Pratique

ARP cache poisoning

Jean-Jamil Khalife

Degré de difficulté

On parle souvent des failles applicatives, celles liées à l'erreur humaine, etc. Ici, nous allons voir que des problèmes de sécurité peuvent aussi venir des protocoles réseaux. L'arp cache poisoning (ou l'empoisonnement de cache arp) est une attaque qui consiste à exploiter la faille du protocole ARP situé en couche 3 du modèle OSI. Le but est de détourner les communications entre deux machines distantes.

En premier lieu, nous verrons de manière assez synthétique comment fonctionne le dialogue entre les machines d'un réseau, selon le modèle OSI. Nous expliquerons ensuite comment il est possible d'usurper l'identité d'une machine d'un réseau local par exploitation d'une faille du protocole ARP. Enfin nous verrons comment faire pour contrer ce type d'attaque.

Tout d'abord, nous vous présenterons i analyse du dialogue.

Analyse du dialogue

Quand vous souhaitez communiquer avec une personne distante, vous allez par exemple utiliser votre téléphone portable pour lui envoyer un sms. Pour cela, vous allez remplir un certain nombre d'informations :

- le numéro de téléphone correspondant à l'adresse,
- le message que vous souhaitez lui envoyer,
- les ondes magnétiques feront quant à elles le reste du travail en transmettant le message à destination.

Un autre exemple : quand vous parlez à quelqu'un : vous allez utiliser la même méthode de communication que celle vue plus haut. Vous avez besoin de l'adresse du destinataire pour pouvoir lui parler, du message que vous allez émettre et enfin des ondes sonores qui vont le transporter jusqu'au récepteur (auditif ici) de son destinataire.

On va voir que pour émettre un message d'une machine à une autre, on utilise le même principe mais basé sur un modèle appelé

Cet article explique...

- Le fonctionnement du dialogue entre les machines d'un réseau local comprenant un récapitulatif du modèle OSI, du routage, du protocole arp.
- Le déroulement d'attaques par empoisonnement de cache arp.
- Les contre-mesures possibles.

Ce qu'il faut savoir...

- Modèle OSI : couches réseaux, protocoles d'échanges (pour mieux comprendre).
- Le principe du routage.
- Bases des systèmes Unix.

modèle OSI.

Rappel du modèle OSI

Nous n'allons pas expliquer en détail tout son fonctionnement car cela serait trop long mais juste faire un rappel suffisant pour que vous puissiez comprendre la suite de cet article.

Le rôle du modèle OSI est de permettre la communication entre les machines d'un réseau. Il comporte au total 7 couches. Chacune des couches comporte une entête qui permettra de stocker des informations relatives au schéma de principe vu précédemment (adresse, message), mais de nombreux autres paramètres seront aussi pris en compte comme par exemple le port de destination, les checksum, etc. Ce qui nous intéressera sera surtout l'adressage des packets (adresse ip et adresse mac). La Figure 1 montre quelles sont les différentes couches et comment elles sont réparties.

Ici les couches qui vont nous intéresser sont : les couches 2 et 3. *Mais comment fonctionne ce modèle ? Comment puis-je envoyer mon message avec tout cela ?*

Si vous souhaitez envoyer un message, vous allez remplir l'entête de chaque couche de la septième à la deuxième et la couche 1 se chargera d'envoyer vôtre message. Lors de l'envoi, le modèle OSI a recourt au principe d'encapsulation. Chaque couche de la septième à la première va s'envoyer le message et y ajouter son entête.

Donc en pratique :

- le message traverse la couche 7 qui y ajoute son entête,
- l'entête de la couche 7 et le message traversent la couche 6 qui y ajoute son entête,
- de même pour les autres couches,
- tout ceci va à la fin former un packet. Et la couche 1 s'occupera d'envoyer la trame ainsi construite sur le réseau.

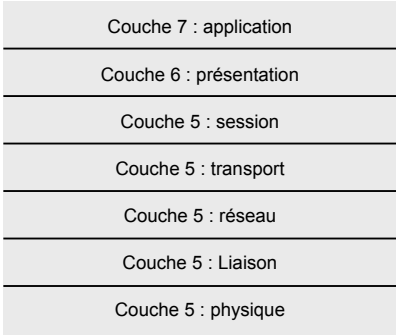

Figure 1. *Couches du modèle OSI*

L'opération inverse a lieu au niveau du destinataire. La première couche réceptionne les données et les envoie à la couche suivante (couche 2) qui garde les informations la concernant, à savoir son entête, puis transmet les informations restantes à la couche 3... et ainsi de suite jusqu'à la couche 7. Chaque couche isole donc l'information qui la concerne. Le but pour les logiciels réseaux est souvent d'envoyer (ou de récupérer) un message. Ce message est placé (ou récupéré) dans l'entête de la couche 7 (la couche application). La Figure 2 représente un récapitulatif du principe d'encapsulation.

