MALWARE ANALYSIS

BUILD WEEK 3

GIORNO 1

Quesiti

1. Parametri e variabili passati alla funzione MAIN
2. Sezioni all'interno del file eseguibile
3. Librerie importate dal malware
4. Ipotesi

1. PARAMETRI E VARIABILI PASSATI ALLA 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
```

- I **parametri** passati nella funzione Main() sono tre:
  - **argc**= dword ptr 8
  - **argv**=dword ptr 0Ch
  - **envp**= dword ptr 10h
- Le **variabili** dichiarate nella funzione Main() sono quattro:
  - **hModule**= dword ptr -11Ch
  - **Data**=byte ptr -118h
  - **var_8**=dword ptr -8
  - **var_4**= dword ptr -4
- Distinguiamo le variabili dai parametri in quanto le variabili hanno un valore negativo in riferimento ad **EBP** mentre i parametri hanno un valore positivo.

2. SEZIONI ALL'INTERNO DEL FILE ESEGUIBILE

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	0000B000	00000000	00000000	0000	0000	40000040

Le sezioni che compongono il nostro malware sono:

- **.text**: contiene le righe di codice, e quindi le istruzioni, che la CPU andrà ad eseguire una volta che il software verrà avviato.
- **.rdata**: contiene le informazioni delle librerie e le varie funzioni esportate e importate dal malware.
- **.data**: contiene le variabili globali e i dati del malware che devono essere disponibili da qualsiasi parte del programma.
- **.rsrc:** include le risorse utilizzate dal malware come ad esempio icone, immagini, menù e stringhe che non fanno parte del malware stesso.

3. LIBRERIE IMPORTATE DAL MALWARE

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

4. PRIMA IPOTESI: KERNELL32.DLL

Module Name	Imports	OFTs	TimeDateStamp	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

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
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007604	00007604	001B	CloseHandle

KERNEL32.dll è una libreria che contiene le funzioni principali per interagire con il sistema operativo, come la gestione della memoria e la manipolazione dei file.

Ipotesi: Dall'analisi delle funzioni evidenziate, è possibile che il malware in questione sia un dropper.

Quest'ultimo, infatti, è un programma malevolo al cui interno si trova il malware nascosto.

Nel momento in cui viene eseguito, il dropper estrarrà il malware per poi salvare su disco, nello specifico nella sezione .rss dell'eseguibile.

4. SECONDA IPOTESI: ADVAPI.DLL

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
000076D0	N/A	00007500	00007504	00007508	0000750C	00007510
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000076AC	000076AC	0186	RegSetValueExA
000076BE	000076BE	015F	RegCreateKeyExA

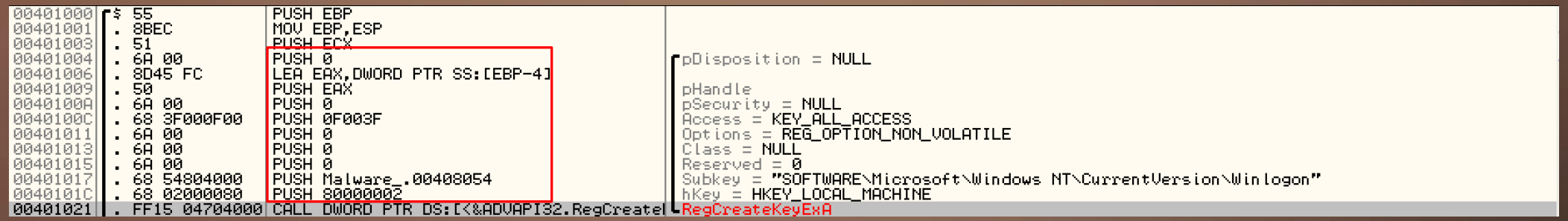
ADVAPI32.dll è una libreria che contiene le funzioni utilizzate per interagire con i servizi e registri del sistema operativo.

Ipotesi: Il malware, tramite la creazione e l'impostazione del registro, ottiene la persistenza.

