The Ingredients to ARP Poison

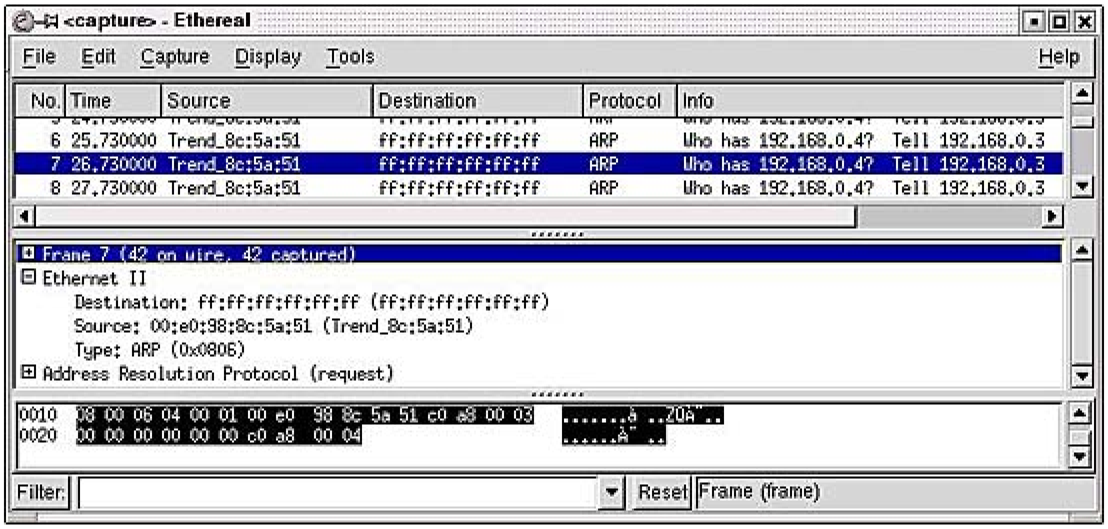
Superficially, a network appears to be nothing more than a collection of hardware devices connected with a seemingly infinite supply of wire and RF waves (wireless networks). However, underneath this mostly visible layer exists another much more complex world full of packets, protocols, and data properties. It is in this virtual realm of the network that the communication challenge is fully experienced. However, it is only through the merger of the physical and virtual realms that data flow correctly. One of the key ingredients in this complex scheme is the Address Resolution Protocol (ARP).

When data is sent out onto a network, it needs a way to find its destination. This is accomplished on several layers, depending on how far the data needs to travel. At the first layer, there exists an address called the Media Access Control address. This theoretically 100% unique value is systematically assigned to each and every network device that is produced. In other words, every network card, router, and switch has a pseudo-serial number that distinguishes it from every other network device in the world. However, the MAC address is used only to communicate within local networking segments, which are called subnets. Once the data passes through a router or switch to another network subnet, the next layer of addressing becomes important. This is because the database required to record every MAC address and its location would be too large for quick processing. Instead, other technologies, such as DNS (Domain Name Service), WINS (Windows Internet Naming Service), and IP (Internet Protocol) manage data flow the farther out the data travels. To facilitate this transmission of data, ARP was designed to act as the intermediary between IP and MAC addresses.

In short, ARP is responsible for managing the relationship between Media Access Control addresses and the IP addresses for network devices. This fundamental technology is part of the core of Internet functionality; in fact, without it a network will fail to work. However, it has been discovered that ARP information can be spoofed, or faked, to facilitate the control of all network data.

ARP Details

As mentioned, ARP is a helper protocol that assists in making networking a little bit easier, more efficient, and more reliable. Both IP addresses and MAC addresses provide an important part to networking. Not only does allowing the use of IP addresses provide a method for keeping internal networks separate from external networks, but IP addresses can also help to logically segment one network from another. While IP seems to single-handedly have delivery under control, relying solely on IP addressing would cause serious issues. For example, what happens if two computers on a network start using the same IP address? In a situation like this, chaos would reign, with both computers competing for each other's data. This is why each NIC has a unique address assigned to it.