Les entêtes de chaque couche contiennent des informations, mais quelles sont-elles ?

Détailler l'intégralité du fonctionnement de chaque entête serait trop long et hors sujet, mais nous conseillons tout de même aux personnes intéressées d'étudier le sujet de leur côté... (l'auteur Guy Pujolles a publié d'excellents ouvrages sur le réseau).

En réalité, nous ne remplissons pas les entêtes des couches directement mais on a recourt pour cela à des protocoles spécifiques pour chacune d'entre elles. On remplit donc l'entête de chaque protocole utilisé et les couches s'occupent d'ajouter ces entêtes à la leur.

De manière générale, la couche 2 s'occupe de l'adressage physique du packet. La couche 3 s'occupe elle, de l'adressage ip servant à gérer la connectivité de plusieurs machines ne se situant pas sur le même réseau.

- pour la couche 2, on utilisera le plus souvent le protocole ethernet,
- pour la couche 3, on utilisera le protocole ip.

Le protocole ethernet contient entre autres dans son entête : l'adresse de la machine source, l'adresse de la machine de destination et d'autres informations. Les adresses en questions sont les adresses MAC, elles sont uniques pour chaque carte réseau.

Le protocole ip quant à lui, possède une entête contenant : l'ip de la machine source, l'ip de la machine de destination et d'autres informations.

Résumé : Pour dialoguer, on utilise un système à couches appelé modèle OSI.

Chaque couche a un rôle dans la modélisation (et la réception) d'un message à envoyer (ou à recevoir)

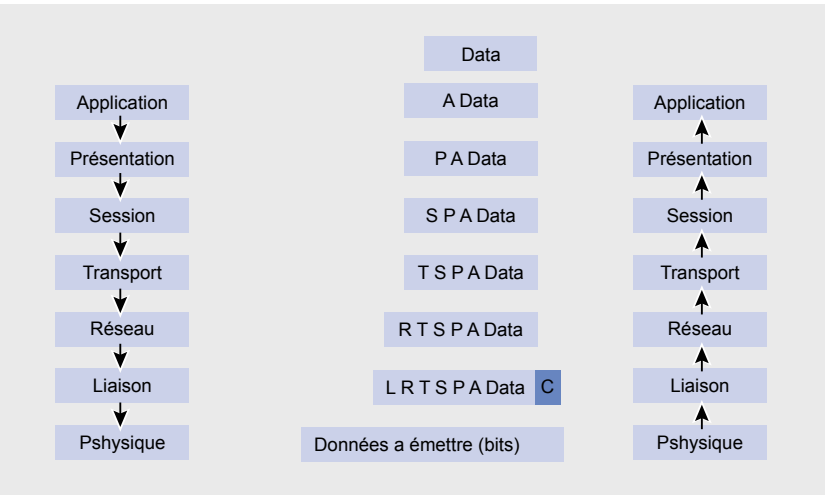

Figure 2. *Encapsulation*

sur le réseau. Ce rôle est régit par différents protocoles (ethernet, ip,...) qui ont pour but d'assurer la bonne transmission des données d'une machine A à une machine B.

Le routage

Il existe plusieurs architectures réseaux pour permettre aux machines de dialoguer entre elles. Nous étudierons le cas du réseau en étoile. Dans un réseau de ce type, les machines sont reliées à un noeud central qui a pour but de diriger les données d'une machine à une autre.

Il existe différents types de périphériques faisant office de noeud, notamment le *hub* et le *switch*.

- Le *hub* : lorsqu'une machine envoie un packet, celui-ci va passer par le hub qui va l'envoyer à tous les hôtes qui sont y branchés, donc toutes les machines branchées aux hub recevront le packet. Ce n'est donc pas sécurisé et qui peut se brancher au hub, peut analyser toutes les données qui transitent sur le réseau.
- Le *switch* : un peu plus sécurisé, à la même fonctionnalité que le hub, sauf qu'il ne diffuse pas les packets à toutes les machines mais seulement à celle de destination. Nous allons voir que le switch n'est pas si fiable que ça.