GIORNO 2

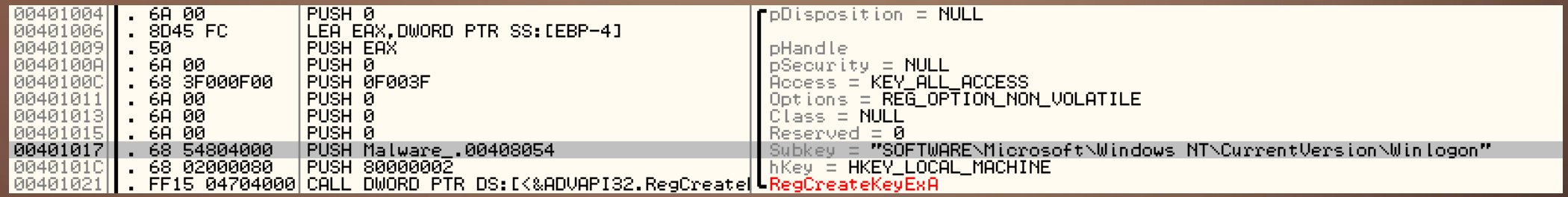
1. Scopo della funzione chiamata alla locazione di memoria 00401021 e come vengono passati i parametri alla funzione.
2. Quale oggetto rappresenta il parametro alla locazione 00401017.
3. Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.
4. Traduzione del punto 3 in C.
5. Valore del parametro «value name» della chiamata alla locazione 00401047.

1. SCOPO DELLA FUNZIONE CHIAMATA ALLA LOCAZIONE DI MEMORIA 00401021 E COME VENGONO PASSATI I PARAMETRI ALLA FUNZIONE.



Lo scopo della funzione è la creazione di una chiave specifica di registro e i parametri vengono passati tramite l'istruzione push.

2. QUALE OGGETTO RAPPRESENTA IL PARAMETRO ALLA LOCAZIONE 00401017.



```
00401004  . 6A 00          PUSH 0                                  pDisposition = NULL
00401006  . 8D45 FC        LEA EAX,DWORD PTR SS:[EBP-4]
00401009  . 50             PUSH EAX                                pHandle
0040100A  . 6A 00          PUSH 0                                  pSecurity = NULL
0040100C  . 68 3F000F00    PUSH 0F003F                             Access = KEY_ALL_ACCESS
00401011  . 6A 00          PUSH 0                                  Options = REG_OPTION_NON_VOLATILE
00401013  . 6A 00          PUSH 0                                  Class = NULL
00401015  . 6A 00          PUSH 0                                  Reserved = 0
00401017  . 68 54804000    PUSH Malware_.00408054                  Subkey = "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
0040101C  . 68 02000080    PUSH 80000002                           hKey = HKEY_LOCAL_MACHINE
00401021  . FF15 04704000  CALL DWORD PTR DS:[<&ADVAPI32.RegCreate RegCreateKeyExA
```

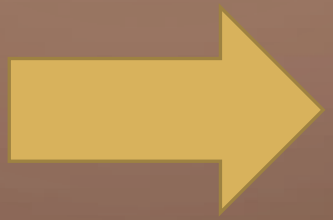
L'oggetto rappresentato dal parametro alla locazione 00401017 è una chiave, nello specifico quella del processo WinLogon che controlla la HKEY_CURRENT_USER. Ma, essendo un Windows Propriety Software, i suoi valori di registro sono allocati nella HKEY_LOCAL_MACHINE. Perciò questo potrebbe essere il primo passaggio compiuto dal malware per ottenere la persistenza.

3. IL SIGNIFICATO DELLE ISTRUZIONI COMPRESE TRA GLI INDIRIZZI 00401027 E 00401029

L'istruzione test, il cui comportamento è simile all'operatore AND, ottiene un risultato in base al quale imposterà lo ZF a 1 o 0.
In questo caso, il risultato è 0, dal momento che il contenuto di EAX è pari a 0, perciò lo ZF sarà settato a 1.
Basandosi sul valore dello ZF precedente, il JE effettuerà il salto o meno se i due operandi saranno uguali. Nel nostro caso il salto verrà effettuato.

