REPORT 19/01/2023 **LORENZO SCORBAIOLI**

TASK:

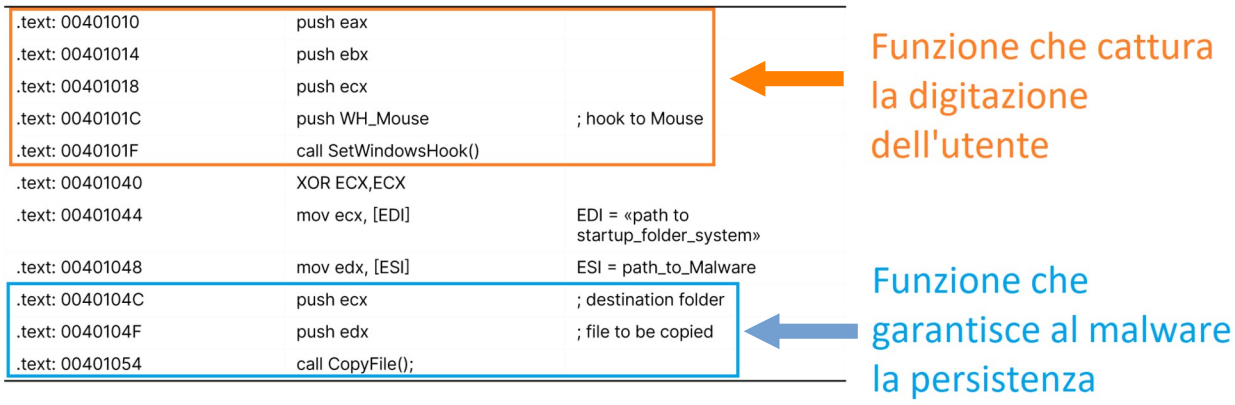
La seguente figura mostra un estratto del codice di un malware:

.text: 00401010	push eax	
.text: 00401014	push ebx	
.text: 00401018	push ecx	
.text: 0040101C	push WH_Mouse	; hook to Mouse
.text: 0040101F	call SetWindowsHook()	
.text: 00401040	XOR ECX,ECX	
.text: 00401044	mov ecx, [EDI]	EDI = «path to startup_folder_system»
.text: 00401048	mov edx, [ESI]	ESI = path_to_Malware
.text: 0040104C	push ecx	; destination folder
.text: 0040104F	push edx	; file to be copied
.text: 00401054	call CopyFile();	

Identificare:

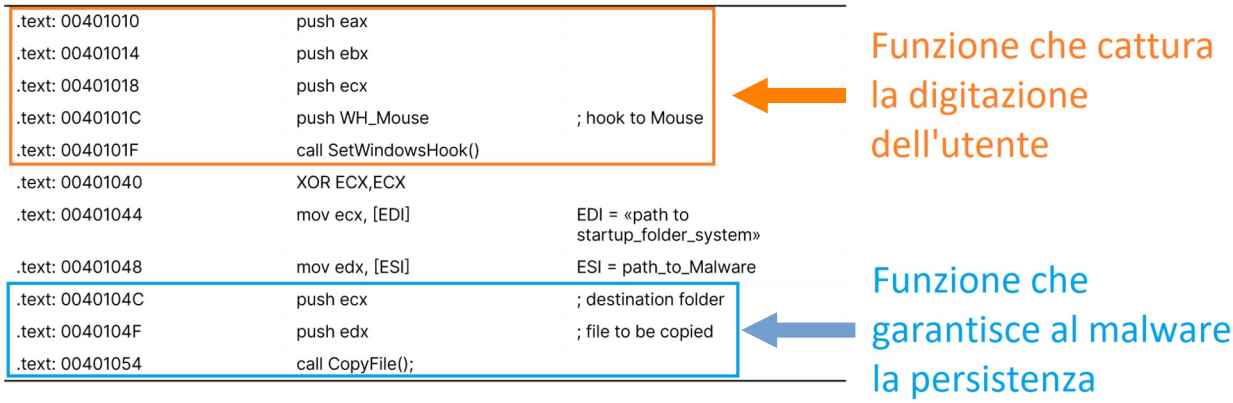
1. Il tipo di malware in base alle chiamate di funzione utilizzate
2. Evidenziare le chiamate di funzione principali aggiungendo una descrizione per ognuna
3. Il metodo utilizzato dal malware per ottenere la persistenza sul sistema operativo
4. BONUS: effettuare un'analisi di basso livello delle singole istruzioni

1)



Come possiamo vedere dall'immagine qui sopra, in questo estratto di codice ci sono 2 chiamate di funzione. La chiamata di funzione nel riquadro arancione è la funzione di chiamata principale ed è quella che si occupa di hook consente di monitorare i messaggi del mouse. Da questo possiamo capire quindi che il malware è un keylogger che si occupa di registrare i tasti del mouse della macchina infetta che vengono premuti. Successivamente con molta probabilità verrà creato un file in cui verranno registrate queste informazioni.