Dans la partie d'avant, nous avions vu qu'il était possible de dialoguer entre deux machines d'un réseau local à l'aide d'un switch ou d'un hub. Mais une question est à se poser : pourquoi a-t-on besoin de l'adresse ip de la machine de destination si on a déjà son adresse mac ? Pourquoi faut-il 2 adresses et pas une seule ?

Si vous devez dialoguer avec une machine qui se trouve sur le même sous-réseau que le vôtre, en théorie, l'adresse MAC suffira mais si la machine de destination se situe sur un réseau externe, il vous faudra absolument son adresse ip afin qu'il soit possible de router le packet vers le bon réseau. En effet, le switch analyse juste l'entête ethernet des packets qu'il reçoit. Il regarde l'adresse mac de destination et envoie le packet à la machine qui la possède. Mais les couches supérieures ne seront jamais prises en compte par le switch, donc l'entête ip ne pourra être analysée et il ne sera donc pas capable de router le packet en dehors du sous-réseau qu'il gère. Pour cela, on utilisera un routeur que l'on branchera sur le switch. Il se comportera comme une machine sauf qu'il sera capable de rediriger les packets dont l'ip de destination n'est pas la sienne.

En général, un routeur possède plusieurs interfaces réseaux. Chaque interface réseau est reliée à un sous-réseau possédant une plage d'adresse ip spécifique. Par exemple l'ip de l'interface 1 du routeur est 192.168.0.1 et fait partie du sous-réseau dont les adresses ip des machines sont comprises entre 192.168.0.1 et 192.168.0.255. Ceci fait intervenir la notion de masque sous-réseau. Nous vous laissons vous documenter là-dessus si vous voulez en savoir plus. La Figure 4 montre un schéma type.

Nous avons donc vu que l'adresse ip était indispensable si l'on souhaitait dialoguer avec une machine faisant partie d'un autre sous-réseau.

Venons en au fait. Quand vous envoyez un message à une machine du même sous réseau que vous, celui-ci va être délivré directement par le switch (c'est ce que l'on appelle la remise directe). L'entête Ethernet va posséder l'adresse mac de la machine source et celle de la machine de destination. De même pour l'entête ip avec les adresses ip.

Mais si vous envoyez le message à une machine provenant d'un sous réseau distinct du vôtre, l'entête Ethernet sera différente, et c'est là que le routeur intervient...vôtre