4. TRADUZIONE DEL PUNTO 3 IN C

```
Registers (FPU)
EAX 00000000
ECX 00000104
EDX 00140608
EBX 0000000E
ESP 0012FE40
EBP 0012FE48
ESI 004080B2 Malware_.004080B2
EDI 0012FEB7 ASCII "_Week_U3.exe"
```



```
6
7       if(eax==eax){
8
9           int ecx = 260;
10
11      }
```

5. VALORE DEL PARAMETRO «VALUE NAME» DELLA CHIAMATA ALLA LOCAZIONE 00401047

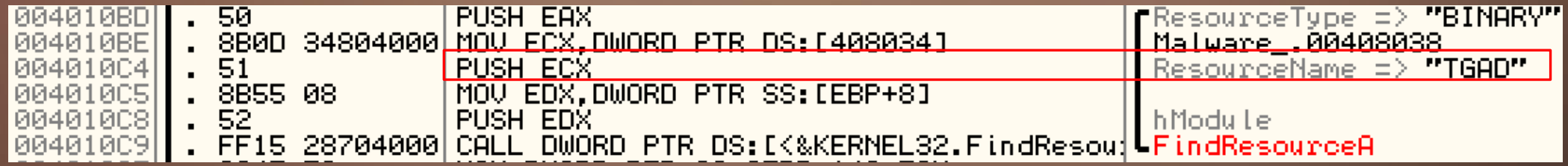
```
.text:0040103C                push    0                       ; Reserved
.text:0040103E                push    offset ValueName ; "GinaDLL"
.text:00401043                mov     eax, [ebp+hObject]
.text:00401046                push    eax                     ; hKey
.text:00401047                call    ds:RegSetValueExA
```

GINA (Graphical Identification and Authentication) è una libreria DLL che viene caricata da Winlogon durante il processo di avvio. Una DLL GINA fornisce procedure personalizzabili di identificazione e autenticazione dell'utente.
Tramite questa libreria software, supponiamo che il malware tenti di ottenere la persistenza.

GIORNO 3
Analisi della sezione di codice tra 00401080 e 00401128

1. Valore del parametro «ResourceName» passato alla funzione «FindResourceA()».
