

Altering ARP Tables (version 1.00)

Index

Introduction
Switching
(R)ARP packets
Altering ARP Tables
Going to reality
Workstations are vulnerable too
A worse scenario
ARP poison vaccine
Conclusion

Introduction

After month's of doing everything except writing a new paper or updating an old one, I'm back with a new subject. Because I only want to write about subjects that are not very common, I will not publish much tutorials/papers in the future. This paper is dedicated to ARP tables and how to alter them remotely. The paper also describes a couple of implementations of ARP poisoning in a bridge based segment and a couple of ways to protect yourself. As usual: I'm not responsible for any of your stupid actions while practicing the following info at places you shouldn't be.

Switching

First to get things straight into your brain, it's rather important to study my/this theory well.

We all have heard of the expression "sniffing" ... setting you network device in promiscuous mode so you can intercept packets destined for your neighbour and such. What many people don't know, is that you can't intercept packets at a switch based segment. Why not...?

Let's have a look at the Table below...

Hosts	Port (At Switch)	MAC	IP
Host 1	Port 1	ABCDEF000001	169.254.0.1
Host 2	Port 2	ABCDEF000002	169.254.0.2
Host 3	Port 3	ABCDEF000003	169.254.0.3

As example:

Host 1 runs a *nix OS with a FTP daemon, Host 2 is the network administrator's workstation and Host 3 is my laptop. These hosts are

connected with eachother through a level 2 switch. The expression "level 2" reference to the datalink layer of the Open System Interconnection (OSI) Model. The kernel of Host 2 is building a packet and the destination will be Host 1, the packet will look like this:

TCP header
 IP header
 802.3 header
 Datagram

Protocol Type	Source IP	Destination IP	Source MAC	Destination MAC
TCP Header	X	X	X	X
IP Header	169.254.0.2	169.254.0.1	X	X
802.3 Header	X	X	ABCDEF000002	ABCDEF000001
Datagram	X	X	X	X

When the switch will be turned on, it will build a table of the MAC addresses of all hosts who are physically connected to the switch. The ARP table will be (re)build by sending ARP requests over the network, and with the returned information (ARP reply's) the ARP table will be (re)build.
 In my example:

The MAC address of the host 'at' Port 1 = ABCDEF000001
 The MAC address of the host 'at' Port 2 = ABCDEF000002
 The MAC address of the host 'at' Port 3 = ABCDEF000003
 And so on...

When the switch receives a packet, it will read only the 802.x or etherII header because it's a level 2 switch... DOH! Then the destination MAC will be compared with it's local ARP table... and soon the switch has a match. The Host behind Port 1 is the destination and the switch sends the packet to Host 1. The packet will not pass over the network interface at Host 3!
 Before we continue with altering ARP tables first a short overview of (R)ARP packets.

(R)ARP Packets

At reguraly base ARP caches will be updated from ARP reply's. First an ARP request is being send over the network and the appropriate host will respond with an ARP reply. Within this packet is information needed to refresh the ARP cache. As noted ARP packets are commonly presented in two forms; ARP requests & ARP reply's. Below is a human readable dump:

ARP request:

Ethernet II
 Destination: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
 Source: 00:12:06:19:82:00 (00:12:06:19:82:00)

Type: ARP (0x0806)

Address Resolution Protocol (request)

Hardware type: Ethernet (0x0001)

Protocol type: IP (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: request (0x0001)

Sender hardware address: 00:12:06:19:82:00

Sender protocol address: 169.254.0.1

Target hardware address: 00:00:00:00:00:00

Target protocol address: 169.254.0.5

ARP Reply:

Ethernet II

Destination: 00:12:06:19:82:00 (00:12:06:19:82:00)

Source: 00:e0:4c:39:65:37 (00:e0:4c:39:65:37)

Type: ARP (0x0806)

Trailer: 202020202020202020202020202020202020...

Address Resolution Protocol (reply)

Hardware type: Ethernet (0x0001)

Protocol type: IP (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: reply (0x0002)

Sender hardware address: 00:e0:4c:39:65:37

Sender protocol address: 169.254.0.5

Target hardware address: 00:12:06:19:82:00

Target protocol address: 169.254.0.2

RARP means Reverse Address Resolution Protocol, it's the opposite of an ARP. Normaly you won't find RARP packets very much at LAN's.

Altering ARP tables

I'm continuing my example of the three hosts... I suppose the 'sniff-problem' is very clear. Imagine we can bind multiple MAC addresses at Port 3 of the switch... in this case we want to bind one extra MAC at Port 3 of the switch.

The MAC table will look like this:

The MAC adress of the host 'behind' Port 3 = ABCDEF000003 & ABCDEF000001

When Host 2 sends a packet to Host 1, the switch will again compare the destination MAC with it's MAC table and now have 2 matches, and now the switch will happily bridge the packet to Host 1 & Host 3. I know there could appear some complications, like switches who update their tables every 30 seconds by sending an ARP request to all hosts who are physically connected to them. And 3COM sells some level 2 switches who refuse to bind more than 1 MAC to a Port, the only disadvantage of it is their price.