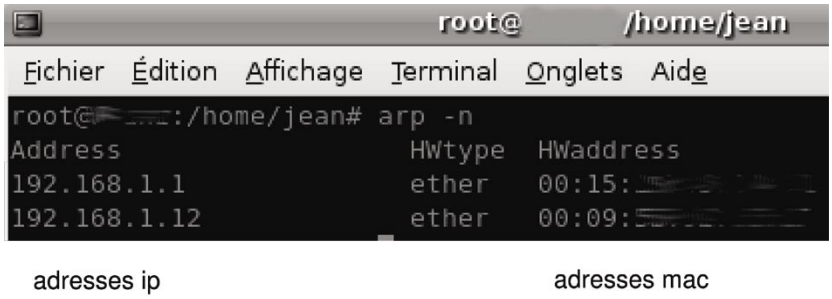

Figure 5. *Exemple de contenu d'un cache arp*

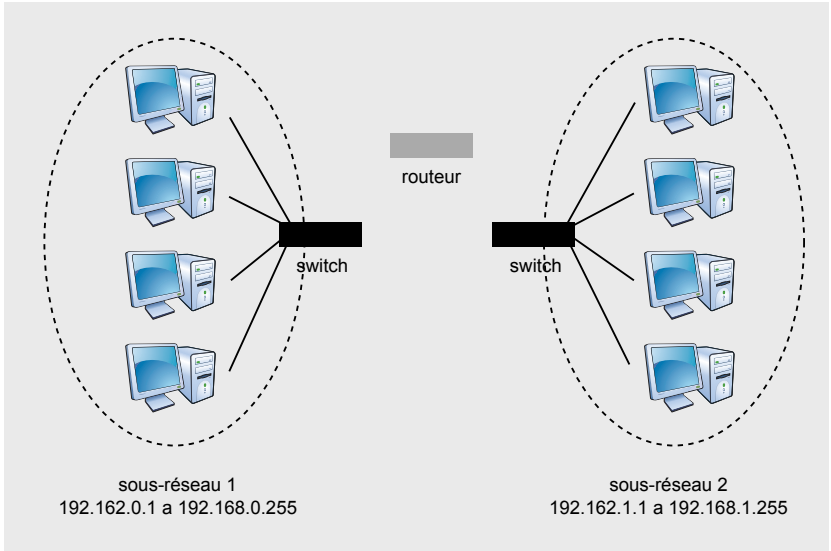

Figure 4. *exemple de routage*

machine va inscrire l'adresse mac du routeur comme destination au lieu de celle de la machine comme si elle voulait dialoguer avec la machine routeur. Le reste ne changera pas par rapport à la remise directe. Le routeur va regarder l'entête ip et il pourra alors savoir via sa table de routage sur quelle interface envoyer le message (sous-entendu à quel sous-réseau) et ensuite à quelle machine. C'est ce que l'on appelle la remise indirecte.

Dans tous les cas, il faudra remplir ces deux entêtes. En général, vous connaissez l'adresse ip de la machine avec laquelle vous vous adressez mais pas son adresse mac. Comment la trouver ?

Le protocole ARP

Il est nécessaire de remplir l'entête Ethernet pour envoyer un packet à une machine et pour cela, comme vu précédemment, nous inscrivons plusieurs informations dont l'adresse mac de la machine qui envoie et celle de destination. La machine qui envoie le message connaît sa propre adresse mac. Nous pouvons d'ailleurs la visionner à l'aide de la commande `ifconfig <votre_interface>` sous Linux, ou de la commande `ipconfig /all` sous Windows.

Le problème est de déterminer celle de la machine distante... Pour cela, il existe un protocole qui s'appelle le protocole ARP.

Il se situe en couche 3 du modèle OSI. Lorsqu'une machine souhaite connaître l'adresse MAC d'une autre, elle envoie à tous les membres de son sous-réseau un packet arp *who-as* en demandant quelle est l'adresse mac de la machine qui a telle adresse ip.

Et si la machine distante se trouve sur un autre sous-réseau ?

- Envoyer un packet arp à la machine cible ne servira a rien. Il ne sera pas routé vers un autre sous-réseau car l'entête ip n'est pas présente ici.
- Nous avons vu dans le chapitre Le routage que si la machine distante est sur un autre sous-réseau, il faut envoyer le packet à la passerelle (le routeur) donc on enverra un packet arp au routeur pour connaître son adresse mac.

Comment fait-on pour envoyer a tout le monde ?

Dans l'entête Ethernet, au lieu de mettre l'adresse mac du destinataire, on met : ff:ff:ff:ff:ff:ff , on dit alors que l'on envoie en broadcast. Seule la machine concernée va y répondre à l'aide d'un packet arp reply contenant son adresse MAC, les autres ignoreront le packet. Une fois le packet de réponse reçu, la machine source peut alors complètement remplir l'entête Ethernet car elle connaît l'adresse MAC du destinataire.

Il faudrait à chaque fois envoyer un packet arp pour connaître l'adresse mac de la machine avec qui on souhaite établir un dialogue ?

Non, nous encombrerions le réseau assez rapidement, surtout si de nombreuses machines y sont connectées. Donc pour résoudre ce problème, chaque hôte (machine ou routeur) possède un cache arp, c'est à dire un endroit en mémoire où il va pouvoir indiquer les correspondances entre les adresses ip des machines et leur adresse mac. Le contenu de ce cache est temporaire. Cela signifie qu'il faudra tout de même réitérer l'envoi de requête ARP mais