2. Funzionalità implementate dal malware nella sezione di codice.
3. È possibile identificare questa funzionalità utilizzando l'analisi statica basica? In caso di risposta affermativa, elencare le evidenze a supporto.
4. Diagramma di flusso

1. VALORE DEL PARAMETRO «RESOURCENAME» PASSATO ALLA FUNZIONE «FINDRESOURCEA()».



Il valore del parametro «ResourceName» equivale a «TGAD», che poi viene passato alla funzione «FindResourceA».

2. FUNZIONALITÀ IMPLEMENTATE DAL MALWARE NELLA SEZIONE DI CODICE

Le funzionalità implementate dal malware nella sezione di codice presa in esame sono:

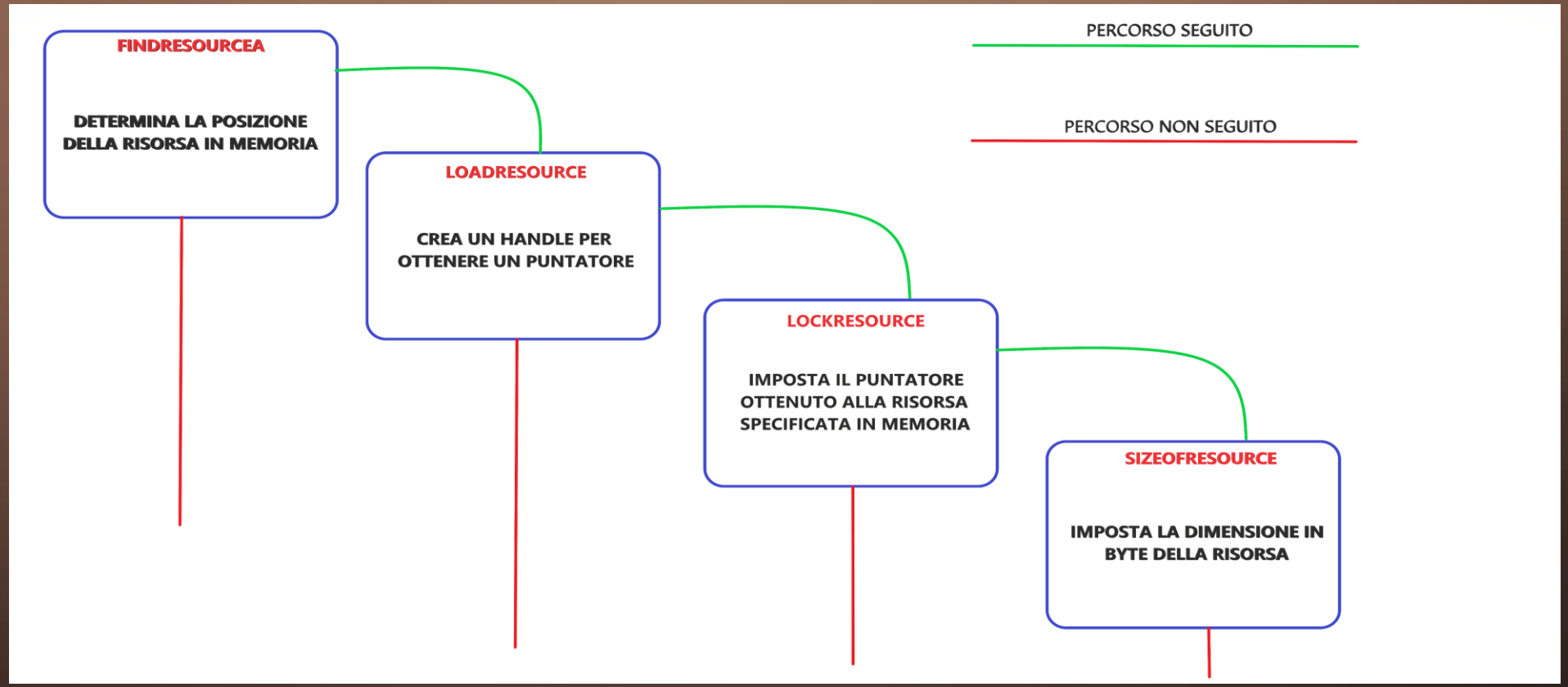
- «FindResourceA»: determina la posizione di una risorsa con il tipo e il nome specificati nel modulo specificato.
- «LoadResource»: imposta un handle che può essere utilizzato per ottenere un puntatore al primo byte della risorsa specificata in memoria.
- «LockResource»: imposta un puntatore alla risorsa specificata nella memoria.
- «SizeofResource»: imposta la dimensione, in byte, della risorsa specificata.

3. È POSSIBILE IDENTIFICARE QUESTA FUNZIONALITÀ UTILIZZANDO L'ANALISI STATICA BASICA? IN CASO DI RISPOSTA AFFERMATIVA, ELENCARE LE EVIDENZE A SUPPORTO