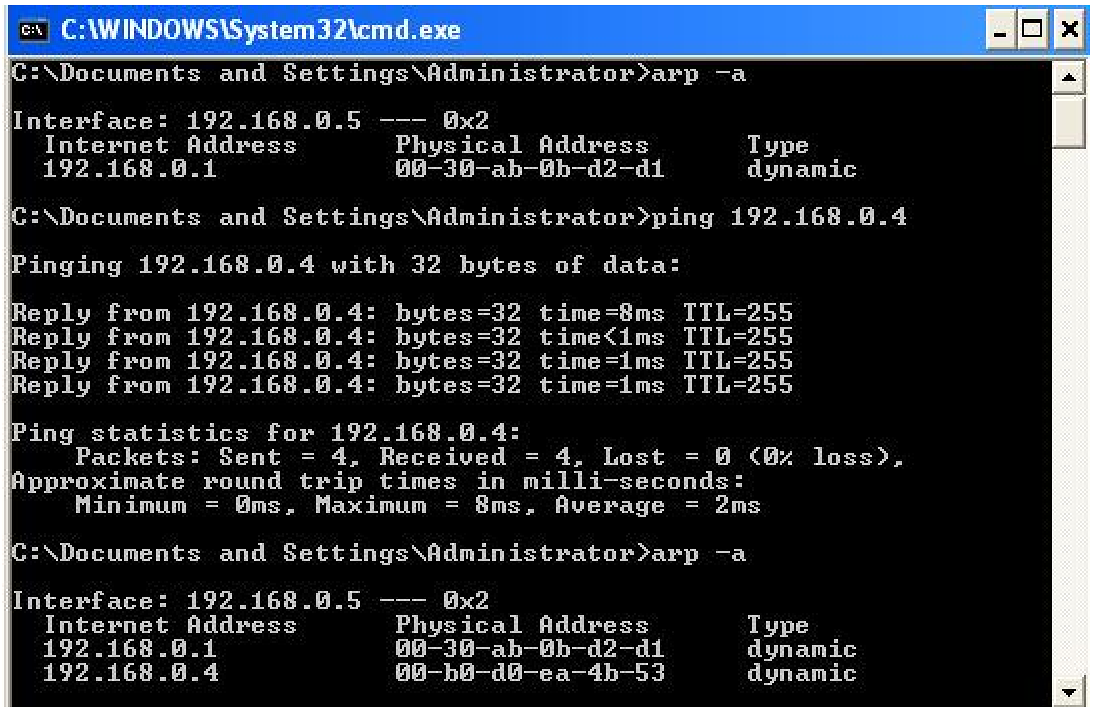

<capture> - Ethereal
File Edit Capture Display Tools Help
Filter: Reset Frame (frame)

```
No. Time      Source           Destination        Protocol Info
  6 25.730000 Trend_8c:5a:51   ff:ff:ff:ff:ff:ff  ARP      Who has 192.168.0.4?  Tell 192.168.0.3
  7 26.730000 Trend_8c:5a:51   ff:ff:ff:ff:ff:ff  ARP      Who has 192.168.0.4?  Tell 192.168.0.3
  8 27.730000 Trend_8c:5a:51   ff:ff:ff:ff:ff:ff  ARP      Who has 192.168.0.4?  Tell 192.168.0.3

⊞ Frame 7 (42 on wire, 42 captured)
⊟ Ethernet II
    Destination: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Source: 00:e0:98:8c:5a:51 (Trend_8c:5a:51)
    Type: ARP (0x0806)
⊞ Address Resolution Protocol (request)

0010  08 00 06 04 00 01 00 e0  98 8c 5a 51 c0 a8 00 03   .......à ..ZQÀ¨..
0020  00 00 00 00 00 00 c0 a8  00 04                     ......À¨ ..
```

Figure 1:

Ethereal capture of ARP request.