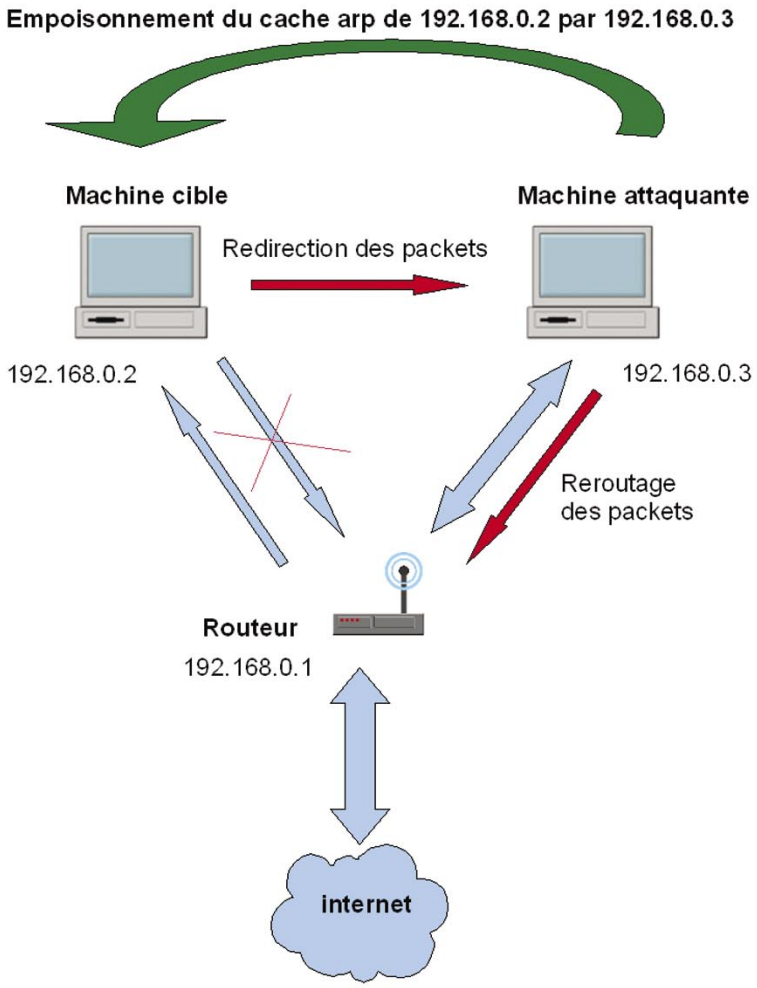

Figure 6. *Illustration de l'attaque*

de façon beaucoup moins fréquente. Pour voir le contenu du cache, tapez la commande suivante : `arp -n` sous Linux ou `arp -a` sous Windows.

Usurpation d'identité par empoisonnement de cache arp

Résumons : pour envoyer un message à une machine du même sous-réseau, il faut notamment son adresse mac et pour avoir son adresse mac, on regarde d'abord dans notre cache arp pour voir si elle y est. Si elle n'est pas présente, on envoie un packet arp à tout le monde et on demande *qui a l'adresse mac associé à cette ip ?*. La machine en question nous répond et notre cache arp est mis à jour. Nous pouvons alors obtenir l'adresse mac de la machine distante et lui envoyer le packet. Dans tout le reste de l'article, on se placera dans le cas d'un réseau étoilé muni d'un point d'accès qui fait à la fois switch et passerelle internet.

Approche de l'attaque

Il n'est pas nécessaire d'attendre qu'une machine vous demande votre adresse mac. Vous pouvez très bien la lui communiquer à n'importe quel moment en lui envoyant un simple packet ARP *reply*. Cela mettra à jour son cache ARP. Maintenant, imaginez que quelqu'un modélise et envoie un packet arp *reply* à une machine avec de fausses informations...C'est à ce moment que l'arp cache poisoning intervient.

Avec tout cela, quels types de risques peuvent surgir ? Dans un réseau local, on pourrait imaginer qu'un hacker se fasse passer pour une machine qu'il n'est pas et de ce fait intercepte le dialogue entre deux hôtes.

Prenons un exemple : si nous corrompons le cache arp de la victime en y inscrivant la correspondance entre l'adresse mac de l'attaquant et l'adresse ip du routeur, tous les packets qui transitent de la machine cible au routeur seraient alors interceptés par l'attaquant. Cela permettrait notamment d'intercepter les requêtes émises sur internet par la machine cible.

Mais il reste tout de même un problème à résoudre pour l'attaquant. Les packets émis vont en effet passer par la machine du hacker mais ils ne seront pas reroutés vers la bonne machine !!! Ainsi, la machine cible ne pourra plus envoyer de paket au delà de son réseau local. Pour pouvoir les intercepter de manière transparente, l'attaquant doit activer le mode routage ip sur sa machine. Cela va permettre de rerouter l'intégralité des packets dont l'adresse ip de destination est différente de la sienne. Pour ce faire :