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
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007604	00007604	001B	CloseHandle

Abbiamo constatato la possibilità di identificare questa funzionalità utilizzando l'analisi statica basica, tramite il programma CFF Explorer.
All'interno della cartella «Import Directory» si trovano le librerie utilizzate dal malware.
Di conseguenza, all'interno della libreria «Kernel32.dll» sono contenute le funzioni evidenziate nell'immagine a sinistra.

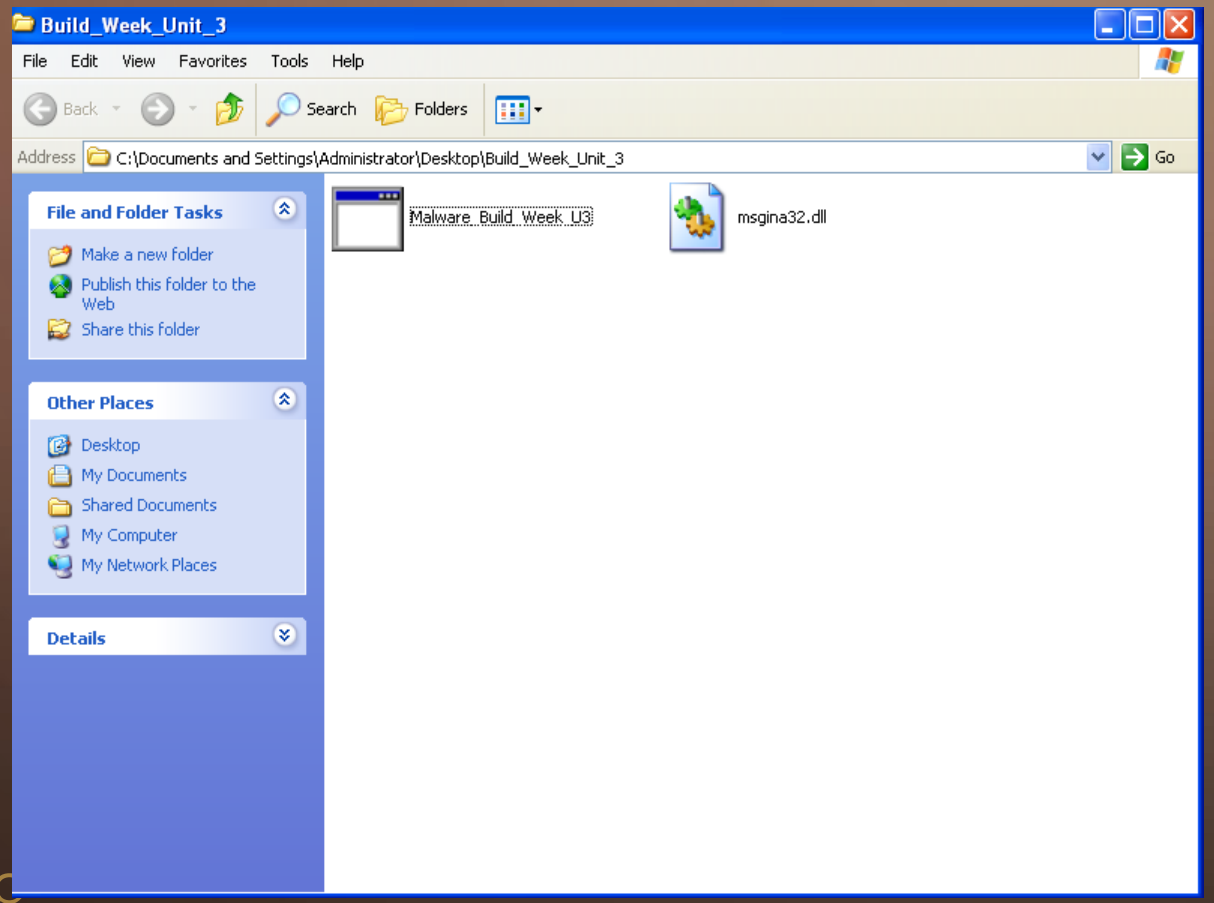
4. DIAGRAMMA DI FLUSSO



GIORNO 4

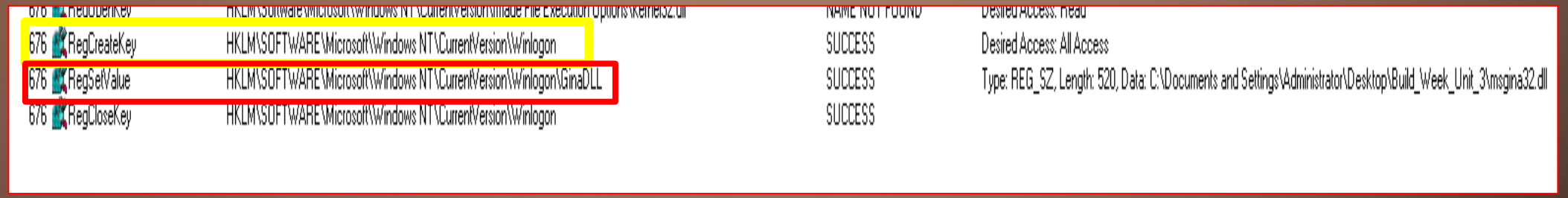
1. Differenze notate all'interno della cartella del malware.
2. Chiave di registro creata e valore associato alla chiave di registro creata.
3. Chiamata di sistema che modifica il contenuto della cartella dove è presente l'eseguibile del malware.
4. Ipotesi di funzionamento del malware (analisi statica e dinamica).

1. DIFFERENZE NOTATE ALL'INTERNO DELLA CARTELLA DEL MALWARE



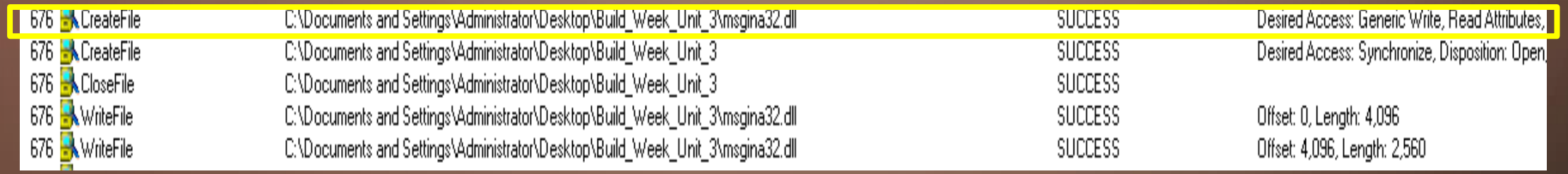
Una volta avviato il «Malware_Build_Week_U3», contenuto all'interno della cartella «Build_Week_Unit3», possiamo notare una prima differenza proprio all'interno della cartella stessa.
