FICHE SYNTHÈSE : ARP (Address Resolution Protocol)

1. Introduction et Contexte

Problématique

Dans un réseau local (LAN), les machines communiquent au niveau de la couche liaison de données (couche 2 OSI) en utilisant des adresses MAC, tandis que les applications utilisent des adresses IP (couche 3). Le protocole ARP résout cette problématique en permettant de faire la correspondance entre une adresse IP et une adresse MAC.

Définition

ARP (Address Resolution Protocol) est un protocole de la suite TCP/IP défini par la RFC 826 (1982). Il permet de découvrir l'adresse MAC (Media Access Control) d'une machine à partir de son adresse IP sur un réseau local.

2. Principe de Fonctionnement

Processus de résolution ARP

Lorsqu'une machine A (192.168.1.10) veut communiquer avec une machine B (192.168.1.20) sur le même réseau local :

1. **Vérification du cache ARP** : La machine A consulte d'abord son cache ARP local
2. **Requête ARP (ARP Request)** : Si l'adresse n'est pas en cache, A diffuse une requête ARP en broadcast
    - Message : "Qui possède l'IP 192.168.1.20 ? Dites-le à 192.168.1.10 (MAC: AA:BB:CC:DD:EE:FF)"
3. **Réponse ARP (ARP Reply)** : La machine B répond directement en unicast
    - Message : "192.168.1.20 est à l'adresse MAC 11:22:33:44:55:66"
4. **Mise en cache** : La machine A stocke cette information dans son cache ARP

Le cache ARP

Le cache ARP est une table temporaire qui stocke les correspondances IP/MAC pour éviter des requêtes répétées. Chaque entrée a une durée de vie limitée (TTL) généralement de 2 à 20 minutes selon les systèmes.

3. Structure d'un Paquet ARP

Un paquet ARP fait 28 octets et contient les champs suivants :

```
+-----------------+-----------------+
| Hardware Type    | Protocol Type    | (4 octets)
+-----------------+-----------------+
| HLEN | PLEN      | Operation        | (4 octets)
+-----------------+-----------------+
| Sender Hardware Address (6 octets) | (6 octets)
+------------------------------------+
| Sender Protocol Address (4 octets) | (4 octets)
+------------------------------------+
| Target Hardware Address (6 octets) | (6 octets)
+------------------------------------+
| Target Protocol Address (4 octets) | (4 octets)
+------------------------------------+
```

Détails des champs :

- **Hardware Type** (2 octets) : Type de réseau (1 = Ethernet)
- **Protocol Type** (2 octets) : Type de protocole (0x0800 = IPv4)
- **HLEN** (1 octet) : Longueur adresse matérielle (6 pour MAC)
- **PLEN** (1 octet) : Longueur adresse protocole (4 pour IPv4)
- **Operation** (2 octets) : Type d'opération
  - 1 = ARP Request
  - 2 = ARP Reply
  - 3 = RARP Request
  - 4 = RARP Reply
- **Sender Hardware/Protocol Address** : MAC et IP de l'émetteur
- **Target Hardware/Protocol Address** : MAC et IP du destinataire

4. Commandes Pratiques

Linux/Unix/macOS

```
# Afficher le cache ARP
arp -a
# ou
ip neigh show

# Afficher une entrée spécifique
arp -a 192.168.1.1

# Ajouter une entrée statique
sudo arp -s 192.168.1.100 00:11:22:33:44:55

# Supprimer une entrée
sudo arp -d 192.168.1.100

# Vider tout le cache ARP
sudo ip -s neigh flush all
```

Windows

```
# Afficher le cache ARP
arp -a

# Ajouter une entrée statique
arp -s 192.168.1.100 00-11-22-33-44-55

# Supprimer une entrée
arp -d 192.168.1.100

# Vider le cache
netsh interface ip delete arpcache
```

Capture avec tcpdump

```
# Capturer les paquets ARP
sudo tcpdump -i eth0 arp -n

# Capturer et enregistrer dans un fichier
sudo tcpdump -i eth0 arp -w arp_capture.pcap

# Afficher en détail
sudo tcpdump -i eth0 arp -vv
```

5. Aspects Sécurité : ARP Poisoning/Spoofing

Vulnérabilités du protocole ARP

ARP n'a **aucun mécanisme d'authentification**, ce qui le rend vulnérable aux attaques. Les principales failles :

1. ARP accepte les réponses non sollicitées (Gratuitous ARP)
2. Pas de vérification de l'identité de l'émetteur
3. Le cache peut être écrasé facilement

ARP Spoofing/Poisoning

Principe : Un attaquant envoie de fausses réponses ARP pour associer son adresse MAC à l'adresse IP d'une autre machine (passerelle, serveur, etc.).

