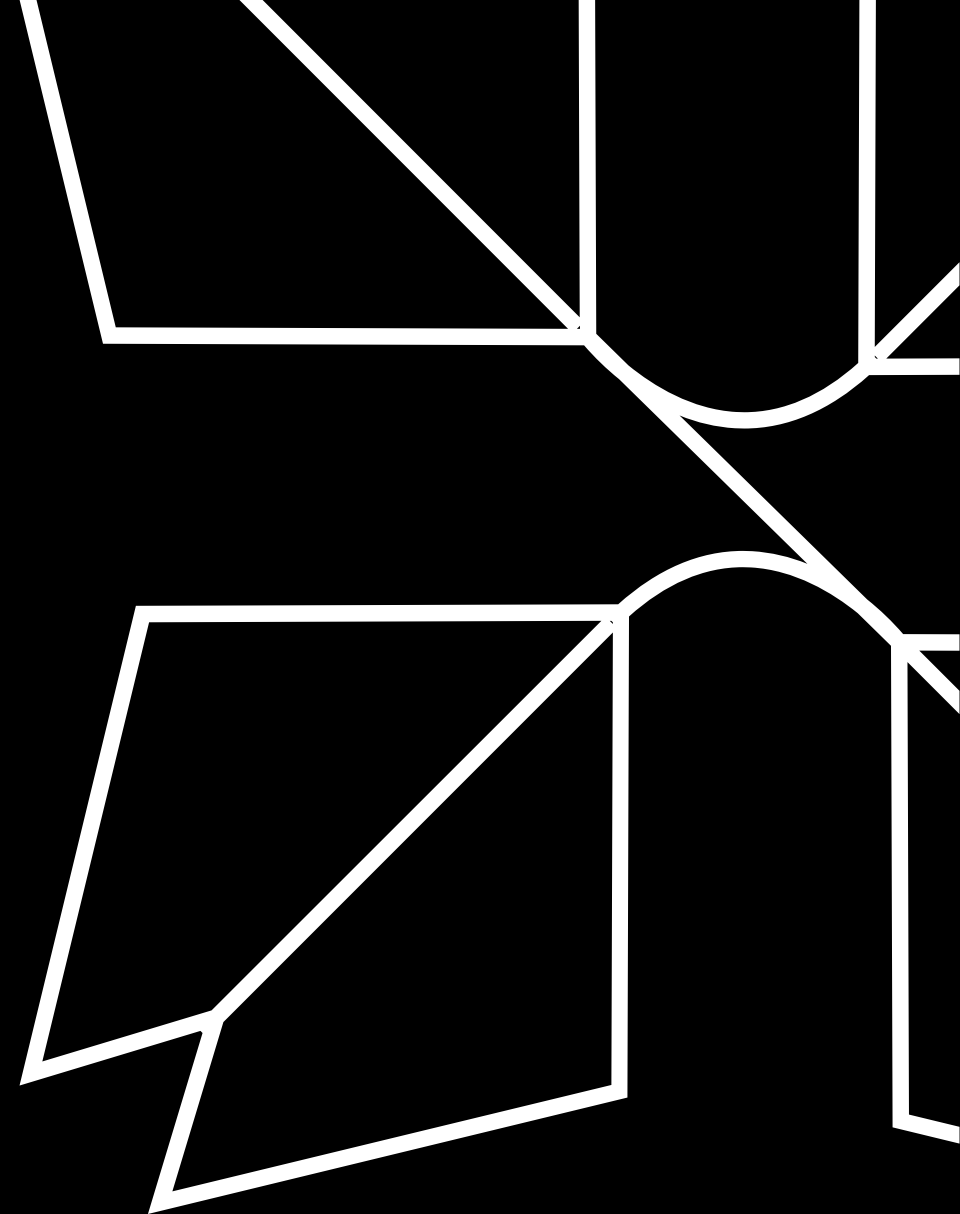
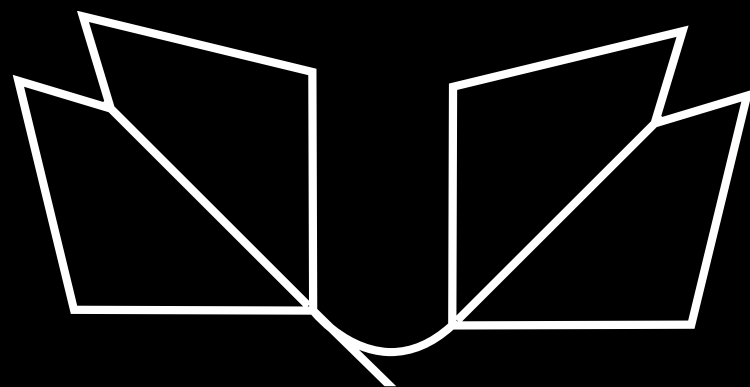
OFTs	FTs (IAT)	Hint	Name
00007558	00007030	000076DE	000076E0
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007604	00007604	001B	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA
000076F0	000076F0	0174	GetVersion

Analizzando CFF Explorer sono state trovate funzioni che caratterizzano la struttura di un Dropper, come ad esempio SizeofResource, FindResource,etc



NETRAIDERS

GIORNO 2



CON RIFERIMENTO AL MALWARE IN ANALISI, SPIEGARE:

- **LO SCOPO DELLA FUNZIONE CHIAMATA ALLA LOCAZIONE DI MEMORIA 00401021**
- **COME VENGONO PASSATI I PARAMETRI ALLA FUNZIONE ALLA LOCAZIONE 00401021 ;**
- **CHE OGGETTO RAPPRESENTA IL PARAMETRO ALLA LOCAZIONE 00401017**
- **IL SIGNIFICATO DELLE ISTRUZIONI COMPRESSE TRA GLI INDIRIZZI 00401027 E 00401029. (SE SERVE, VALUTATE ANCHE UN'ALTRA O ALTRE DUE RIGHE ASSEMBLY)**
- **CON RIFERIMENTO ALL'ULTIMO QUESITO, TRADURRE IL CODICE ASSEMBLY NEL CORRISPONDENTE COSTRUTTO C**
- **VALUTATE ORA LA CHIAMATA ALLA LOCAZIONE 00401047, QUAL È IL VALORE DEL PARAMETRO « VALUENAME »?**

NEL COMPLESSO DELLE DUE FUNZIONALITÀ APPENA VISTE, SPIEGATE QUALE FUNZIONALITÀ STA IMPLEMENTANDO IL MALWARE IN QUESTA SEZIONE.




```
.text:0040101C      push    80000002h      ; hKey
.text:00401021      call    ds:RegCreateKeyExA
.text:00401027      test    eax, eax
```

Nella locazione 00401021 passa la funzione **RegCreateKeyExA**. Questa funzione è una API di Windows e serve a creare una nuova chiave, o a modificarne una già esistente, nei registri del sistema operativo

```
.text:00401007      push    eax             ; lpSecurityAttributes
.text:0040100A      push    0               ; lpSecurityAttributes
.text:0040100C      push    0F003Fh         ; samDesired
.text:00401011      push    0               ; dwOptions
.text:00401013      push    0               ; lpClass
.text:00401015      push    0               ; Reserved
.text:00401017      push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe".
.text:0040101C      push    80000002h       ; hKey
.text:00401021      call    ds:RegCreateKeyExA
.text:00401027      test    eax, eax
.text:00401029      jz      short loc_401032
```

I Parametri vengono passati nello stack tramite l'istruzione **PUSH**, per poi essere chiamati dalla funzione. Il push evidenziato è un valore speciale che fa riferimento alla chiave di base HKLM