- sous Linux (il faut être logué en

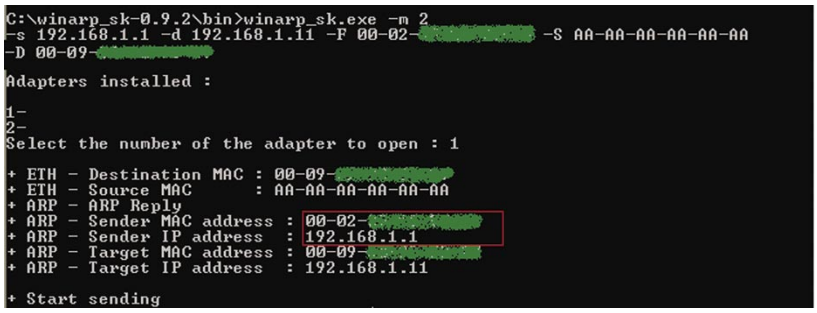

Figure 10. *Winarp_sk*

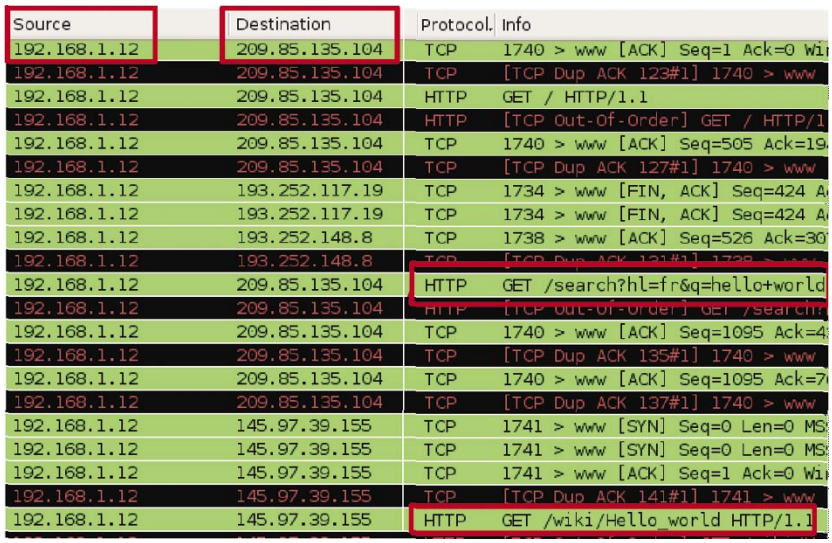

Figure 9. *Sniffing de requêtes http de la machine A*

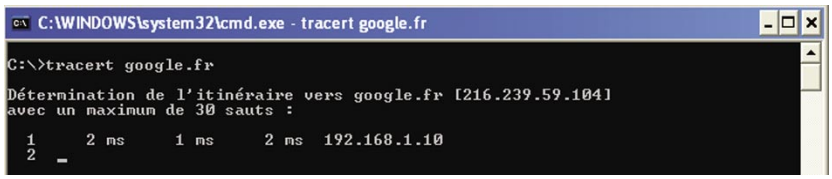

Figure 8. *trace du packet envoyé*

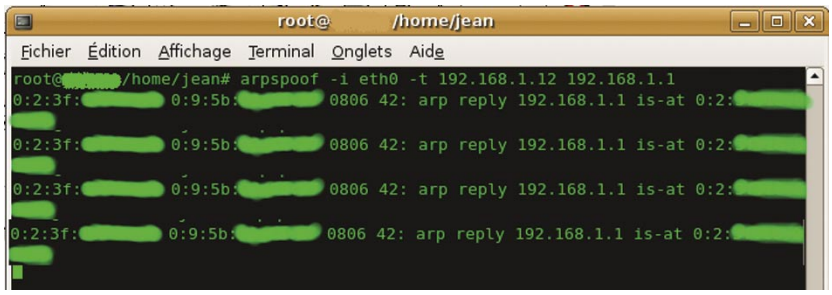

Figure 7. *Lancement de arpspoof*

root) : `echo 1 > /proc/sys/net/ipv4/ip_forward`

- sous Windows, il suffit d'ajouter la valeur suivante dans la base de registre. Dans : `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters` ajouter la valeur : `IPEnableRouter`, avec comme type : `REG_DWORD`, et comme donnée : `1`.

Nous pouvons aussi imaginer que l'attaquant ne souhaite pas rerouter les packets interceptés pour empêcher le dialogue entre les machines.