Conséquences :

- Man-in-the-Middle (MITM)
- Interception de trafic
- Déni de service
- Session hijacking

Exemple d'attaque :

```
Machine légitime : 192.168.1.1 → MAC: AA:AA:AA:AA:AA:AA
Attaquant : 192.168.1.50 → MAC: BB:BB:BB:BB:BB:BB
Victime : 192.168.1.100

L'attaquant envoie à la victime :
"192.168.1.1 est à BB:BB:BB:BB:BB:BB"

Le cache ARP de la victime est empoisonné !
```

Outils d'attaque (à des fins éducatives)

```
# arpspoof (dsniff package)
sudo arpspoof -i eth0 -t 192.168.1.100 192.168.1.1

# ettercap
sudo ettercap -T -M arp:remote /192.168.1.1// /192.168.1.100//

# bettercap
sudo bettercap -iface eth0
> set arp.spoof.targets 192.168.1.100
> arp.spoof on
```

Détection d'ARP Spoofing

```
# Utiliser arpwatch (surveillance du réseau)
sudo arpwatch -i eth0

# XArp (Windows/Linux) - détection d'anomalies
# Wireshark - filtres pour détecter des anomalies
arp.duplicate-address-detected
arp.opcode == 2 && arp.src.hw_mac != eth.src
```

Contre-mesures

1. **Entrées ARP statiques** : Pour les machines critiques

   ```
   sudo arp -s 192.168.1.1 AA:BB:CC:DD:EE:FF
   ```

2. **Dynamic ARP Inspection (DAI)** : Sur les switches managés
   - Validation des paquets ARP contre une base de données
   - Bloque les réponses ARP suspectes
3. **Port Security** : Limitation du nombre d'adresses MAC par port
4. **Segmentation réseau** : VLANs pour limiter la portée des broadcasts
5. **IPv6 et NDP** : Le protocole de découverte de voisins IPv6 inclut des mécanismes de sécurité (SEcure Neighbor Discovery - SEND)
6. **Outils de monitoring** :
   - ArpON (ARP handler inspection)
   - XArp
   - Arpwatch

6. Variations et Protocoles Liés

RARP (Reverse ARP)

Protocole obsolète permettant d'obtenir une adresse IP à partir d'une adresse MAC (remplacé par DHCP).

Proxy ARP

Un routeur répond aux requêtes ARP au nom d'autres machines, permettant la communication entre réseaux sans que les machines n'aient de route configurée.

Gratuitous ARP

Une machine envoie une requête ARP pour sa propre adresse IP, utilisé pour :

- Détecter des conflits d'adresses IP

- Mettre à jour les caches ARP après un changement de MAC
- Annonce de présence sur le réseau

IPv6 : Neighbor Discovery Protocol (NDP)

Remplace ARP en IPv6, utilise ICMPv6 avec des mécanismes de sécurité améliorés.

7. Script Python pour Analyser ARP

```
from scapy.all import ARP, sniff, Ether

def arp_monitor(packet):
    if packet.haslayer(ARP):
        if packet[ARP].op == 1:  # ARP Request
            print(f"[REQUEST] Qui a {packet[ARP].pdst}? Dit à
{packet[ARP].psrc}")
        elif packet[ARP].op == 2:  # ARP Reply
            print(f"[REPLY] {packet[ARP].psrc} est à {packet[ARP].hwsrc}")

# Capturer les paquets ARP
print("Surveillance ARP en cours...")
sniff(prn=arp_monitor, filter="arp", store=0)
```

8. Points Clés à Retenir

- ARP fonctionne uniquement en **couche 2** sur un réseau local
- Les requêtes ARP sont envoyées en **broadcast** (FF:FF:FF:FF:FF:FF)
- Les réponses ARP sont envoyées en **unicast**
- Le cache ARP améliore les performances mais présente des **risques de sécurité**
- ARP **n'a pas de mécanisme d'authentification** (vulnérable aux attaques)
- Pour les communications inter-réseaux, le protocole interroge la **passerelle**
- IPv6 utilise **NDP** au lieu d'ARP avec de meilleures sécurités

9. Ressources Complémentaires

- RFC 826 : An Ethernet Address Resolution Protocol
- RFC 5227 : IPv4 Address Conflict Detection
- RFC 3927 : Dynamic Configuration of IPv4 Link-Local Addresses
- Man pages : arp(8), ip-neighbour(8)
- Wireshark : https://www.wireshark.org/
- Scapy : https://scapy.net/