GIORNO 2

```
408054 ; char SubKey[]  
408054 SubKey          db 'SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon',
```

Il Parametro passato alla locazione 00401017 è la chiave di registro
SOFTWARE\ \Microsoft\ \Windows\CurrentVersion\WinLogOn

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

***La costruzione del codice
ricorda un ciclo if come segue:***

```
if (hObject == 0) { //Quello che viene caricato in eax  
    RegSetValueExA(hObject, ValueName, Reserved, dwType, val_lpData, val_cbData);  
    //per quanto riguarda i parametri, vengono inseriti i loro valori  
    CloseHandle(hObject);  
}
```

- *test eax, eax* → Esegue un AND bit a bit tra EAX e EAX ; Modifica i flag senza cambiare il valore di EAX; *jz short*
- *loc_401032* → Salta all'indirizzo *loc_401032* se il flag ZF è 1 (risultato AND = 0) *mov eax, 1* → Carica il valore 1 in EAX
- *jmp short loc_40107B* → Salta incondizionatamente all'indirizzo *loc_40107B*

GIORNO 2

```

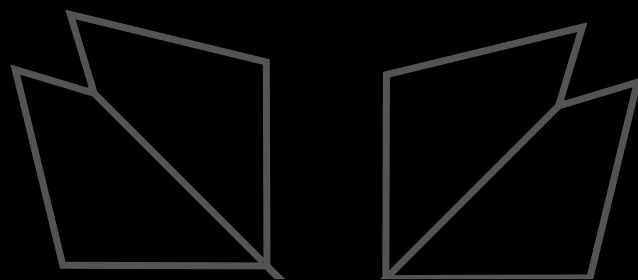
* .text:00401035      mov     ecx, [ebp+cbData]
* .text:00401036      push    ecx                ; cbData
* .text:00401039      mov     edx, [ebp+lpData]
* .text:0040103A      push    edx                ; lpData
* .text:0040103C      push    1                  ; dwType
* .text:0040103E      push    0                  ; Reserved
* .text:00401043      push    offset ValueName ; "GinaDLL"
* .text:00401046      mov     eax, [ebp+hObject]
* .text:00401047      push    eax                ; hKey
* .text:0040104D      call    ds:RegSetValueExA
* .text:0040104F      test    eax, eax
* .text:00401051      jz      short loc_401062
* .text:00401051      mov     ecx, [ebp+hObject]

```

Spostandoci alla locazione *00401047* dobbiamo dire quale sia il valore di *ValueName*. facendo una prima analisi possiamo vedere che il valore del parametro è "***GinaDLL***"

Considerando quel che è stato visto fino ad adesso e le funzionalità viste il malware sembra sfruttare il registro per creare una ***persistenza*** all'interno del sistema.

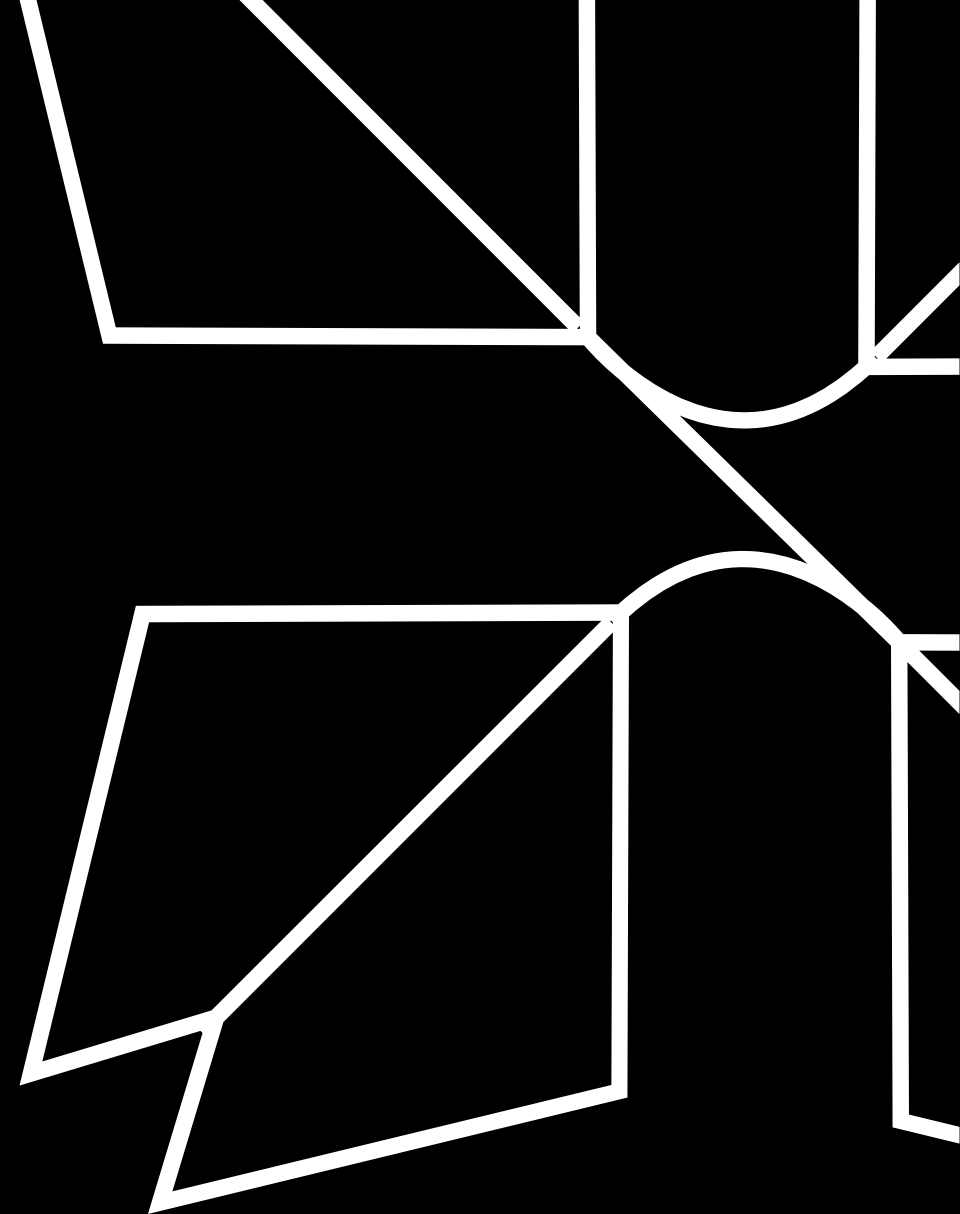
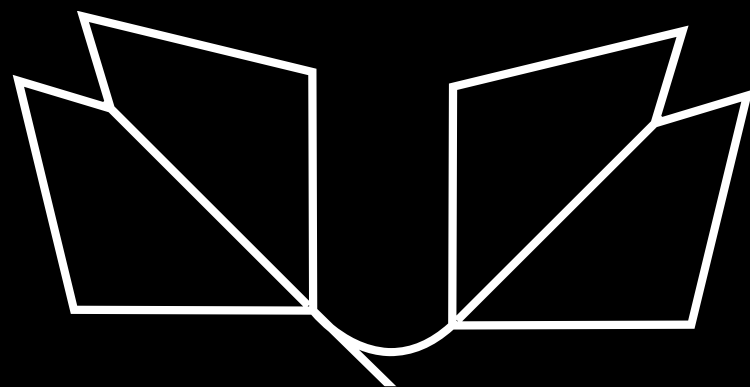
Un'altra particolarità viene da *GinaDLL*: Lo scopo di una *DLL GINA* è fornire procedure personalizzabili di ***identificazione*** e ***autenticazione*** dell'utente, questo fa presupporre che il malware voglia intercettare i login sfruttando questo sistema di windows di autenticazione.