Tenendo presente l'immagine a sinistra, vediamo che il malware carica all'interno della cartella la libreria «msgina.dll».

2. CHIAVE DI REGISTRO CREATA E VALORE ASSOCIATO ALLA CHIAVE DI REGISTRO CREATA (PROCMON)



Dopo l'avvio del malware, possiamo identificare i processi creati da questo attraverso il programma «Process Monitor».
All'interno del riquadro giallo, vediamo la creazione della chiave di registro attraverso il processo «RegCreateKey» nel path «HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\WinLogon» richiedendo tutti gli accessi.
All'interno del riquadro rosso, invece, possiamo identificare il valore associato alla chiave di registro, ovvero «GinaDLL» che viene configurato con il processo «RegSetValue».

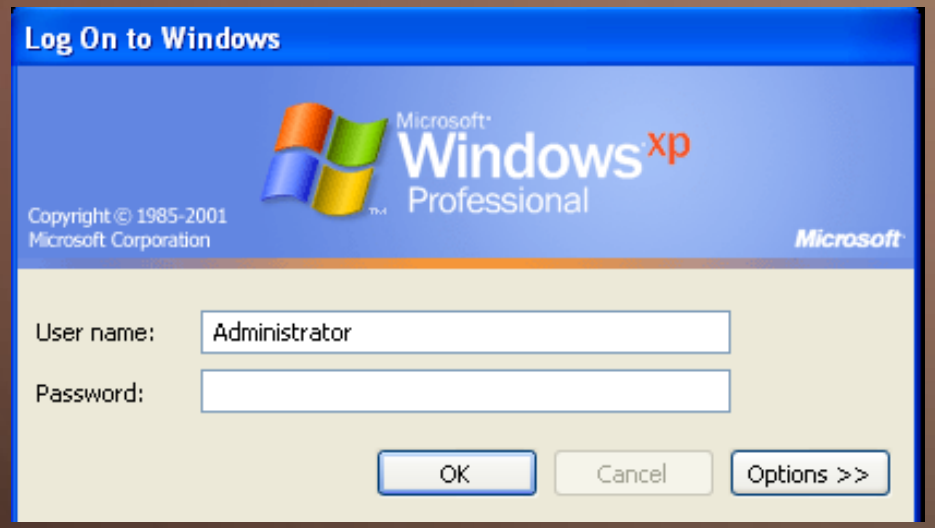
3. CHIAMATA DI SISTEMA CHE MODIFICA IL CONTENUTO DELLA CARTELLA DOVE È PRESENTE L'ESEGUIBILE DEL MALWARE



676	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	Desired Access: Generic Write, Read Attributes,
676	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	Desired Access: Synchronize, Disposition: Open,
676	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	
676	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	Offset: 0, Length: 4,096
676	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	Offset: 4,096, Length: 2,560

Filtrando, all'interno di Process Monitor, le attività di file System possiamo individuare la chiamata di sistema che permette di modificare il contenuto della cartella del malware.
La suddetta chiamata è evidenziata dal riquadro giallo in figura.

4. IPOTESI DI FUNZIONAMENTO DEL MALWARE (ANALISI STATICA E DINAMICA)



Analizzando il malware sia dal punto di vista statico che dinamico, possiamo ipotizzare che il funzionamento del malware sia quello di ottenere la persistenza e di eseguire dei passaggi comuni al dropper per poter mostrare a schermo, nella fase di avvio della macchina, l'interfaccia grafica di autenticazione GINA.

Quest'ultimo passaggio è reso possibile dall'inserimento e caricamento della libreria «msgina32.dll».

