

BYTE REBELS



UDP FLOOD

BYTE REBELS

CANNAVACCIUOLO DAVIDE
DI MAIO PAOLO
FORLENZA SIMONE
RUSSO FEDERICO - LEADER
TIZZI FEDERICO
VAN ZWAM ARJEN

L'esercizio di oggi è scrivere un programma in **Python** che simuli un **UDP flood**, ovvero l'invio massivo di richieste UDP verso una macchina target che è in ascolto su una **porta UDP casuale**.

```

kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~/Documents/Programmazione_Epicode/Python]
$ sudo python Test.py

BYTE REBELS

L'esercizio di oggi consiste nel scrivere un programma in Python che simuli un
flood UDP, ovvero l'invio massivo di richieste UDP verso una macchina target
che è in ascolto su una porta UDP casuale.
Inserisci l'indirizzo IP del malcapitato da scansire: 10.0.2.15
Inserisci il port range da scansire (es: 1-100,201-300): 1230-1240
Porta 1230 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1231 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1232 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1233 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1234 su 10.0.2.15 Ã" APERTA
Porta 1235 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1236 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1237 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1238 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1239 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1240 su 10.0.2.15 Ã" CHIUSA o non risponde
Inserisci indirizzo: 

```

UDP FLOOD CLIENT .PY

Abbiamo progettato questo programma con due funzionalità principali, port scanner e UDP Packet Sender.

Port Scanner: Consente di eseguire una scansione delle porte su un determinato indirizzo IP per determinare quali porte sono aperte e quali sono chiuse o non rispondono. L'utente può specificare un range di porte da scansionare e l'indirizzo IP del target. Il programma utilizza thread per eseguire la scansione in modo più efficiente. Poi si passerà alla funzione vera e propria del progetto, il flood.

Il programma utilizza il modulo socket di Python per comunicare tramite socket, nonché il modulo threading per gestire i thread per le operazioni concorrenti.

```
Inserisci il port range da scansire (es: 1-100,201-300): 1230-1240
Porta 1230 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1231 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1232 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1233 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1234 su 10.0.2.15 Ã" APERTA
Porta 1235 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1236 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1237 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1238 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1239 su 10.0.2.15 Ã" CHIUSA o non risponde
Porta 1240 su 10.0.2.15 Ã" CHIUSA o non risponde
```

PORT 03 SCANNER

A cosa serve?

Consente di eseguire una scansione delle porte su un determinato indirizzo **IP** per determinare quali porte sono aperte e quali sono chiuse o non rispondono.

L'utente, inoltre, può specificare un range di porte da scansionare e l'indirizzo **IP** del target. Il programma utilizza *thread* per eseguire la scansione in modo più efficiente.

Utilizzo

All'utente viene chiesto di inserire l'indirizzo IP del target da scansire. • Successivamente, l'utente inserisce il range di porte da scansionare, specificando la porta minima e la porta massima desiderata separate da un trattino. Ad esempio: 1-100.

Il programma esegue la scansione delle porte e stampa i risultati in ordine numerico, comunicando un feedback.

Scopo

Trovare le porte aperte della vittima

UDP PACKET SENDER

```
Il pacchetto 13 viene inviato da Thread-7
Il pacchetto 14 viene inviato da Thread-7
Il pacchetto 15 viene inviato da Thread-7
Il pacchetto 16 viene inviato da Thread-7
Il pacchetto 17 viene inviato da Thread-7
Il pacchetto 18 viene inviato da Thread-7
Il pacchetto 19 viene inviato da Thread-2
Il pacchetto 20 viene inviato da Thread-2
Thread-2 Terminato.
Il pacchetto 17 viene inviato da Thread-5
Il pacchetto 14 viene inviato da Thread-3
Il pacchetto 13 viene inviato da Thread-8
Il pacchetto 19 viene inviato da Thread-7
Il pacchetto 20 viene inviato da Thread-7
Il pacchetto 13 viene inviato da Thread-6
Il pacchetto 14 viene inviato da Thread-6
Il pacchetto 15 viene inviato da Thread-6
Il pacchetto 16 viene inviato da Thread-6
Il pacchetto 17 viene inviato da Thread-6
Il pacchetto 18 viene inviato da Thread-6
Il pacchetto 19 viene inviato da Thread-6
Il pacchetto 20 viene inviato da Thread-6
```

A cosa serve?

Permette di inviare un numero *specificato* di **pacchetti UDP** ad un determinato indirizzo IP e porta associata. L'utente può specificare l'indirizzo IP di destinazione (*vittima*), la porta ed il numero di pacchetti da inviare. Anche questa funzionalità utilizza *thread* per inviare i pacchetti in parallelo.