NETRAIDERS

GIORNO 3



GIORNO 3

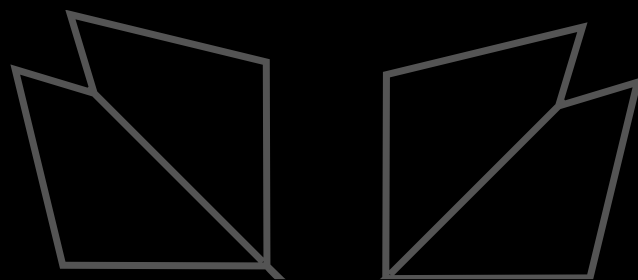
RIPRENDETE L'ANALISI DEL CODICE, ANALIZZANDO LE ROUTINE TRA LE LOCAZIONI DI MEMORIA 00401080 E 00401128 :

- QUAL È IL VALORE DEL PARAMETRO «RESOURCE_NAME » PASSATO ALLA FUNZIONE FINDRESOURCEA ();
- 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 ?
- È POSSIBILE IDENTIFICARE QUESTA FUNZIONALITÀ UTILIZZANDO L'ANALISI STATICA BASICA? (DAL GIORNO 1 IN PRATICA)
- 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().

DISEGNARE UN DIAGRAMMA DI FLUSSO PRINCIPALI) CHE COMPRENDA LE 3 FUNZIONI.

MALWARE VISTE FINORA SONO RICHIAMATE ALL'INTERNO DELLA FUNZIONE (INSERITE ALL'INTERNO DEI BOX SOLO LE INFORMAZIONI CIRCA LE FUNZIONALITÀ
MAIN ()



GIORNO 3

```
.text:004010B8  
.text:004010B8 loc_4010B8: ; CODE XREF: sub_401080+2F↑j  
* .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+hModule]  
* .text:004010C8      push    edx           ; hModule  
* .text:004010C9      call   ds:FindResourceA  
* .text:004010CF      mov     [ebp+hResInfo], eax  
* .text:004010D2      cmp     [ebp+hResInfo], 0  
* .text:004010D6      jnz     short loc_4010DF  
* .text:004010D8      xor     eax, eax  
* .text:004010DA      jmp     loc_4011BF  
.text:004010DF ; -----
```

La funzione FindResourceA ha bisogno di tre parametri che sono:

- lpType;
- lpName;
- hModule.

Andando poi a controllare il valore di lpName possiamo vedere che esso è "TGAD"

```
.data:00408030 lpType      dd offset aBinary ; DATA XREF: sub_401080+100↑  
.data:00408030 ; "BINARY"  
* .data:00408034 ; LPCSTR lpName  
* .data:00408034 lpName      dd offset aTgad ; DATA XREF: sub_401080+3E↑  
* .data:00408034 ; "TGAD"  
* .data:00408038 aTgad      db 'TGAD',0 ; DATA XREF: .data:lpName↑  
* .data:0040803D align 10h  
* .data:00408040 aBinary     db 'BINARY',0 ; DATA XREF: .data:lpType↑
```


GIORNO 3

```
.text:004010B8 loc_4010B8:      ; CODE XREF: sub_401080+2F↑j
.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+hModule]
.text:004010C8      push    edx                ; hModule
.text:004010C9      call    ds:FindResourceA
.text:004010CF      mov     [ebp+hResInfo], eax
.text:004010D2      cmp     [ebp+hResInfo], 0
.text:004010D6      jnz     short loc_4010DF
.text:004010D8      xor     eax, eax
.text:004010DA      jmp     loc_4011BF
```

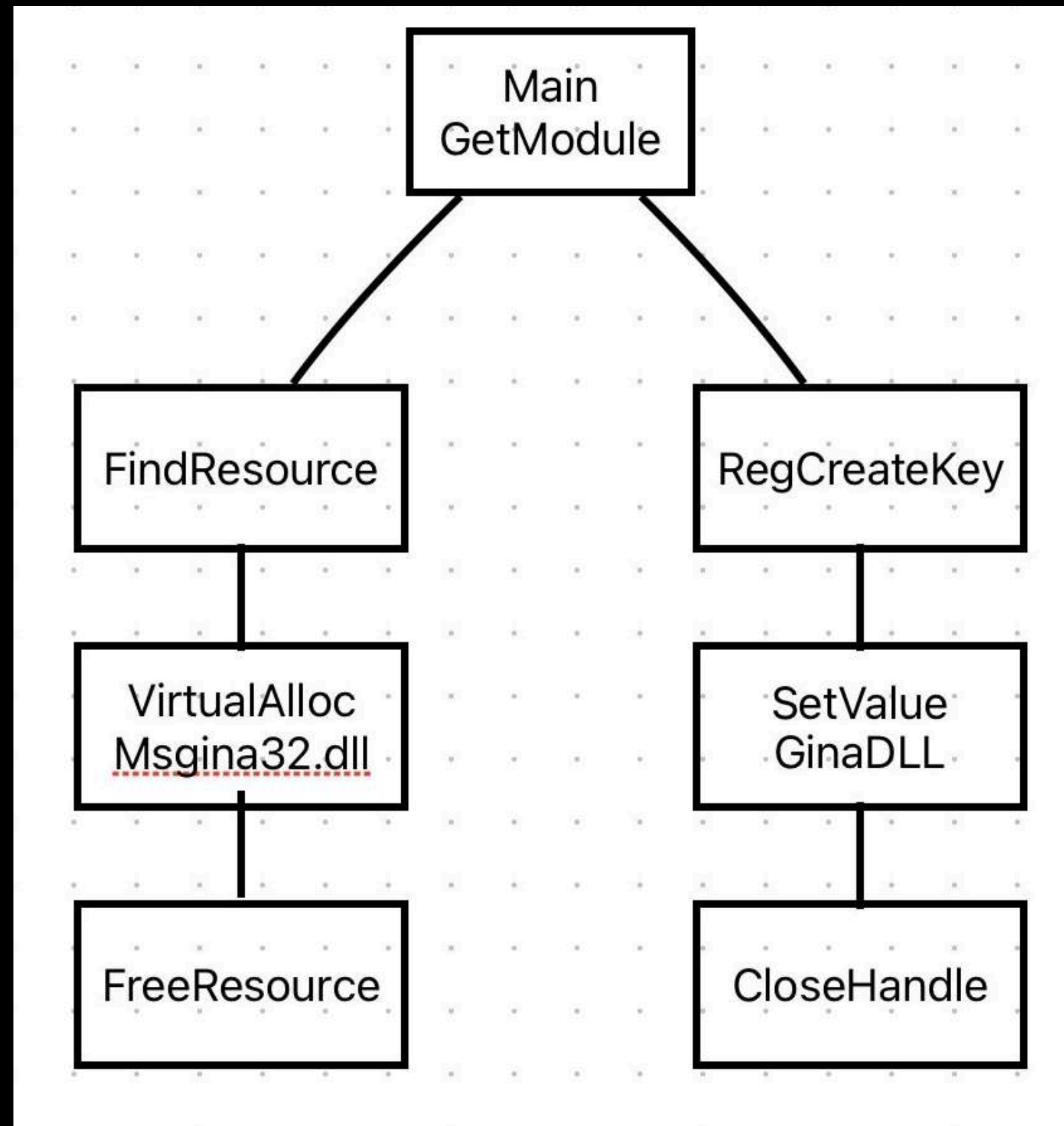
Il malware cerca una risorsa di tipo binario
all'interno della sezione risorse TGAD all'interno
del suo PE, che desumiamo essere msgina32.dll,
per poi copiarla nella stessa cartella
dell'eseguibile del malware.