Mise en place de l'attaque

Une attaque peut donc être réalisée envers n'importe quelle machine d'un réseau local (hôtes, routeurs,...). Nous resterons dans le cadre d'un réseau local switché.

Voici le cas pratique sous Linux.

Nous considérons trois hôtes :

- un routeur : ip = 192.168.1.1
- une machine A (la cible) : ip = 192.168.1.12 OS : Windows
- une machine B (l'attaquant) : ip = 192.168.1.10 OS : Linux

on se placera du côté de la machine attaquante.

Commençons par mettre nôtre machine en mode routage : `echo 1 > /proc/sys/net/ipv4/ip_forward`

On empoisonne ensuite le cache *arp* de la victime. Nous allons utiliser l'outil *arpspoof* disponible sous Linux. Pour l'installer : `apt-get install dsniff`

Il s'utilise de la manière suivante :

```
arpspoof -i <iface> -t <target> host
```

- `<iface>` : interface réseau,
- `<target>` : ip de la machine à empoisonner,
- `host` : l'adresse ip que vous voulez associer à votre adresse mac.

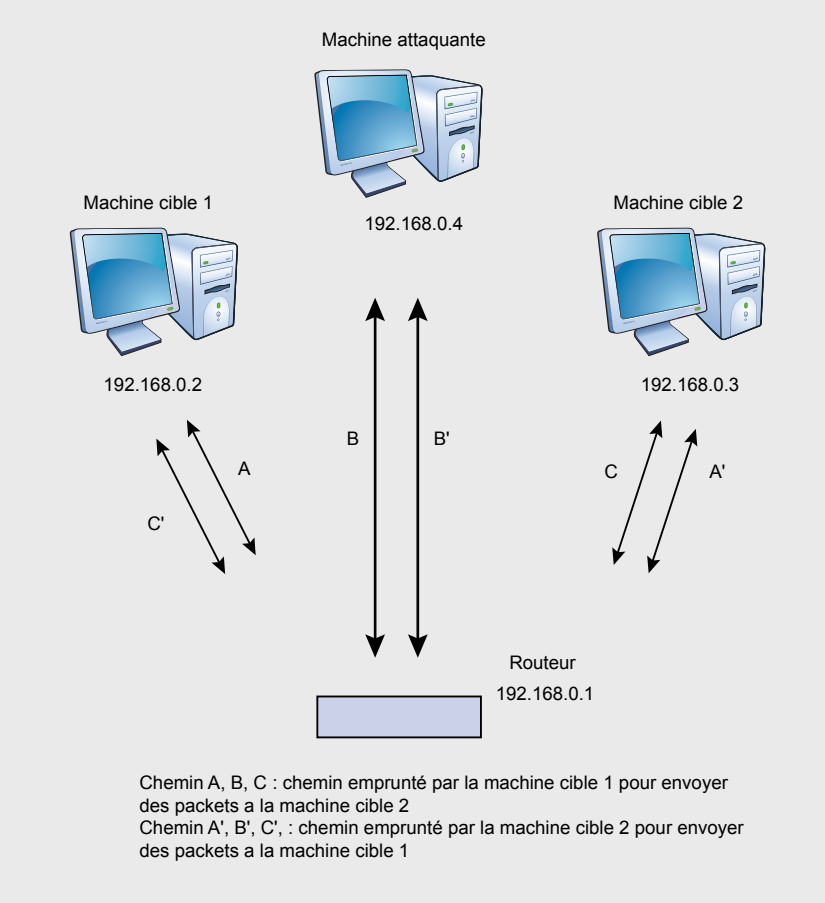

Figure 5. *Illustration attaque man in the middle*

on empoisonne le cache *arp* de la machine A en disant que notre adresse mac correspond à l'adresse *ip* du routeur. De, ce fait, tous les packets qui voudront sortir du sous-réseau seront routés vers nous.

Vérifions si les packets passent bien par notre machine avant d'être reroutés vers la passerelle : pour cela, on va faire un *tracert* (traceroute sous Linux) depuis la machine cible.

On constate bien que les packets passent d'abord par 192.168.1.10 (nôtre machine).

Il ne nous reste plus qu'à intercepter les packets qui transitent. Pour cela, nous pouvons utiliser le sniffer : wireshark (anciennement ethereal).

Nous avons réalisé cette attaque sous Linux. Mais il est également possible de le faire sous Windows. L'outil wireshark existe aussi sous cet OS. Pour réaliser l'attaque précédente sous Windows, nous allons utiliser l'outil `winarp_sk`. Il s'utilise aussi en ligne de commande.