Utilizzo

Dopo aver individuato la porta tramite il port scanner all'utilizzatore viene chiesto di inserire l'indirizzo IP della vittima per l'invio dei **pacchetti UDP**.

Quindi, l'utente inserisce il numero di porta e il numero di pacchetti UDP da inviare.

Il programma avvia un certo numero di *threads* per inviare i pacchetti UDP in **parallelo**.

Scopo

Intasare di pacchetti fittizi l'**IP** target

```
(kali@kali) - [~/Documents/Programmazione_Epicode/Python]
$ sudo python Test.py
```

BYTE REBELS

L'esercizio di oggi consiste nel scrivere un programma in Python che simuli un flood UDP, ovvero l'invio massivo di richieste UDP verso una macchina target che è in ascolto su una porta UDP casuale.

```
Inserisci l'indirizzo IP del malcapitato da scansire: 10.0.2.15
Inserisci il port range da scansire (es: 1-100,201-300): 1230-1240
Porta 1230 su 10.0.2.15 A CHIUSA o non risponde
Porta 1231 su 10.0.2.15 A CHIUSA o non risponde
Porta 1232 su 10.0.2.15 A CHIUSA o non risponde
Porta 1233 su 10.0.2.15 A CHIUSA o non risponde
Porta 1234 su 10.0.2.15 A APERTA
Porta 1235 su 10.0.2.15 A CHIUSA o non risponde
Porta 1236 su 10.0.2.15 A CHIUSA o non risponde
Porta 1237 su 10.0.2.15 A CHIUSA o non risponde
Porta 1238 su 10.0.2.15 A CHIUSA o non risponde
Porta 1239 su 10.0.2.15 A CHIUSA o non risponde
Porta 1240 su 10.0.2.15 A CHIUSA o non risponde
Inserisci indirizzo: █
```