To handle the conversion during a data session, an ARP request is initially sent out by the client computer. This request basically asks the following question, "Will the computer with IP address xxx.xxx.xxx.xxx please send me your MAC address." (See Figure 1 for an example of ARP request). Since this information is important for data flow, ARP requests are sent out in broadcast mode, which means every computer on the LAN will receive the request. Eventually, the target computer receives the ARP request and sends back its MAC address back to the requesting computer. To save time in the future, this MAC address to IP address information is placed into a small database called the ARP table. If the destination is beyond the network's boundaries, the request is passed to the gateway devices, which uses the other technologies to find the target. Figure 2 illustrates this ARP process by showing you a before and after shot of an ARP table on Windows XP.



```
C:\Documents and Settings\Administrator>arp -a

Interface: 192.168.0.5 --- 0x2
  Internet Address      Physical Address      Type
  192.168.0.1           00-30-ab-0b-d2-d1     dynamic

C:\Documents and Settings\Administrator>ping 192.168.0.4

Pinging 192.168.0.4 with 32 bytes of data:

Reply from 192.168.0.4: bytes=32 time=8ms TTL=255
Reply from 192.168.0.4: bytes=32 time<1ms TTL=255
Reply from 192.168.0.4: bytes=32 time=1ms TTL=255
Reply from 192.168.0.4: bytes=32 time=1ms TTL=255

Ping statistics for 192.168.0.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 8ms, Average = 2ms

C:\Documents and Settings\Administrator>arp -a

Interface: 192.168.0.5 --- 0x2
  Internet Address      Physical Address      Type
  192.168.0.1           00-30-ab-0b-d2-d1     dynamic
  192.168.0.4           00-b0-d0-ea-4b-53     dynamic
```

Figure 2:

ARP in action.

The first couple of lines show us the original ARP table. This is gotten by issuing the command "arp –a". As you can see, there is only one entry, which is the IP address of the Internet gateway device. Next, we perform a PING on another computer on the LAN, which results in a positive reply. However, as we discussed previously, my network card first discovered the MAC address of the target computer before any packets could be sent to it. This accomplished, the information is entered into the ARP table, and the PING commences. The final part of the screen shot shows us the new ARP table, which now includes the MAC address for the recently targeted computer.

> **NOTE**: "arp " will output a detailed ARP table in Linux.
>
> **NOTE**: Use the command "arp –d *" to delete the ARP table in Windows and "arp –d <hostname>" in Linux.
>
> **NOTE**: Use the "arp -s" command to manually add static ARP entries. It is useful for connecting to unaddressed Ethernet devices.

There are various other things that the use and understanding of ARP can facilitate. For example, if a new Ethernet device is added to an existing network, but it has no method of predefining an acceptable IP address, ARP can be used to statically assign an IP address to the device using the unique MAC address. Using the command "ARP -s 192.168.0.10 00-20-4A-24-BF-C1" will make an entry in the local ARP table that points to this device on the Ethernet. Now, a user can use tools such as TELNET and PING to send data to this new device, which will in turn respond. However, there are ways ARP can be abused as well.

ARP Weaknesses

When Ethernet devices use ARP, it is within a set of rules and conditions. However, just because these rules exist does not mean they need to be followed. Thus, ARP can be abused and twisted to turn it into a hacker's tool. This section studies how ARP weaknesses can be, and are, exploited.

The first thing that needs to be understood is that ARP is NECESSARY for your network to work properly. In other words, if you or a hacker starts altering ARP tables incorrectly, the whole network could be taken offline. As we explain ARP hacking techniques, this message will be repeated several times. By the end of this article, you will understand why and how ARP works, and what damage it can do to a network.

ARP Spoofing

As we learned, Ethernet devices use MAC addresses to communicate. On top of this fundamental layer, other layers are used that are easier to read and understand, such as DNS names, WINS names, and IP addresses. In addition, we also learned that a MAC address to IP address table is usually stored locally on each computer. This helps speed up data transfer due to the simple fact that the MAC address doesn't have to be verified each and every time one device wants to communicate with another device. However, this advantage has a negative side.