Abbiamo la conferma che si tratti di un *Dropper*

```
.text:00401113 loc_401113:      ; CODE XREF: sub_401080+8C↑j
.text:00401113      mov     eax, [ebp+hResInfo]
.text:00401116      push    eax                ; hResInfo
.text:00401117      mov     ecx, [ebp+hModule]
.text:0040111A      push    ecx                ; hModule
.text:0040111B      call    ds:SizeofResource
.text:00401121      mov     [ebp+Count], eax
.text:00401124      cmp     [ebp+Count], 0
.text:00401128      ja      short loc_40112C
.text:0040112A      jmp     short loc_4011A5
.text:0040112C
```

```
.text:004010DF      mov     eax, [ebp+hResInfo]
.text:004010E2      push    eax                ; hResInfo
.text:004010E3      mov     ecx, [ebp+hModule]
.text:004010E6      push    ecx                ; hModule
.text:004010E7      call    ds:LoadResource
.text:004010ED      mov     [ebp+hResData], eax
.text:004010F0      cmp     [ebp+hResData], 0
.text:004010F4      jnz     short loc_4010FB
.text:004010F6      jmp     loc_4011A5
.text:004010FB      ;
.text:004010FB      loc_4010FB:      ; CODE XREF: sub_401080+74↑j
.text:004010FB      mov     edx, [ebp+hResData]
.text:004010FE      push    edx                ; hResData
.text:004010FF      call    ds:LockResource
.text:00401105      mov     [ebp+Str], eax
.text:00401108      cmp     [ebp+Str], 0
.text:0040110C      jnz     short loc_401113
.text:0040110E      jmp     loc_4011A5
.text:00401113
```

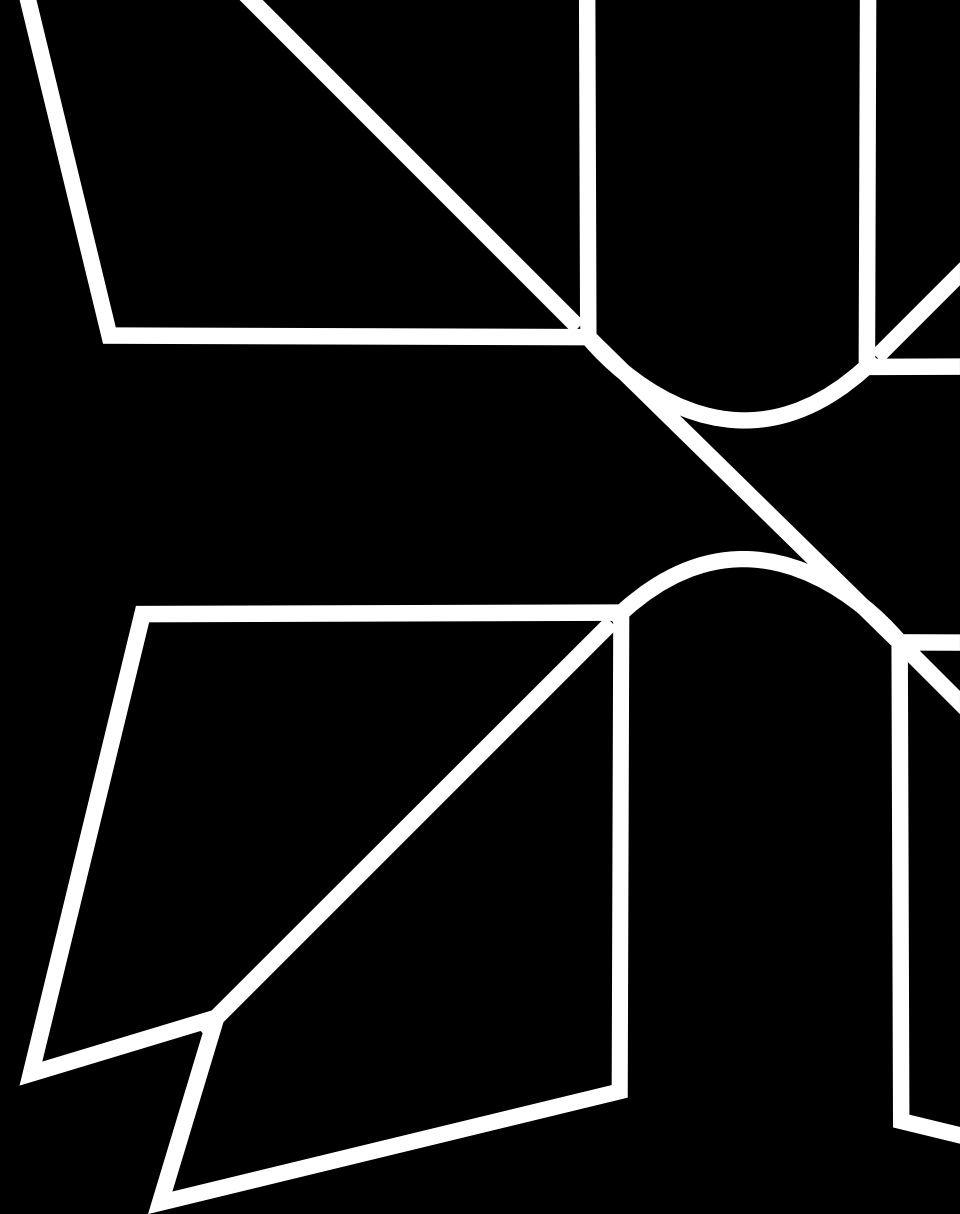
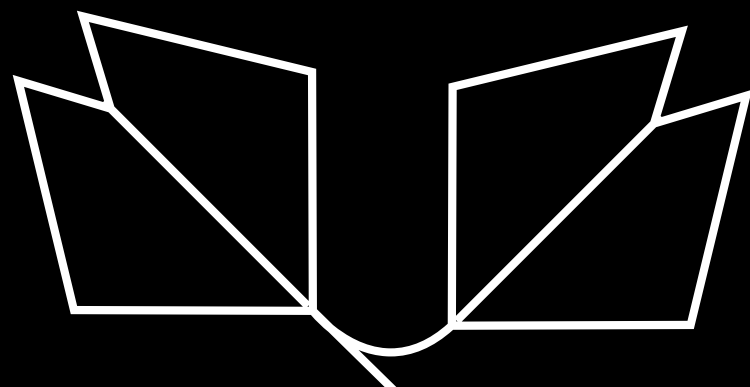

Come ipotizzato durante il primo giorno, il malware ha una funzione di Dropper, andando a caricare sul sistema un file DLL, in questo caso msgina32.dll. Si può affermare che è possibile capire questa funzionalità dall'analisi statica basica, in quanto le funzioni, che contraddistinguono questo tipo di malware, erano già presenti











NETRAIDERS

GIORNO 4



Una volta eseguito il malware possiamo notare che, all'interno della cartella dove esso è contenuto, viene creato un file dll chiamato msgina32

Nome	Ultima modifica	Tipo	Dimensione
 Malware_Build_Week_U3	17/01/2024 17:48	Applicazione	52 KB
 Malware_Build_Week_U3.id0	14/05/2024 07:55	File ID0	384 KB
 Malware_Build_Week_U3.id1	14/05/2024 07:55	File ID1	184 KB
 Malware_Build_Week_U3.nam	14/05/2024 07:55	File NAM	16 KB
 Malware_Build_Week_U3.til	08/05/2024 10:53	File TIL	1 KB
 msgina32.dll	14/05/2024 07:56	Estensione dell'ap...	7 KB