Nous considérerons le même cas de Figure que le précédent sauf que l'adresse ip de la machine cible sera cette fois 192.168.1.11

- `-m 2` : permet d'envoyer des packets arp reply,
- `-s` : adresse ip de la machine cible,
- `-d` : adresse ip de la machine qui envoie le packet (ici l'ip du routeur),
- `-F` : adresse mac de la machine qui envoie le packet (ici la nôtre),
- `-S` : adresse mac de la machine qui envoie le packet,
- `-D` : adresse mac de la machine qui reçoie le packet.

Pour le reroutage de packets, il faut modifier (ou créer si elle n'y est pas) une clef dans la base de registre (voir chapitre II.1)

Exemple d'attaque : man in the middle

Cette attaque est sûrement l'une des plus connues dans l'exploitation

de ce type de faille. L'idée est de s'interposer virtuellement entre deux machines afin d'espionner leur conversation. La Figure 5 illustre ce principe.

Pour cela, il va falloir empoisonner le cache arp des deux machines (les machines 1 et 2). De ce fait, Si nous reprenons l'exemple de la Figure 5, l'attaquant va associer son adresse mac avec l'adresse ip de la machine 1 et ajouter cette correspondance dans le cache arp de la machine 2. et de même pour l'autre machine : il va associer son adresse mac avec l'adresse ip de la machine 2 et ajouter cette correspondance dans le cache arp de la machine 1.

En pratique, il suffit d'ouvrir deux consoles et de taper les lignes de commandes suivantes : `arpspoof -i eth0 -t 192.168.0.2 192.168.0.3` et `arpspoof -i eth0 -t 192.168.0.3 192.168.0.2`, sans oublier d'activer le routage ip sur votre machine.

Vous recevrez ainsi les communications de 1 vers 2 et de 2 vers 1.

Contre-mesures

Arp statique, La première solution est de définir les correspondances adresse_ip/MAC dans le cache arp de chaque machine, et ceci de manière permanente (ou jusqu'à redémarrage de la machine). Pour faire ceci, il faudrait pour chaque machine utiliser la commande : `arp -s inet adresse_ip adresse_mac`. Le problème est qu'il va falloir appliquer cette commande pour toutes les machines du réseau, ce qui peut être assez lourd si le réseau est doté de nombreuses machines.

Le filtrage, une autre solution consiste à filtrer les adresses mac entrantes. L'idée est d'obliger chaque machine d'un sous-réseau à vérifier si l'adresse mac source et l'adresse ip source du packet qu'elles reçoivent concordent bien avec celles des machines sources.

Pour cela nous pouvons utiliser sous Linux, l'outil netfilter de la manière suivante.

```
[root@Linux]# iptables -A INPUT -m mac
--mac-source 00:09:11:5A:8B:25
-s hote -j ACCEPT
```

Ici, on accepte les packets si l'adresse mac source est 00:09:11:5A:8B:25, et l'adresse ip source celle de hôte. Il faudrait ensuite faire de même pour chaque autre machine. On pourrait par exemple faire un script shell qui se chargerai de lancer cette ligne de commande pour les adresses (*mac* et *ip*) de chaque autre machine du sous-réseau.

Conclusion

Nous avons pu voir et démontrer, en s'appuyant à la fois sur des principes fonctionnels et à la fois sur des exemples pratiques, que le protocole ARP fait partie des protocoles sensibles.

Il est en effet assez aisé de détourner les communications entre deux machines distantes faisant partie d'un même réseau local, notamment grâce à l'utilisation de petits programmes comme `arpspoof`, `winarp_sk` ou `scapy` (outil réseau écrit en python qui permet entre autres de forger des packets dont les packets `arp-reply` et de les envoyer). L'attaquant n'aura plus qu'à utiliser un sniffer pour lire les packets qu'il aura détournés, et il pourra rerouter ceux-ci à l'aide d'une simple ligne de commande sous Linux ou en modifiant la valeur d'une clef dans la base de registre de Windows.

Il est donc important d'y prêter attention en utilisant l'arp statique et d'appliquer, si possible des filtres à l'aide de pare-feu tel que netfilter.

À propos de l'auteur

Auteur du présent article est étudiant à l'Université de technologie de Belfort-Montbéliard. Il poursuit un cursus d'ingénieur et se passionne pour la sécurité informatique.