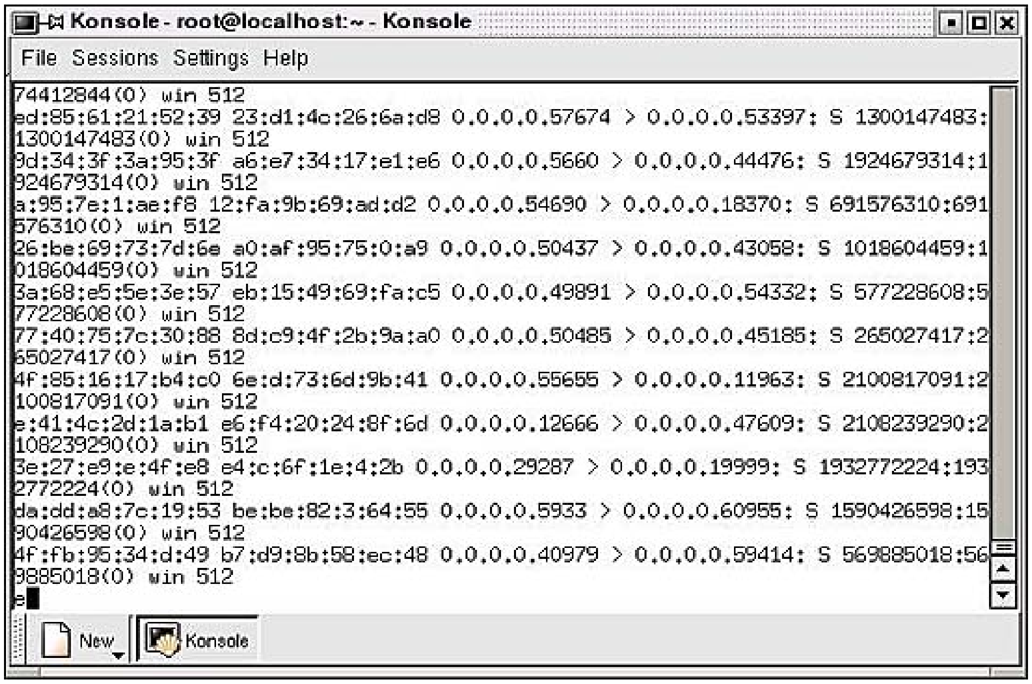
By storing the MAC addresses in the ARP table, a potential weakness arises. What would happen if a remote hacker could control an ARP table of a computer? They could change MAC to IP address entries, which could cause traffic to be redirected from the correct target to a target of the hacker's choice.

> **NOTE**: All MAC addresses are fictitious. They were selected to make illustration easier to understand. Do not attempt this on a network you do not OWN (and this doesn't mean illegally own).

In our example, a hacker wants to be able to intercept and sniff all data passing between computer A and the gateway. This would be one of the first choices for any hacker, due to the popularity of the Internet and the number of secure items that typically pass through a gateway. Another target would be an email server, Unix server (with TELNET), or an FTP server. Since these services typically send passwords in plain text, it would not take long before a hacker could glean a few passwords from the network.

Depending on how a hacker wanted to proceed, it is possible to attack the switch first. The reason for this is that the switch regulates the flow of data between its ports. It actively monitors the MAC address on each port, which helps it pass data only to its intended target. This is the main difference between a switch and passive hub. A passive hub has no mapping, and thus broadcasts line data to *every* port on the device. The data is typically rejected by all network cards, except the one it was intended for. However, in a hubbed network, sniffing data is very easy to accomplish by placing a network card into promiscuous mode. This allows that device to simply collect *all* the data passing through a hubbed network. While this is nice for a hacker, most networks use switches, which inherently restrict this activity.

However, the extra data management on the switch takes time and processing power. The following question then arises: What happens if the switch is asked to process a constant stream of MAC addresses? In certain circumstances and on certain switches, this will cause the switch to go into a fail-safe mode, in which it basically turns into a hub. In other words, by overloading the switch, a hacker could have access to all the data passing through the switch! One tool for doing this is called "macof", which is illustrated in Figure 3. To use "macof", you will need to install the 'dnsiff' suite of tools available at "http://monkey.org/~dugsong/dsniff/".