Dalle analisi precedenti ci aspettavmo una creazione di un file poiche il malware cercava una risorsa che poi chiamava msgina32.dll nella quale inietta probabilmente il suo codice

GIORNO 4

UTILIZZANDO POI PROCESS MONITOR POSSIAMO NOTARE:

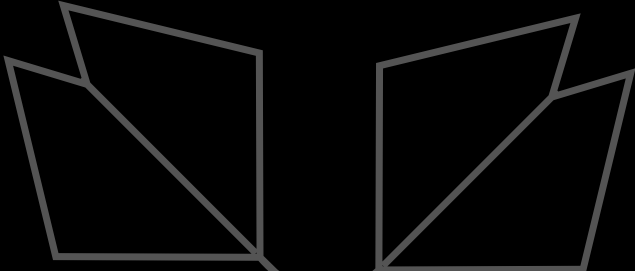
07:59:23,3640771	Malware_Build_Week_U3.exe	2200	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
07:59:23,3684023	Malware_Build_Week_U3.exe	2200	RegQueryKey	HKLM	SUCCESS
07:59:23,3685352	Malware_Build_Week_U3.exe	2200	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
07:59:23,3687671	Malware_Build_Week_U3.exe	2200	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CURRENTVERSION\Winlogon	SUCCESS
07:59:23,3688942	Malware_Build_Week_U3.exe	2200	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CURRENTVERSION\Winlogon	SUCCESS
07:59:23,3690113	Malware_Build_Week_U3.exe	2200	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CURRENTVERSION\Winlogon\Gin...	ACCESS DENIED
07:59:23,3690993	Malware_Build_Week_U3.exe	2200	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CURRENTVERSION\Winlogon	SUCCESS

La chiave di registro creata riguarda Winlogon che cerca si settare ma non gli viene concesso di farlo

07:59:23,3685352	Malware_Build_Week_U3.exe	2200	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All Accesses
07:59:23,3687671	Malware_Build_Week_U3.exe	2200	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CURRENTVERSION\Winlo...	SUCCESS	KeySetInformationClass: Ki...
07:59:23,3688942	Malware_Build_Week_U3.exe	2200	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CURRENTVERSION\Winlo...	SUCCESS	Query: HandleTags, Handl...
07:59:23,3690113	Malware_Build_Week_U3.exe	2200	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CURRENTVERSION\Winlo...	ACCESS DENIED	Type: REG_SZ, Length: 5...
07:59:23,3690993	Malware_Build_Week_U3.exe	2200	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CURRENTVERSION\Winlo...	SUCCESS	
07:59:23,3700502	Malware_Build_Week_U3.exe	2200	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS	

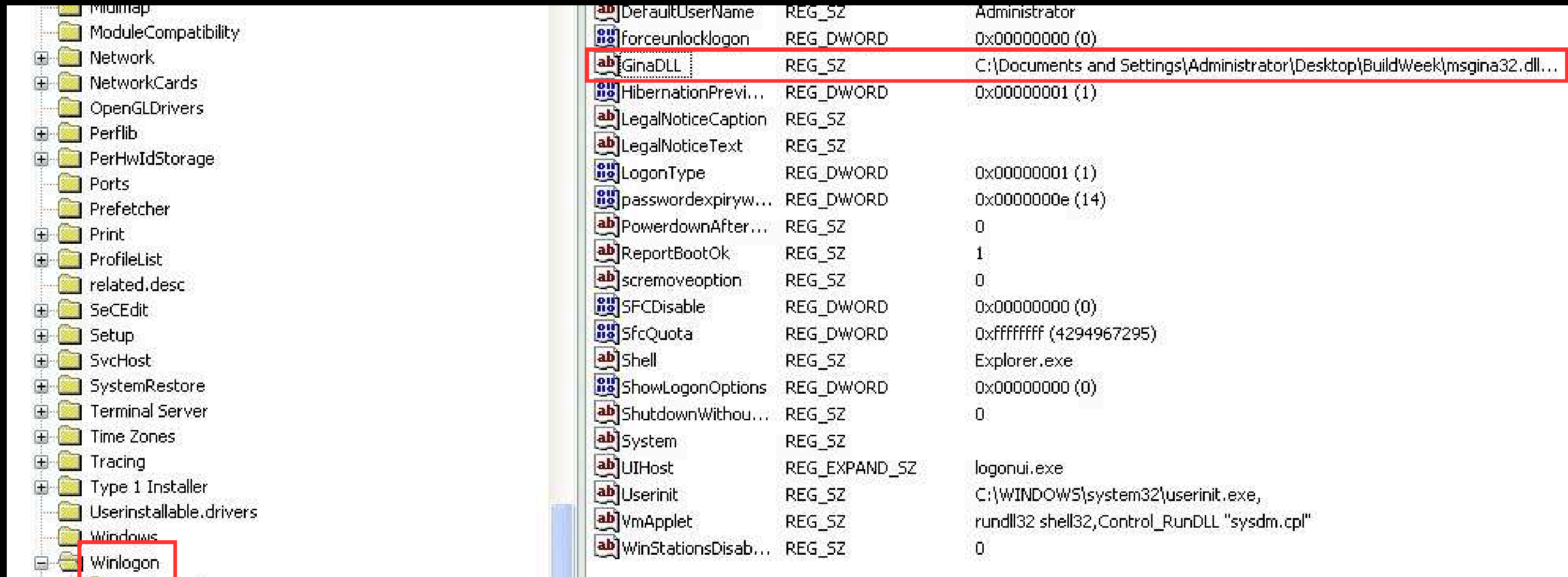
Qui il malware cerca di settare il valore della chiave ma non va a buon fine poichè viene bloccato l'accesso

Con la chiamata " CreateFile" la cartella del malware è stata aperta e manipolata per inserire il nuovo file che esso ha creato



GIORNO 4

A seguito di ricerche sul web si è notato che i dll gina vengono ignorati da windows vista in poi, per questo motivo andremo a testare questo malware su windows xp per capire il funzionamento delle chiavi e dei vari registri di sistema.



COME SI PUÒ NOTARE IN QUESTO CASO VIENE CORRETTAMENTE CREATA LA CHIAVE DLL "HKLM//SOFTWARE/MICROSOFT/WINDOWS NT/CURRENTVERSION/WINLOGON/GINADLL". AL SUO INTERNO VIENE COPIATO IL PATH DEL DLL MSGINA32.DLL CREATO NELLA STESSA CARTELLA DELL'ESEGUIBILE DEL MALWARE , IN QUESTO CASO "C//:DOCUMENTS AND SETTINGS/ADMINISTRATOR|DESKTOP|BUILDWEEK/MSGINA32.DLL". QUESTO DOVREBBE PERMETTERE DI INSERIRE SU UN DLL MALEVOLO DI GINA LE CREDENZIALI DI ACCESSO PER POI ESSERE UTILIZZATE DA UN UTENTE MALINTENZIONATO PER PRENDERE IL CONTROLLO DEL COMPUTER.

Dall'insieme delle analisi effettuate fino ad adesso si è notato che il malware vuole avere a che fare con questa dll chiamata *gina*. Quest'ultima è responsabile dell'interfaccia grafica e delle funzionalità di autenticazione quando si accede a un sistema Windows.