Going to reality

A highly interesting part for the scriptkiddies will be this one. Also because you won't need much networking knowledge to use ARP poisoning tools. I mainly used the ARP poisoning tool version 0.5 B from Steve Buer. It has an easy way to send your own customized packets over a LAN.

Now i'm going to show how you can bring down the route between 2 remote hosts. We have 3 hosts at a swichted based LAN.

```
Host   IP           MAC
Host 1: 169.254.0.1 / 00:10:4B:01:88:F3
Host 2: 169.254.0.2 / 00:12:06:19:82:00
Host 3: 169.254.0.5 / 00:E0:4C:39:65:37
```

We are Host 1 and we want to terminate the route between Host 2 & 3. It's recommended to know if both hosts are online, let's send some ICMP_requests to the them.

```
GateKeeper:~/arpoison # ping 169.254.0.2
PING 169.254.0.1 (169.254.0.1): 56 data bytes
64 bytes from 169.254.0.2: icmp_seq=0 ttl=255 time=2.009 ms
64 bytes from 169.254.0.2: icmp_seq=1 ttl=255 time=1.103 ms
[1]+  Stopped                  ping 169.254.0.2
```

```
GateKeeper:~/arpoison # ping 169.254.0.5
PING 169.254.0.5 (169.254.0.5): 56 data bytes
64 bytes from 169.254.0.5: icmp_seq=0 ttl=128 time=2.924 ms
64 bytes from 169.254.0.5: icmp_seq=1 ttl=128 time=0.990 ms
[2]+  Stopped                  ping 169.254.0.5
```

Well both hosts are online (for now :)....

After you have compiled the ARP poisoning tool source fill in the right information.

```
GateKeeper:~/arpoison # ./arpoison
Usage: ./arpoison -i <device> -d <dest IP> -s <src IP> -t <target MAC> -r <src MAC>
```

It doesn't matter which host we are going to poison because both are vulnerable. I'll poison Host 2... remember that the information you type after "-r" must not be the real MAC address!!! Otherwise you have build a valid packet and the route will not be broken.

```
GateKeeper:~/arpoison # ./arpoison -i eth1 -d 169.254.0.2 -s 169.254.0.5 -t 00:12:06:19:82:00 -r
AA:BB:CC:DD:EE:FF
ARP packet sent via eth1
```

From now until approx. 30 seconds further the route will be down between these 2 remote host.

What we exactly have done is very simple

to explain. We have send an malformed ARP reply to Host 2, this packet contains invalid information (the MAC address) about Host 3.

When Host 2 wants to connect to Host 3, the kernel at Host 2 will build a packet and the information like the MAC destination will be get from the local ARP table. Only this information is corrupt and the packet will never arrive at Host 3.

Note that every Operating System will refresh the ARP table within 30 or 60 seconds. In other words, if you permanently want to bring down the route between two remote hosts you'll have to send every 30 or 60 seconds a malformed ARP reply.

Another note is that this tool will spoof the real attacker IP and MAC address, so it's very hard for system administrators to pinpoint the real location where the packets are being sent from.

Workstations are vulnerable too

Like switches & routers also *nix, Macintosh & Windows machines aren't invulnerable to ARP poisoning. All these Operating Systems have the same problem... they will happily update their ARP tables (with wrong information :). I know BSD and *nix machines can be made pretty invulnerable when the kernel has been recompiled with some extra options. But I suppose not many users will even recompile the kernel from source if it's already working fine.

Below I have made several dumps from ARP tables from various Operating Systems... starting with Linux SuSE, my all round favorite.

OS: Linux SuSE 7.1

```
GateKeeper:~ # arp -nv -i eth0
Address      HWtype HWaddress      Flags Mask      Iface
212.187.0.1  ether  00:30:7B:94:31:C8 C              eth0
Entries: 4   Skipped: 3   Found: 1
```

```
GateKeeper:~ # arp -nv -i eth1
Address      HWtype HWaddress      Flags Mask      Iface
169.254.0.101 ether  00:E0:4C:39:65:6D C              eth1
169.254.0.6  ether  00:C0:4F:A7:63:81 C              eth1
169.254.0.2  ether  00:12:06:19:82:00 C              eth1
Entries: 4   Skipped: 1   Found: 3
```

OS: Windows '98 (second edition)

C:\>arp -a

```
Interface: 169.254.0.2 on Interface 0x20000002
Internet-adres      Fysiek adres      Type
169.254.0.1         00-10-4b-01-88-f3 dynamisch
169.254.0.101      00-e0-4c-39-65-6d dynamisch
```