Konsole - root@localhost:~ - Konsole
File Sessions Settings Help
New
Konsole

```
74412844(0) win 512
ed:85:61:21:52:39 23:d1:4c:26:6a:d8 0.0.0.0.57674 > 0.0.0.0.53397: S 1300147483:
1300147483(0) win 512
9d:34:3f:3a:95:3f a6:e7:34:17:e1:e6 0.0.0.0.5660 > 0.0.0.0.44476: S 1924679314:1
924679314(0) win 512
a:95:7e:1:ae:f8 12:fa:9b:69:ad:d2 0.0.0.0.54690 > 0.0.0.0.18370: S 691576310:691
576310(0) win 512
26:be:69:73:7d:6e a0:af:95:75:0:a9 0.0.0.0.50437 > 0.0.0.0.43058: S 1018604459:1
018604459(0) win 512
3a:68:e5:5e:3e:57 eb:15:49:69:fa:c5 0.0.0.0.49891 > 0.0.0.0.54332: S 577228608:5
77228608(0) win 512
77:40:75:7c:30:88 8d:c9:4f:2b:9a:a0 0.0.0.0.50485 > 0.0.0.0.45185: S 265027417:2
65027417(0) win 512
4f:85:16:17:b4:c0 6e:d:73:6d:9b:41 0.0.0.0.55655 > 0.0.0.0.11963: S 2100817091:2
100817091(0) win 512
e:41:4c:2d:1a:b1 e6:f4:20:24:8f:6d 0.0.0.0.12666 > 0.0.0.0.47609: S 2108239290:2
108239290(0) win 512
3e:27:e9:e:4f:e8 e4:c:6f:1e:4:2b 0.0.0.0.29287 > 0.0.0.0.19999: S 1932772224:193
2772224(0) win 512
da:dd:a8:7c:19:53 be:be:82:3:64:55 0.0.0.0.5933 > 0.0.0.0.60955: S 1590426598:15
90426598(0) win 512
4f:fb:95:34:d:49 b7:d9:8b:58:ec:48 0.0.0.0.40979 > 0.0.0.0.59414: S 569885018:56
9885018(0) win 512
e
```

Figure 3:

The macof tool, flooding the LAN with false MAC addresses in hopes of overloading the switch.

While this would be nice for any hacker, it doesn't usually work. Instead, a hacker needs to find a way to control the ARP tables of the Ethernet devices. To illustrate, we will walk through the spoofing of our own ARP table.

Typically, an Ethernet devices ARP table is updated when they request the MAC address of another device, or they need to communicate with another device. This is easy to duplicate by looking at a before and after shot of the ARP table of your computer, which we previously demonstrated. To see this again, click Start, Run and type "CMD" (for Windows NT/2K/XP) or "command" for all other flavors of Windows. If in *nix, you just need to open a shell.

Once the shell window opens, type "arp -a" to see the current ARP table. The following is an example:

```
Address          HWtype  HWaddress          Flags Mask          Iface
192.168.0.1      ether   00:10:DB:14:7B:70   C                  eth0
```

Depending on whether or not you have connected to any other Ethernet devices, you may have more or fewer entries. However, to add one, simply ping a network device that is not listed. To do this, type "ping <ip address>" in the same shell window. When at least one ping has completed, hit Control+C to stop the ping program and then type "arp -a" again. You should now see a new entry, listing the new IP address and its corresponding MAC address. The following is my new ARP table after ping IP 192.168.0.5. Note the different MAC addresses.

```
Address          HWtype  HWaddress          Flags Mask          Iface
192.168.0.1      ether   00:10:DB:14:7B:70   C                  eth0
192.168.0.5      ether   02:07:01:24:29:64   C                  eth0
```

