

BYTE REBELS



UDP FLOOD

BYTE REBELS

ENG

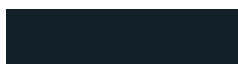
CANNAVACCIUOLO DAVIDE
DI MAIO PAOLO
FORLENZA SIMONE
RUSSO FEDERICO - LEADER
TIZZI FEDERICO
VAN ZWAM ARJEN

Today's exercise is to write a **Python** program that simulates a **UDP flood**, that is the mass sending of requests to a target machine that is listening on a random UDP port.

```
File Actions Edit View Help
(kali@kali)-[~]
$ python Progettopy.py

BYTE REBELS
Today's exercise is to write a Python program that simulates an UDP flood,
which involves massive sending of UDP requests to a target machine listening
on a random UDP port.
Enter the IP address of the target to scan: 192.168.1.15
Enter the port range to scan (e.g., 1-100,201-300): 1230-1240
█
```

UDP FLOOD CLIENT .PY



We projected this program with two main functionalities, port scanner and UDP Packet Sender.

Port Scanner: Allows to scan ports on a given ip address to determine which ports are open and which are closed or are not giving response. The user can specify a range of ports to scan and the target IP address . Then we will move on to the actual function of the project, the flood.

The program uses thread to run the scan in a more efficient way.

```
Port 1231 on 192.168.1.15 is CLOSED or unresponsive
Port 1232 on 192.168.1.15 is CLOSED or unresponsive
Port 1233 on 192.168.1.15 is CLOSED or unresponsive
Port 1234 on 192.168.1.15 is OPEN
Port 1235 on 192.168.1.15 is CLOSED or unresponsive
Port 1236 on 192.168.1.15 is CLOSED or unresponsive
Port 1237 on 192.168.1.15 is CLOSED or unresponsive
Port 1238 on 192.168.1.15 is CLOSED or unresponsive
Port 1239 on 192.168.1.15 is CLOSED or unresponsive
Port 1240 on 192.168.1.15 is CLOSED or unresponsive
```

03 ENG PORT SCANNER

What is it for?

Allows you to perform a port scan on a given **IP** address to determine which ports are open and which are closed or unresponsive.

Furthermore, the user can specify a range of ports to scan and the **IP** address of the target. The program uses threads to scan *more efficiently*.

Usage

Subsequently the user enters the range of ports to scan specifying the lower port and the higher port separated by a dash(-). For example 1-1000 The script runs the scan of the ports and prints the results in numerical order, giving a feedback.

Scope

Find the victim's open doors

```
Port 1231 on 192.168.1.15 is CLOSED or unresponsive
Port 1232 on 192.168.1.15 is CLOSED or unresponsive
Port 1233 on 192.168.1.15 is CLOSED or unresponsive
Port 1234 on 192.168.1.15 is OPEN
Port 1235 on 192.168.1.15 is CLOSED or unresponsive
Port 1236 on 192.168.1.15 is CLOSED or unresponsive
Port 1237 on 192.168.1.15 is CLOSED or unresponsive
Port 1238 on 192.168.1.15 is CLOSED or unresponsive
Port 1239 on 192.168.1.15 is CLOSED or unresponsive
Port 1240 on 192.168.1.15 is CLOSED or unresponsive
Enter the address: 192.168.1.15
Enter the port number: 1234
Enter the number of packets to send: 4
Packet 1 sent by Thread-1
Packet 2 sent by Thread-1
Packet 3 sent by Thread-1
Packet 4 sent by Thread-1
Thread-1 Terminated.
Packet 1 sent by Thread-2
Packet 2 sent by Thread-2
Packet 3 sent by Thread-2
Packet 4 sent by Thread-2
Thread-2 Terminated.
```

UDP PACKET SENDER

04
ENG

What is it for?

It allows to send a certain amount of **UDP packets** to a given IP address and associated port. The user can specify the destination IP(*victim*) , the port and the amount of packets to send. This feature also uses thread to send the packets simultaneously.

Usage

After having identified the port using the scanner, the user is prompted to enter the "victim" IP address to send the **UDP packets**. So the user enters the port number and number of packets to send. The script runs a certain amount of threads to send the UDP packet **simultaneously**.

Scope

Clogging the target IP with fake packets

```
(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 ENG

ANALYSES: Client - Flood.py

- 1.The program starts by asking for the target IP address.
- 2.Entering the range of ports to scan.
- 3.Enter the victim IP address.
- 4.Insert the previously acquired port.
- 5.Entering the amount of UDP packets to send.