C:\>

OS: FreeBSD 4.3

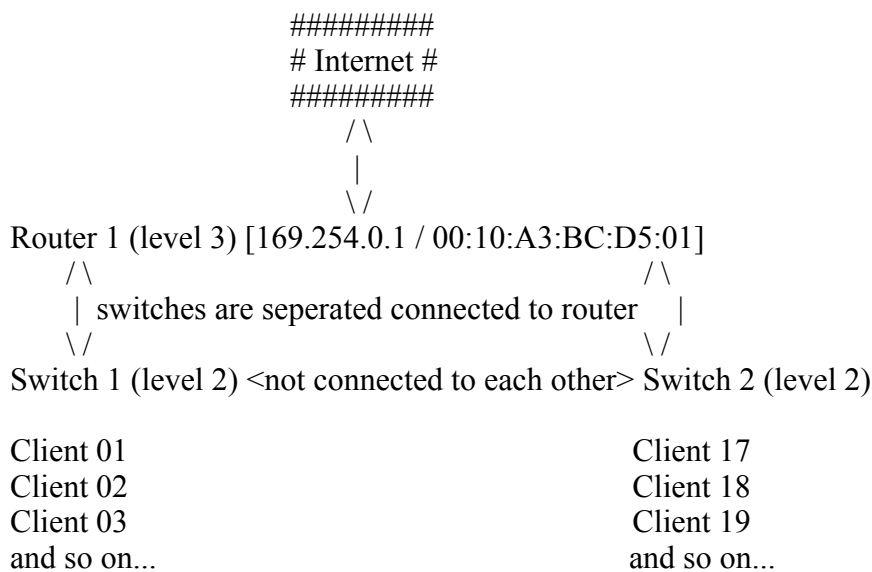
```
unreal@NederWiet:~ > arp -a
? (169.254.0.2) at 0:50:bf:5d:4e:6a [ethernet]
? (169.254.0.3) at 0:20:18:3a:fa:12 [ethernet]
? (169.254.0.5) at 48:54:e8:90:5f:cc [ethernet]
```

unreal@NederWiet:~ >

A worse scenario

The following lines I have wrote aren't based from the reality or whatsoever. I'm only guessing what damage could be done when information like this in combinations with the right tools in the wrong hands. Note, all information and thoughts have been tested by me in an isolated environment (LAN).

Imagine an ISP has a route segment with 32 clients, these 32 client have been divided in groups of 16 clients into 2 bridge segments. Below you see a simple picture.



Let's say Client 12 (located at switch 1) wants to bring down the route between Client 03 (also located at switch 1) and the router (also located at switch 1). If he succeeds, the victim cannot do anything but contact the hosts at his own bridge segment.

The attacker has two or three way's to accomplish this:

- 1) ARP poisoning Client 03.
- 2) ARP poisoning router (most times the smartest way).
- 3) ARP poisoning switch 1.

Another example...

Let's say Client 19 (located at switch 2) wants to bring down the route between Client 02 (located at switch 1) and the router. Note that this example does really differs from the first one. Simply because the attacker is located in another segment. And do you remember ARP reply's cannot pass through routers? Well it's pretty simple to accomplish this task. The attacker can still ARP poison the router with invalid information about Host 02.

Last example...

For the people who are still understanding my story...

You can even take a route down between 2 remote hosts which is for example 15 hops away... It just takes some more time though.

Because you will need to be at the segment where the target host is located. And the only way to accomplish such a task you will have to own/hack the router to that segment.

ARP poison vaccine

Above I described two ways how to '(mis)use' ARP poisoning at LAN's, ofcourse there are some goodies to prevent such 'updates'.

The most easiest way to prevent ARP poisoning at workstations and server with Open Source Operating Systems is to M-lock the ARP cache line by line. This means when the ARP table has an valid entry like this:

```
212.187.0.1      ether  00:30:7B:94:31:C8  C           eth0
```

You can lock this entry by typing: "arp -v -i eth0 -s 212.187.0.1 00:30:7B:94:31:C8" (without quotes)

Check the ARP cache again by typing "arp -nv -i eth0", the output will be:

```
212.187.0.1      ether  00:30:7B:94:31:C8  CM          eth0
```

See the difference? :)

As long as you won't unlock the ARP cache, restart the eth devices or reboot the system, nobody can refresh the entry above.

Another way would be installing a (level 2!!) firewall at the workstation, but the only difference between this and my way (above) will be the price. The firewall will exactly do the same, it's not making your system any more invulnerable or whatsoever!

Conclusion

With ARP poisoning you can do various things, first of all is sniffing at switched based segments by poisoning the remote hosts or switches.

Second, and most times much worse is altering ARP tables of routers, which renders LAN segments isolated from the other segments.

I strongly believe that in short time these kind of attacks will grow in number fast worldwide...

especially at educational places where security

isn't/can't be priority number one! And in the future there will be a time that ISP's, schools,

governments and much more have to upgrade

there switches/routers with flash utility's to create un-ARP'able LAN's. And this is maybe one of the most expensived vulnerability's ever found!

And if someone ask me when does this happen... my answer will be: When the majority will discover the advantages of ARP poisoning.

It's just a matter of time...

Shout outs to:

Pool: for correcting my grammar mistakes :)

NederWiet: for helping me the past 2 years with building up my *nix knowledge

DarkWhite: The Lan Party last week at your house was great ;)

KD: I'm not in love with you any more :~(

ssuzeJJ: Are you still pissed on me?

Copyright (C) 2001, DataWizard, The Netherlands.