GIORNO 5

1. Sostituzione del file gina.dll lecito in file .dll malevolo che intercetta i dati inseriti
2. Grafico

1. SOSTITUZIONE DEL FILE GINA.DLL LECITO IN FILE .DLL MALEVOLO CHE INTERCETTA I DATI INSERITI

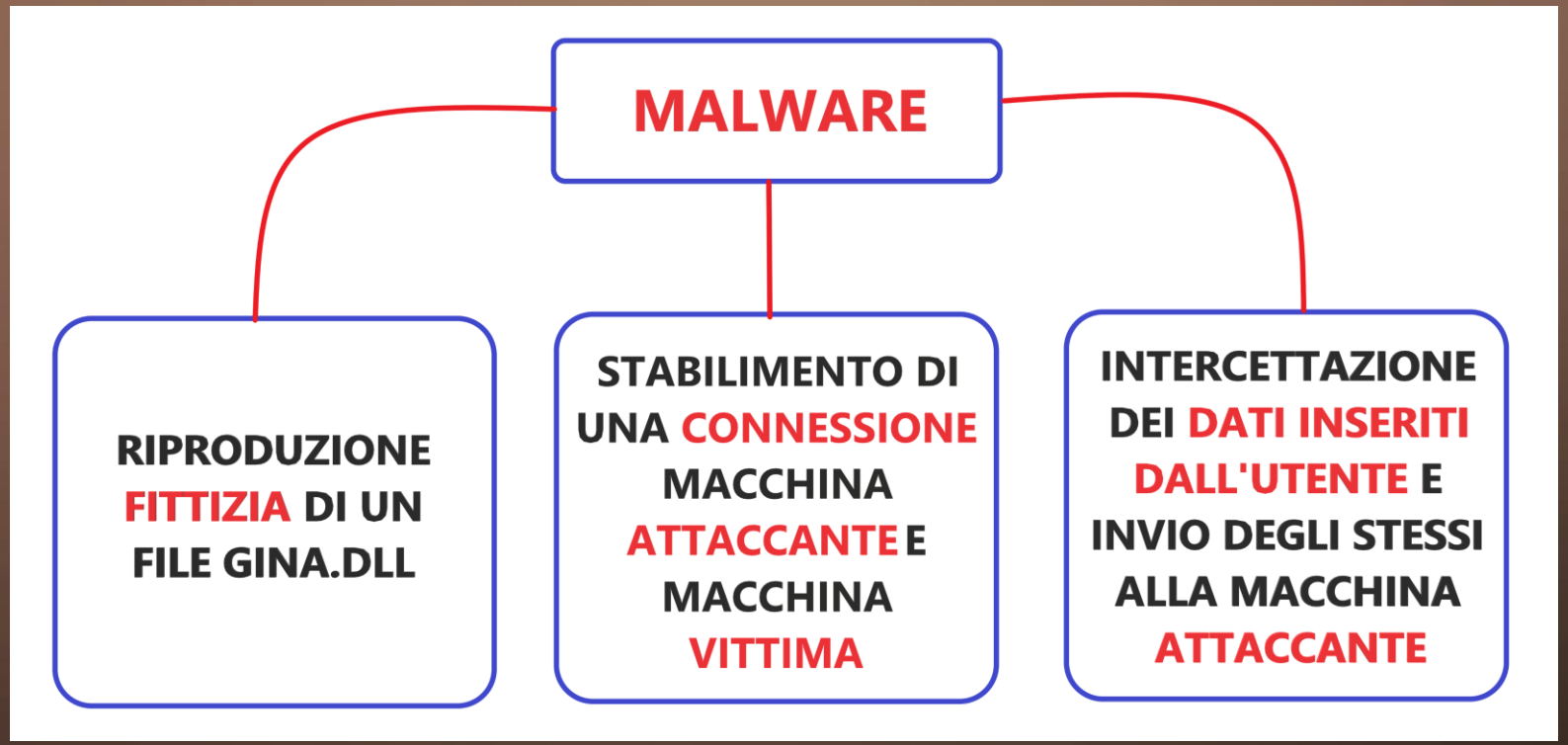
Nell'ipotesi in cui il malware stia cercando di intercettare i dati, possiamo supporre che il .dll malevolo, inserito al posto del .dll lecito, vada a riprodurre l'interfaccia grafica di GINA per poi intercettare i dati inseriti all'interno dei campi «username» e «password».

Possiamo ipotizzare che il funzionamento di tale malware sia quello di salvare i dati inseriti nei campi per poi inviarli alla macchina attaccante.

Naturalmente, la macchina attaccante ha bisogno di stabilire una connessione con la macchina target, al fine di poter ricevere le informazioni rubate alla macchina vittima.

In base a quanto detto sopra, possiamo categorizzare il malware come spyware, il quale può essere accoppiato ad un keylogger.

2. GRAFICO


MALWARE
RIPRODUZIONE FITTIZIA DI UN FILE GINA.DLL
STABILIMENTO DI UNA CONNESSIONE MACCHINA ATTACCANTE E MACCHINA VITTIMA
INTERCETTAZIONE DEI DATI INSERITI DALL'UTENTE E INVIO DEGLI STESSI ALLA MACCHINA ATTACCANTE