```
\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(\xa8\xa0\x05<\xa3\x98C\xa8\x1d\x04\xb7p-\xdf:]\x8c\xfe\x1fY\xdb\xb7\x84\x12uW\x07e\x0e\xebT\
x8fm\xea\xf1\xb8\xa4\xce\xc2\x00LQ/\x98\x7f\x09\x13\x04\x9e\xcc\x9c\x874\''\xa2\\\xeb\x05\xab\x0d\x04\xa7f;/l\xdfA\xbbRp\x1d\
xfc\xa1\x0c\x05\x90\xdd\x12\xa8S\xb0\xb5\x9d0\xad\xf3\x96q-\''\xc3vG\xe3".E\xb8V\xb0\x97\xb5\xbf\xef\xe0\xe3"\xf3j\x06\x9f\x
eb\xb4S\x90\x0L\x04\x0b-\xfc\xdf\xc2\x13\x81\x9a\x05\x98\x05\x08\x0d\x1b\xf2\x1d2\xad\x0e?\xc4\x87Z2\x09gq\x8eGR2>\x92w<\xdf\x
cce\xc7\x0b)\xfeV\x04/\x0e\x8f\xaf\xc5\x06\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'\x83\x95\x89\x15\xa4#\x9dw(_l\xda\xaa
\xcdG\n\x09\\\x11\\\x90\xffa\xa5\xf6@\xb8{-\xb9\x0f@\xe0\xe8%\x0b\x03\x03\x0T\rH#\xf5\x0d+^\xc3\xc5\xf6\x11\x92\xca\xa0\xcd
\x069\xb9 \xd3\x04\xdf\x08EWcpj\xb7\x01\xb5\x04\xa8\xaf\xddR<\xa9\x87\xdeR\x8b\x0c2\x02+0\x02\x02\x06\x86G/\xe5Cm\xa9+\x95\x94\x
b5\xef2\x05\x08\x96\x9fGp\xa7\xe4\xb9\xb4k?h\xcc\x08\xcd\x1e\x9d\xbc\x0f81v"\xb9-7\xe99W\xca\x13\xf9\xa8P\x100600\x19\x0fay6\x
12k\x09(\xc2L\x00\x04\xfej\x1c\x09\xf1\xf5\xa2\x0f\x0d9\te\xf9\x03\xca\x93\xb9\x02\xde\xae\xaaQ\x08\x95\x85\x03ok\xcd\xa7\xcb
qR\x19+\xd3\x08T@\x0f\x06h\x84@<r\x03vE"\xee\xec\x0d1r\xfd\x08_ \x93w>\x0erN\xa2\x07f\x8a\xfe\x05@\xc7\xe9\x9eV\xec\xe1\x13\x18
l-\xf7\x0d26\xae\x06f\xa7Z\x0e\x0f8-\xa5\xcc \x13\x08c\x0b\x0cU\x08\x07\x0c\x19r'\x0e\xeb\xa5\x13\x0b8v\x0d\x09b\xce\x0bT\x83\x
8b\xbf\x02\x08\x0e\x0f5\x1c\x1d\x09\x89P\n\xa6\x04\x05\x0f1\x061x\xfbH\T\xbf\x06\x0333t\x08a-\x19\x06\x94\x9c\x08\x0b1q\x08\xa8
0j\xafk\x05a\xaf\x0f\x95K\x1cs^\x04m\rn\x80\x0d\x0d\x19\xff\xe5\xbc\xa7\x0b\x06\xa4\xeb\xee\x18\x05\xad\xabXI\x0d\x0dVa\xff\
xd7\x0fci\xa2\x0eK\xa3@\x0e\x192 \x0d\x93a\x05n\x19\x0e\x0f_v\x08\x05n\ '0A\x07\x0e\x19,\xa0\x09C0\x05\x0b\x03\x0b1\x01\x01\x
1eV^c j(\x03\x032\x01\x1c\x09\x19\x024n"\x091<3\xcc9I\x0b\x0d3l\x83\x02\x13\x06'\x0b\xa2r-\x1999\x83C\x0e\x07f\xff\x0d4Aeg\xe1\x091\x
8baC\x1e\x0e\x08\x14T\x07GL\x0f07H\x08\xa9j\xa8\x0e9\x0d\x0c9\x0f9\x07\x07wNA\x1d\x0efc\x05\x0d\x0e4\x0f4g\x0e\x0e\x0f8\x05\x0a\x02
"\x0e\x09e\x17\x01t\x05D\x09c|\xb4gn\x00$\x0b\x04\x0f8\x09f\x04\x0aA\x09\x0f26o\x0d\x0e64\x072\x00\x0df-\xa9\x0c-\xf4\x0e\x06\x087$\
xfc\x00\xa8\x94P\x84r\xaa\x0c\xaa\x0d\x05\x0dY\x0b\x0aff[Zg":\x0a\x0d7\x0e\x02\x0d\x02""\x0a\x0c3\x0f9-0\x13S\xa9\x0b1\xaa1\x0b0\x
d1\x00\x0e9\x0f8\x0b7\x1a\x0eeI\x04\x0b93j^R\x0c0;\xb1\x0bT\x06G\x0eT\x07f\x0b7\nG\x0e?\xc8\x12S\x0eb?\x13\x08f\x0d9\x086\x0f3,\nZ\x087
\xbbu\x0cdV\x0fd\x06U+\x04\x06\x18n\x0c6\x0c0\x042\x084>\x0a\x0d4\x0f8\x1a)\x19\x0a3\x0ac1\x055\x0d\x0d4b/\x0b4\x0d2/\x0e6\x1805\x0d0\x080\
```

ANALYSES: Server.py

We simply created a mini server that listens for UDP on its port (we assumed 1234).