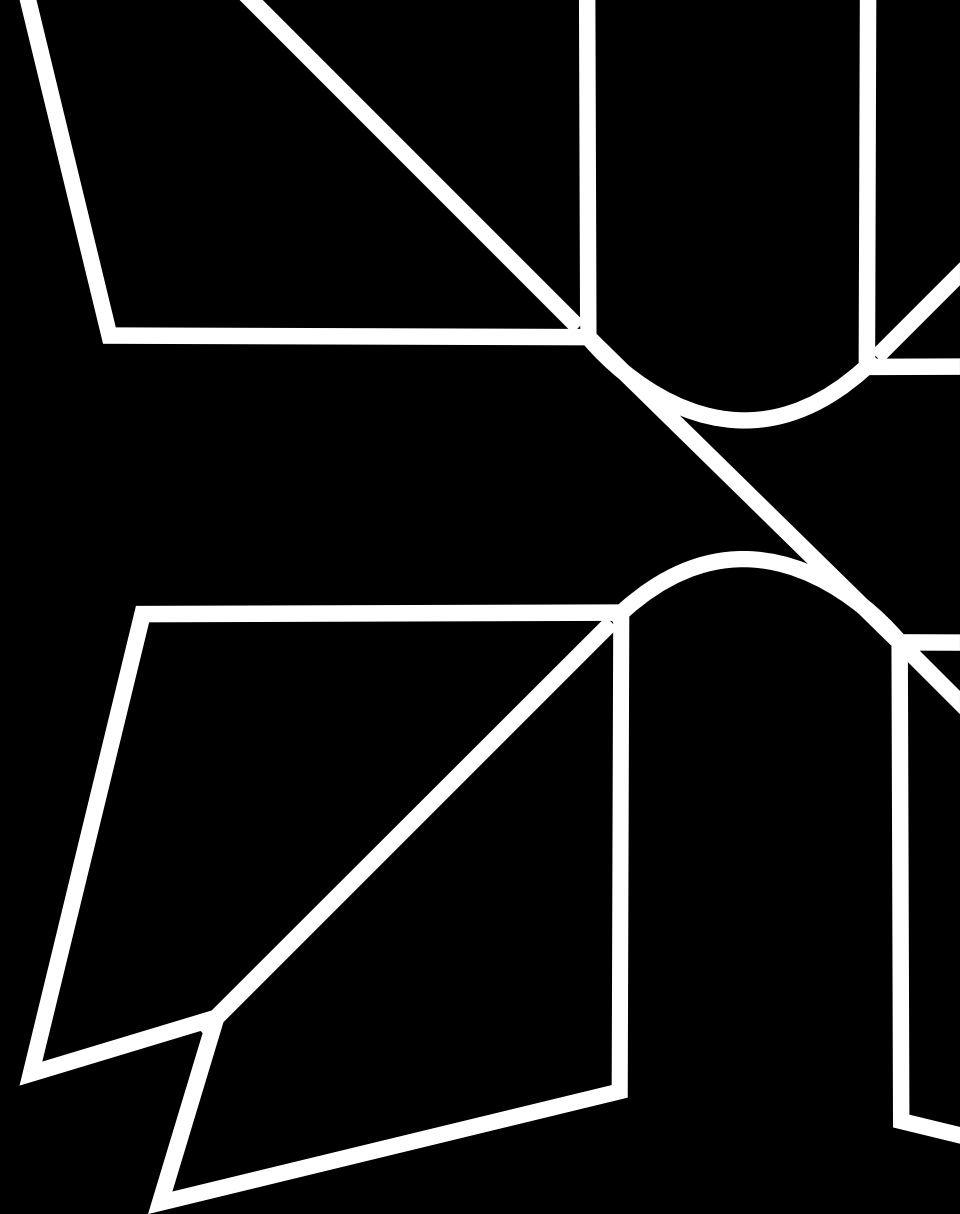
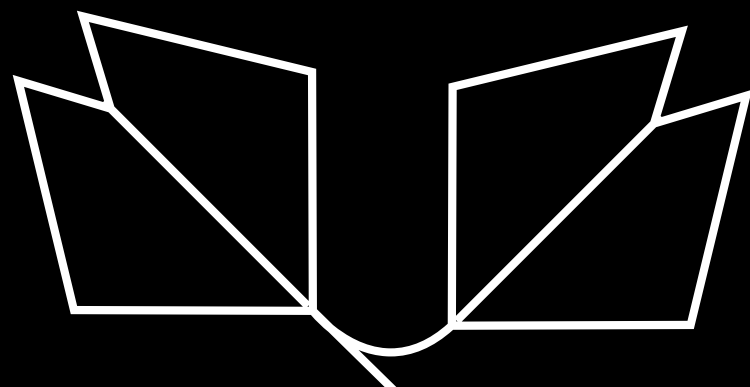
Dal tipo di target che il malware ha porta a pensare che esso abbia intenzione di sostituire la chiave di registro di gina con quella da lui creata così da causare problemi a livello di autenticazione.

Questa sembra essere la funzione del malware da ciò che è stato visto. comportamento associabile ad un malware di tipo Trojan



NETRAIDERS

GIORNO 5

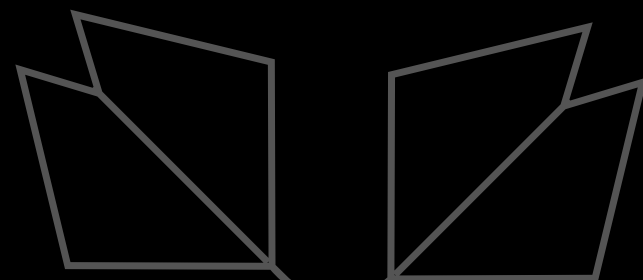


GINA (Graphical identification and authentication) è un componente lecito di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica – ovvero permette agli utenti di inserire username e password nel classico riquadro Windows

- **Cosa può succedere se il file. di 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.



GIORNO 5

```

.data:100032A0 aErrorCodeErro: ; DATA XREF: sub_10001570+9C↑to
Program control flow: A0 unicode 0, <ErrorCode:%d ErrorMessage:%s. >
A0 dw 0Ah, 0
.data:100032E0 ; wchar_t Format
.data:100032E0 Format: ; DATA XREF: sub_10001570+61↑to
* .data:100032E0 unicode 0, <%s %s - %s >,0
.data:100032F8 ; wchar_t Filename
.data:100032F8 Filename: ; DATA XREF: sub_10001570+28↑to
* .data:100032F8 unicode 0, <msutil32.sys>,0
* .data:10003312 db 0
* .data:10003313 db 0
* .data:10003314 db 0

```

A seguito di un controllo, su IDA pro, del DLL creato dal malware si è riscontrata la presenza, all'interno della sezione .data, di quello che sembra un file di sistema.

Andando a cercare la subroutine di riferimento, alla locazione 10001570, ci si rende subito conto che viene aperto un file in modalità scrittura, denominato appunto msutil32.sys