2)



Le due chiamate di funzione sono:

- call SetWindowsHook()[1]:

Installa una routine hook definita dall'applicazione in una catena di hook. Si installerebbe una procedura hook per monitorare il sistema per determinati tipi di eventi. Questi eventi sono associati a un thread specifico o a tutti i thread nello stesso desktop del thread chiamante. Secondo quanto riportato dal sito di Microsoft, questa funzione necessita di 4 parametri:

1) [in] idHook: indica il tipo di procedura hook da installare. Nel nostro codice è WH_Mouse
2) [in] lpfn: puntatore alla routine hook
3) [in] hmod: handle per la DLL contenente la routine hook a cui punta il parametro lpfn
4) [in] dwThreadId: identificatore del thread a cui deve essere associata la routine di hook

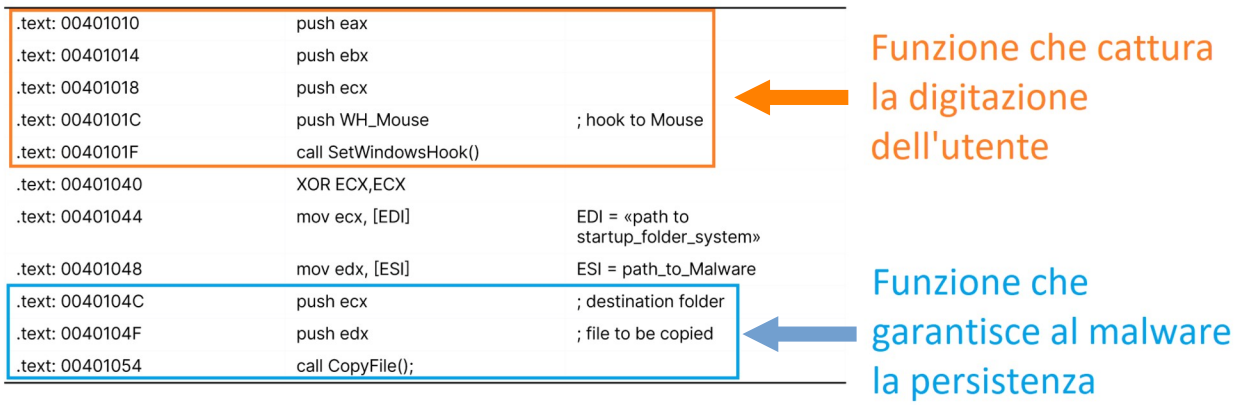
- call CopyFile()[2]:

Copia un file esistente in un nuovo file. Secondo quanto riportato dal sito Microsoft, questa funzione richiede i seguenti parametri:

1)[in] lpExistingFileName: indica il nome di un file esistente
2)[in] lpNewFileName: nome del nuovo file
3)[bFailIfExists: se questo parametro è TRUE e il nuovo file esiste già la funzione ha esito negativo. In caso contrario ha esito positivo.

1 https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setwindowshookexa
2 https://learn.microsoft.com/it-it/windows/win32/api/winbase/nf-winbase-copyfilea

3)



In aggiunta alla funzione di cui ho parlato nel punto 1, abbiamo anche una seconda chiamata di funzione, quella all'interno della cornice blu. Questa seconda funzione invece ha il compito di fornire la persistenza al malware, andando a copiare il suo eseguibile nella cartella di startup, in modo che esso venga eseguito all'avvio del sistema.

4)

RIGA DI CODICE	SIGNIFICATO
Push eax	Pusha il registro eax sulla cima dello stack
Push ebx	Pusha il registro ebx sulla cima dello stack
Push ecx	Pusha il registro ecx sulla cima dello stack
Push WH_Mouse	Pusha l'hook in cima allo stack
Call SetWindowsHook()	Chiama la funzione SetWindowsHook()
XOR ecx,ecx	Viene azzerato il registro ecx con l'operatore XOR
Mov ecx,[edi]	Sposta il contenuto dell'indirizzo di memoria del registro edi (che contiene il percorso della cartella di startup del sistema) nel registro ecx
Mov edx,[esi]	Sposta il contenuto dell'indirizzo di memoria del registro esi (che contiene il path del malware) nel registro edx
Push ecx	Pusha il registro ecx sulla cima dello stack
Push edx	Pusha il registro edx sulla cima dello stack
Call CopyFile()	Chiama la funzione CopyFile()