Now that you understand how the ARP table is updated, it is time to start having some fun! The first thing we will do is prove to you that the ARP table can be 'lied' to. To illustrate, let's use the "arp" command again. This time, instead of just listing the ARP entries, we will make a manual, or static entry. In fact, we will tell our computer that the MAC address of the two computers listed in our ARP table are the same. As you know, this is theoretically impossible since the MAC address is supposed to be a globally unique number. To add this entry, use the "arp -s <IP address> <MAC address>" command. In our example, we will type "arp -s 192.168.0.1 02:07:01:24:29:64". Once this is done, we take another look at our ARP table by using the "arp" command yet again. The following is the results of our tinkering with the ARP table.

```
Address          HWtype  HWaddress          Flags Mask          Iface
192.168.0.1      ether   02:07:01:24:29:64   CM                 eth0
192.168.0.5      ether   02:07:01:24:29:64   C                  eth0
```

Do you see the problem? Note that both entries in the HWaddress field are the same! Obviously, we now have a problem. To correct this problem, you only need to use the "arp -d" command to remove all arp entries. You will WANT to do this as soon as possible because incorrect ARP entries will cause havoc for your network connectivity.

WARNING: Playing with ARP tables can cause your network to stop working! Do not do this on a network you do not "own." For example, if you statically replace your gateway's IP address ARP entry with a false entry, you WILL lose Internet connectivity!

At this point, you know that the ARP table can be lied to locally; however, you can also lie to an arp table remotely! In order to do this, an Ethernet device only needs to receive a spoofed, or forged ARP reply packet. While there are many programs available online that can do this, we will demonstrate ARP spoofing using the "arpspoof" program included in dnsiff suite.

To illustrate the power of arpspoof, let's place ourselves in a hacker's shoes (though this may not be the most pleasant of places to be). The following is an illustration of a sample network that a hacker has just gained access to. In this case, they have plugged their computer into two ports off a switch and will be attempting to sniff the data traveling between 192.168.0.3 and the router (gateway) 192.168.0.1. The hacker has the IP addresses of 192.168.0.5 and 192.168.0.6. See Figure 4 to see the general layout of the network. We will also assume that 192.168.0.3, and the router have previously communicated, which means the gateway, switch, and target computer will all have ARP entries.

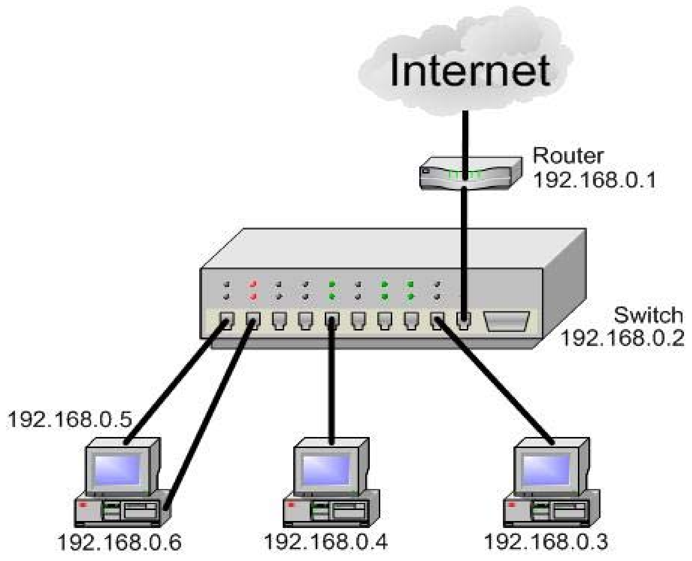


Figure 4:

General network diagram.

Again, the first step a hacker must take is to determine what method they will take to gain access to the data. While ARP spoofing would most likely work, flooding the switch with bogus MAC addresses would take far less time. Therefore, one would assume that this method would be employed first. In other words, a security-conscious network administrator could place a warning system in place that monitored the network for the use of a program such as macof. However, since this works less often than a hacker could hope for, the next step is to play with ARP entries and intercept data flowing between the target and another device (typically the gateway).