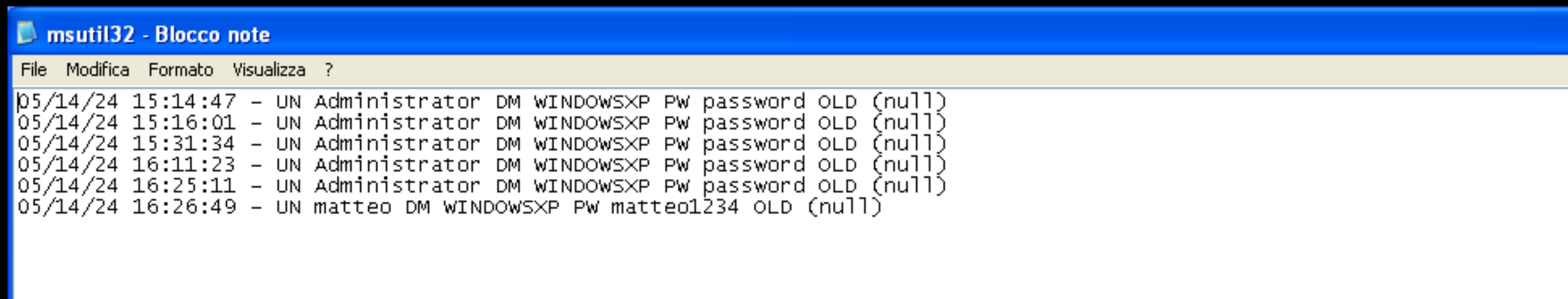
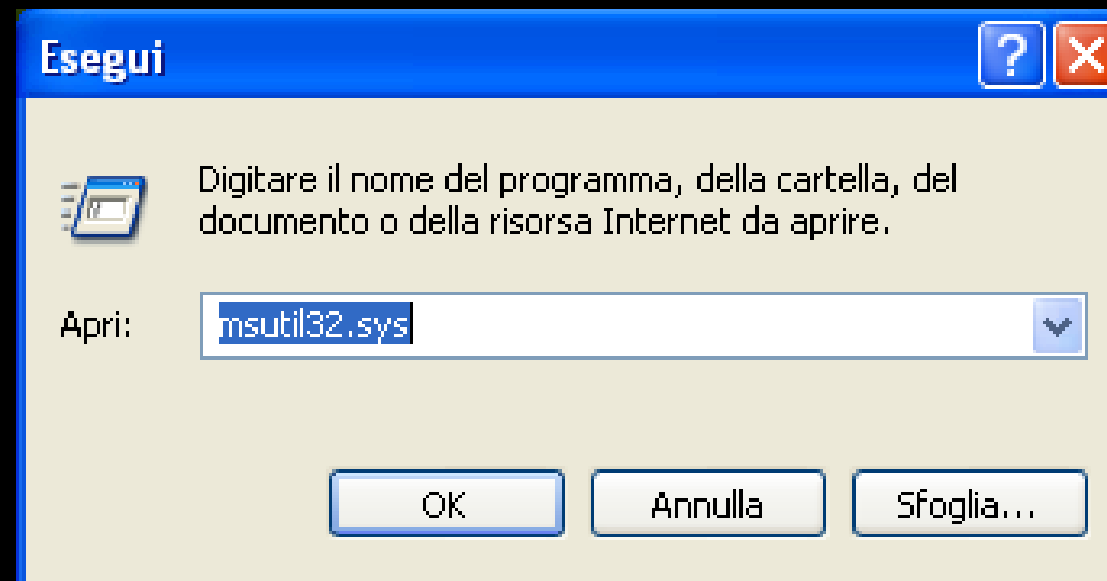
```

mov     ecx, [esp+Format]
sub     esp, 854h
lea     eax, [esp+854h+Args]
lea     edx, [esp+854h+Dest]
push    esi
push    eax                ; Args
push    ecx                ; Format
push    800h               ; Count
push    edx                ; Dest
call    _vsnwprintf
push    offset Mode        ; Mode
push    offset Filename    ; "msutil32.sys"
call    _w fopen
mov     esi, eax
add     esp, 18h
test    esi, esi
jz      loc_1000164F

```

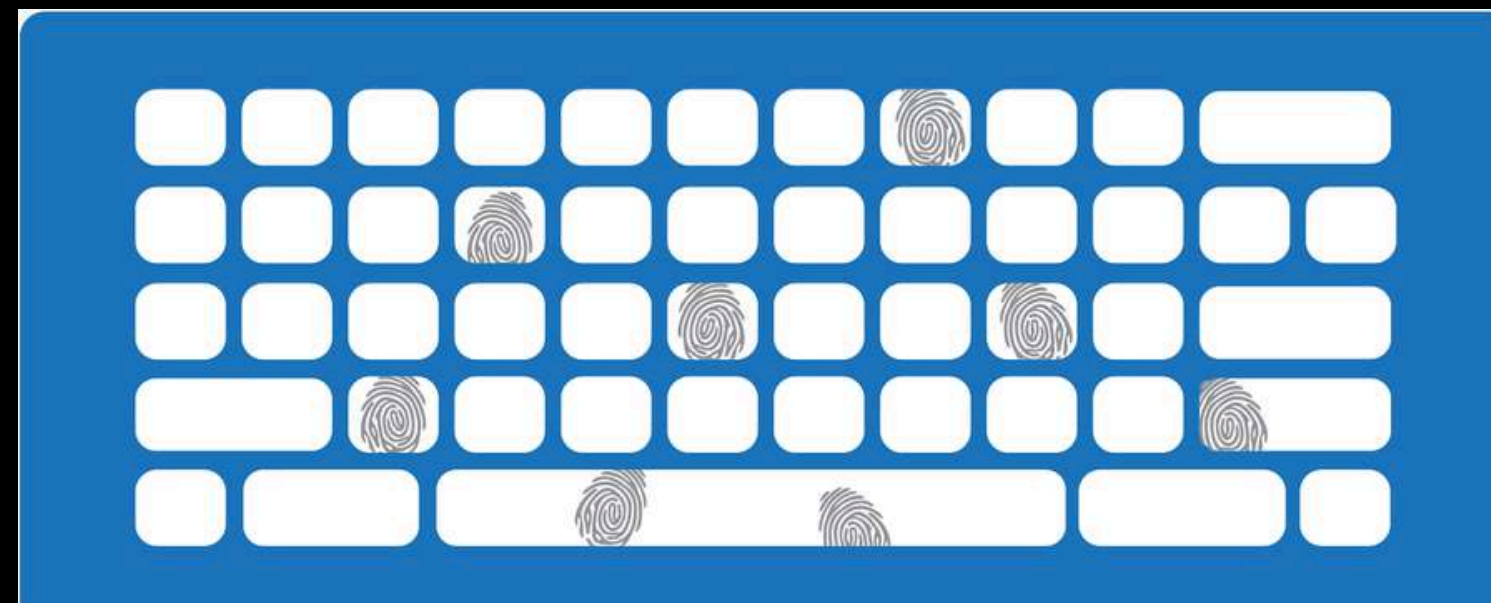

GIORNO 5

Cercando il nome del file su Windows XP, ci siamo imbattuti in un notepad conteneti username e password dell'utente loggato.