05

ANALISI: Client - Flood.py

1. Il programma inizia chiedendo l'indirizzo IP Target
2. Inseriremo dunque il range di porte che si vogliono scansire
3. Inserire l'indirizzo della vittima
4. Inserire la porta acquisita precedentemente
5. Inserimento della quantità di pacchetti UDP da mandare

```
\xfa\xeb\x1d\x87+\xd3\x10\xe4\xf4\\\x02\x9c+\xf6<\x81\rrv\x81\x9e\x8f8NW\xb02E\x82\x9c\xe2[\x1d\x12Q0(\x83\xb3E\x90R/\xa0\x9
e\xb9\xeeEf\x87\xfd9\x05(\x8a\xea\xbb5<\xa3\x98C\xa8\x1d\x04\xb7p-\xdf:]\x8c\xf1e\x1fY\xdb\xb7\x84\x12uW\xc7e\xb0\\\x0e\xebT\
x8fM\xea\xfb1\xb8\xa4\xce\xc2\x00LQ/\x98\x7f\xc9\x13\xd4\x9e\xcc\x9c\x874'\x02\\\xeb\x05\xab\x1d\x04\xa7f;/l\xdfA\xbbRp\x1d\
xfc\xa1\xc8\x05\x90\xdd\x12\xa8S\xb0\xb5\x9d0\xad\xf3\x96q-\x'\xc3vG\xe3".E\xbb0V\xb0\x97\xb5\xbf\xef\xe0\xe3"\xf3j\xd06\x9f\x
eb\xb4S\x90\xd0L\xd4\xb9-\xfc\xdf\xc2\x13\x81\x9a\xb5\x98\xd50\x8d\x1b\xf2\x1d2\xad\x0e?\xc4\x8722\xd9gq\x8eGR2>\x92w<\xdf\x
cce\xc7\x0b)\xfeV\xd4/\xde\x8f\xaf\xc5\xb6\x15\x16L"
Response sent to ('192.168.1.15', 40073): Response from UDP server
UDP packet received from ('192.168.1.15', 47141): b'\x9eJ-\xa1)\xb6\xf3 \x04g\x1cp\xcf'\x03\x95\x89\x15\xa4#\x9dw(_l\xda\xaa
\xcdG\n\x9f\\\x11\\\x90\xffa\xa5\xf6@\xb8{-\xb9\x0f@\xe0\xe8%\x0b\x03\x03\xc0T\rH#\xf5\xd0+\xc3\xc5\xf6\x11\x92\xca\xa0\xcd
\x9c69\xb9 \xd3\xd4\xdf\x9c8EWcpj\xb7\xc1\xb5\x04\xa8\xaf\xddR<\xa9\x87\xdeR\x8b\x0c2\xc2+0\x02\xd6\x86G/\xe5Cm\xa9*\x95\x94\x
b5\xef2\xc5\xc8\x96\x9fGp\xa7\xe4\xb9\xb4k?h\xcc\x08\xcd\x1e\x9d\xbc\xf81v"\xb9~7\xe99W\xca\x13\x9f\xa8P\x10D600\x19\x9fay6\x
12k\x9f(\xc2L\x00\x04\xfej\x1c\x9c\x9f\x1f5\xa2\xf0\x9d9tE\xf9\xcd3\xca\x93\xb9\x02\xde\xae\xaaQ\x8d\x95\x85\x03ok\xcd\xa7\xcb
qR\x19*\xd3\xf8T@\x8f\x06h\x84@<r\xbb3vE"\xee\xec\x1d1r\xfd\x80_\x93w>\x0erN\xa2\x7f\x8a\xfe\x950\x9c7\x9e9\x9eV\xec\x9e1\x13\x18
L-\xf7\xd26\xae\x0d6[\xa72\xed\x9f8-\xa5\xcc \x13\x9c8c\xb1\x00U\x86\x07\x9c1\x91r'\x9c\xeb\xa5\x13\xb0v\xd0\x9b\xce\x9b9T\x83\x
8b\xbf\xf2\x8e\x0e\xf5\x1c1\x9d9\x89P\n\xa6\xd4\xc5\xf1\xf6Ix\xfbHt\xbfX[\x06\xb33t\x8a-\x19\xb6\x94\x9c\xe98\xb1q\x88\xa8
0j\xafk\x05a\xaf\xcf\x95K\x1cs"\x04e\rn\x80\x8d\xbd\x19\xff\xe5\xbc\xa7\x0b\xb6\xa4\xeb\xee\x18\xb5\xad\xabXI\xd1\xedVa\xff\
xd7\xfc1\xa2\xfeK\xa3@u\xce\x192 \x9d\x93a\x85n\x19\xed\x8f_v\x9e8\x85\n'0A\xf7\xed\x19,\xa0\xe9C0\x85\xdb\x93\xb1\x91\x01\x
1eV^cJ(\xe3\x032\xc1\x1c\x91\x924n"\x91<3\xcc9I\x9b\xd3l\x83\xc2\x13\x86l\xbc\xa2r-\x19%9\x83C\xde\x7f\xff\xd4lAeg\x9e1\x91\x
8baC\x1e\x9e\x8e\x14T\xd7G1\x9f0H\x98\xa9j\xa8\xbe9\x06\x9f9\x9c7\x9c7wNA\x1d\xefc\xf5\x8d\xe4\xf4g\xce\xde\x9f\x95\xa8\x9c2
"\x8e\x9e\x17\x01t\xb5D\x9c|\xb4gn\x90$\x9b\xb4\xf8\x9f\x9c4\xbaA\x94\xf26o\xdc\xe64\xf72\x00\xdf-\xa9\x0c-\xf4\xea\xb6\x87$\
xfc\x90\xa8\x94P\x84r\xaa\x9c\xaa\xbd\x9c5\xbdY\x0b\xaff[Zg":\x9a\x07\xae\x02\xbd\x92""\xbba\xc3\xf9-8\x135\xa9\xb1\xaaL\xb9\x
d1\x00\xe9\xfb7\xb7\x1a\xeeI\x84\xb93j^"R\x06;\xb1\xbbT\x06G\xceT\x7f\xb7\nG\x0e?\xc8\x125\xeb?\x13\x8f\x9d9\x986\xf3,\nZ\x87
\xbbu\xcdV\xfd\x96Ut\x0a\x06\x18n\xdc6\x9c0\x063\x86>\x9a\x0d4\x08\x1a\x19\x03\x9c1\x05ld\x04b/\xb6\x02/\x9e6\x1095\x9d40\x00
```

ANALISI: Server.py

Abbiamo semplicemente creato un mini server si mette in ascolto UDP nella sua porta(abbiamo ipotizzato 1234).

PILLOLE DI SCRIPT

IMPORT

Abbiamo importato le varie librerie per il corretto svolgimento.

DEF FUNZIONI

Abbiamo dunque definito le funzioni `send_udp_packets`, la def `port_scan_range`, `def_port_scanner` e ovviamente la `main`

IMPORTANTE

`s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)`
`SOCK_`**DGRAM** per indicare il protocollo **UDP** e non TCP
`data = bytearray(random.getrandbits(8) for _ in range(1024))`
 Quindi: 8 è 1Byte, **for** _ in range(1024): per arrivare fino a 1024 byte
 L'utilizzo dei SOCKET in più punti è stato volutamente introdotto per renderlo più rapido ed efficiente.