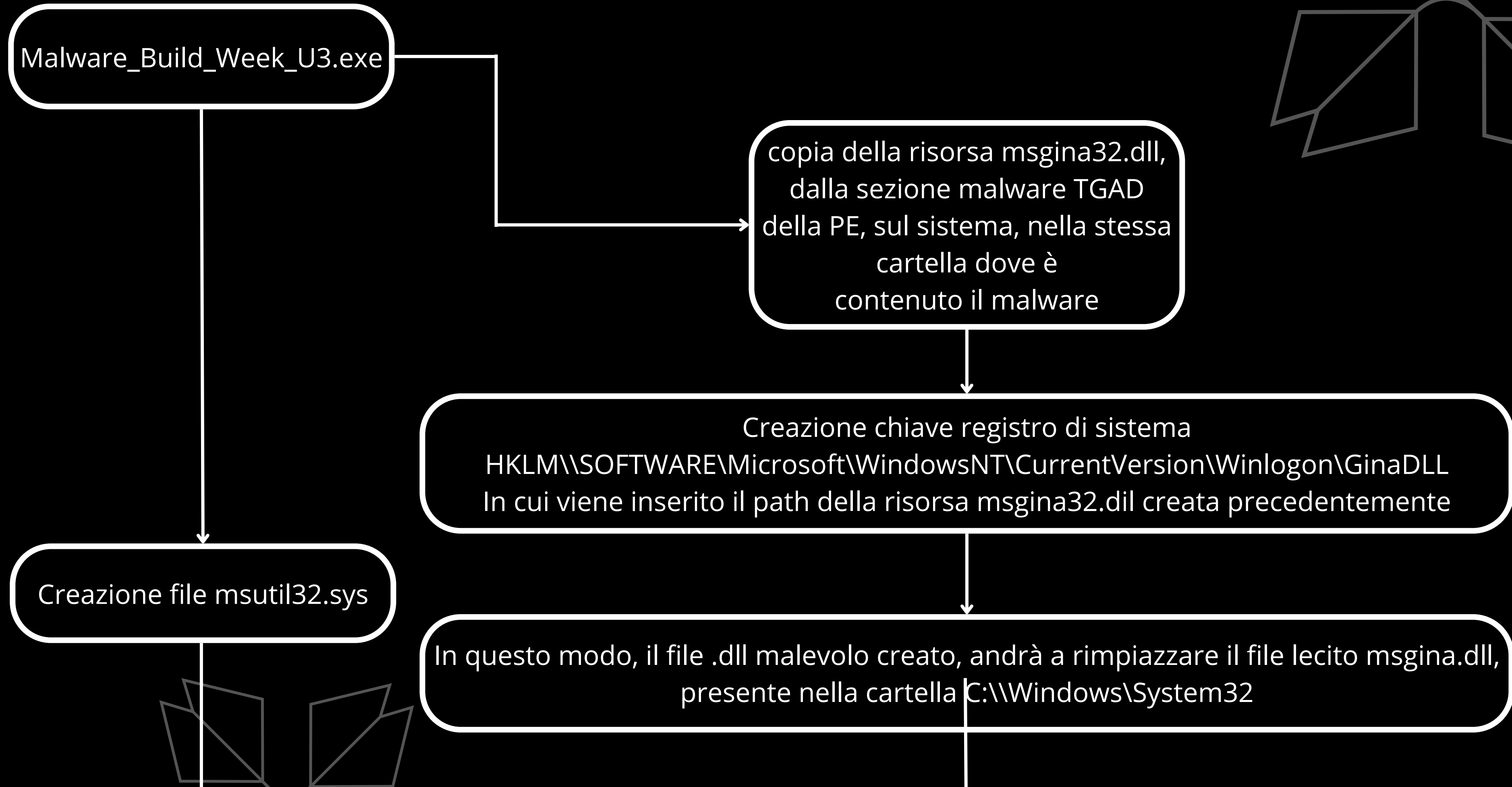


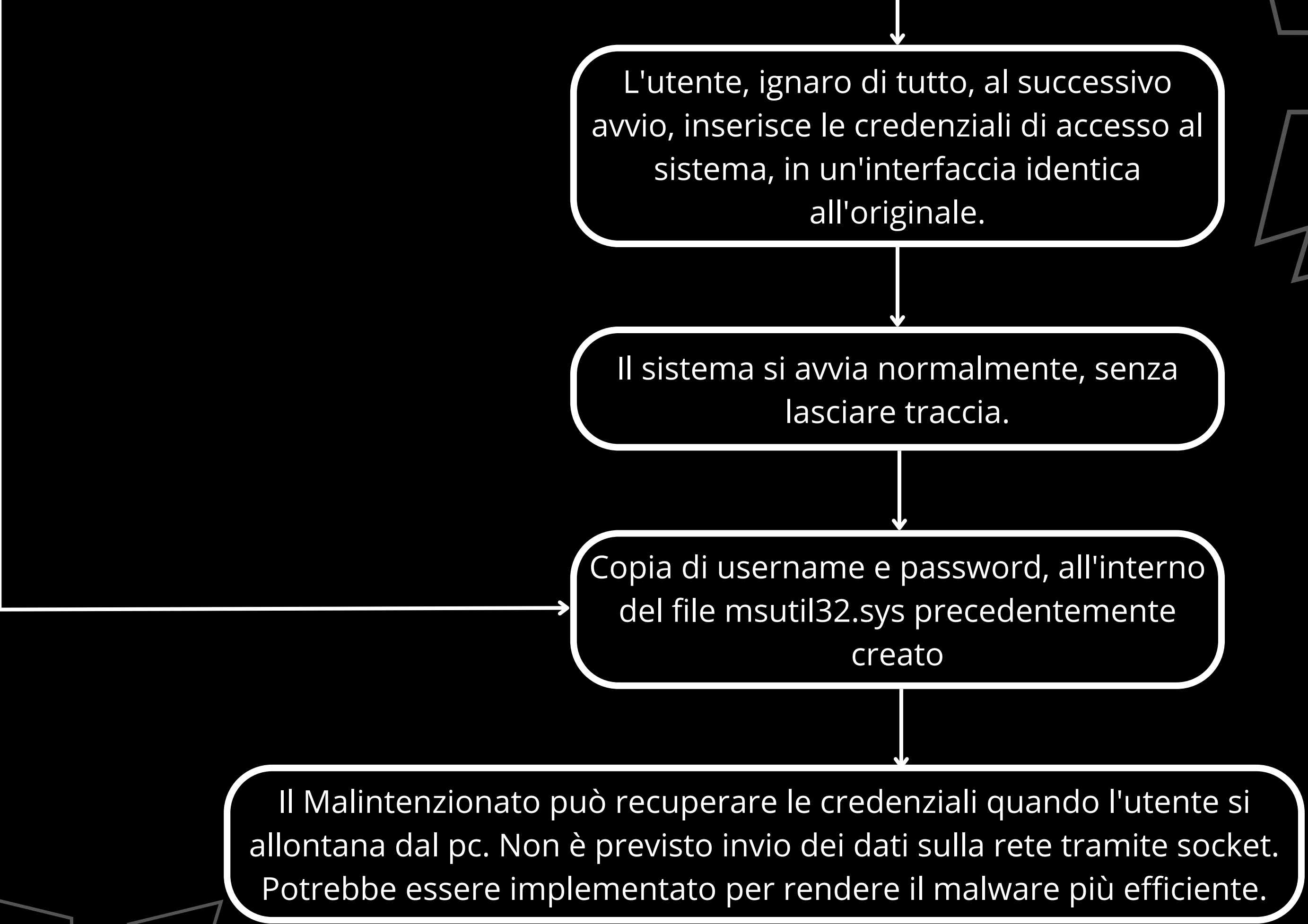


In questo caso il dll lecito di Gina viene rimpiazzato da uno malevolo che intercetta i dati inseriti, per poi copiarli su msutil32.sys



Spiegazione processo Malware





```
graph TD; A[ ] --> B[L'utente, ignaro di tutto, al successivo avvio, inserisce le credenziali di accesso al sistema, in un'interfaccia identica all'originale.]; B --> C[Il sistema si avvia normalmente, senza lasciare traccia.]; C --> D[Copia di username e password, all'interno del file msutil32.sys precedentemente creato]; D --> E[Il Malintenzionato può recuperare le credenziali quando l'utente si allontana dal pc. Non è previsto invio dei dati sulla rete tramite socket. Potrebbe essere implementato per rendere il malware più efficiente.]; D --> A;
```

L'utente, ignaro di tutto, al successivo avvio, inserisce le credenziali di accesso al sistema, in un'interfaccia identica all'originale.

Il sistema si avvia normalmente, senza lasciare traccia.

Copia di username e password, all'interno del file msutil32.sys precedentemente creato

Il Malintenzionato può recuperare le credenziali quando l'utente si allontana dal pc. Non è previsto invio dei dati sulla rete tramite socket. Potrebbe essere implementato per rendere il malware più efficiente.



THANK YOU

**MATTEO LEONI
STEFANO DI PROSPERO
LORENZO MORO
ROSARIO GIAIMO
GIANMARCO MAZZONI**