> **NOTE**: Before attempting to try this, ensure that IP Forwarding is enabled on the "attacking" computer. Without this enabled, all traffic between target and gateway will be blocked! This a dead giveaway that something is wrong.

To successfully intercept the data, the attacker's computer needs two network cards and an operating system that allows full control over data flow (Linux is the typical choice). This will allow the attacker to communicate with the target on one NIC and the destination point (gateway) with the other NIC. The attacking computer also needs to have IP_Forwarding enabled so that data will pass from one NIC to the other. To do this, type the following in a shell window (Linux):

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Once complete, type "cat /proc/sys/net/ipv4/ip_forward". This should result in a reply of "1".

Step 1 Computer A (192.168.0.4) wants to communicate with gateway (192.168.0.1) to access Internet.

Step 2 Computer A sends out ARP request to gateway requesting MAC address.

Step 3 Switch receives request (which is broadcasted) and passes this request along to every connected computer. Switch also updates its internal MAC address to port table.

Step 4 Gateway receives ARP request from Computer A, and replies with MAC address.

Step 5 Gateway updates internal ARP table with MAC address and IP address of Computer A.

Step 6 Switch receives ARP reply to Computer A, checks its table, and finds Computer A's MAC address listed at port 1. It passes this information to port 1 and then updates MAC table with MAC address from gateway.

Step 7 Computer A receives ARP information from gateway, and it updates it ARP table with this information.

Step 8	Computer A sends information out to gateway using updated MAC address information, and communication channel is established.

At this point, the hacker needs to quickly trick both the gateway and the target computer into passing all information to him. This is handled by opening two shells and executing arpspoof twice (once to trick the target into thinking the hacker's computer has the MAC address of the gateway, and the other into convincing the gateway that the hacker's computer has the MAC address of the target). In other words, the hacker wants to turn his computer into a router, which means all data traveling between the target and the gateway has to first pass through the hacker's computer. Figure 5 illustrates the data flow once this has been accomplished.

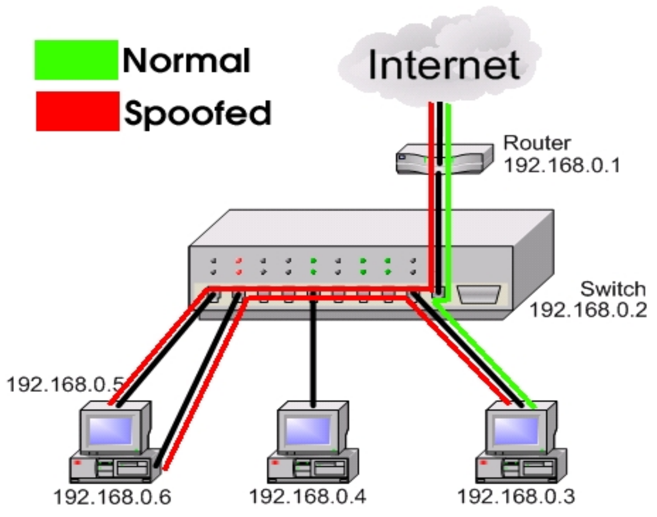


Figure 5:

Data flow using ARP spoofing techniques.

At this point, the hacker owns the data. He can capture it, monitor it, change it, and even perform advanced tricks[md]such as controlling SSL connections to "secure" sites. However, there are ways to detect ARP spoofing.

ARP Spoofing Detection

While stopping ARP attacks is impossible due to the inherent part it plays in data transfer, spoofed ARP requests are very easy to detect. While there are many tools and programs available that attempt to warn administrators of ARP attacks, they all basically work the same way.

One program that does this is *arpwatch*. This program basically monitors all ARP/IP address pairing and alerts its user when changes occur. It does this by listening on the network, much like a sniffer, and comparing all captured replies against a database. Other programs take a snapshot of all related IP/MAC addresses, and periodically request updates from networked computers. However, these methods often result in numerous false alarms due to DCHP networks, which dynamically assign IP addresses.

The only real solution to avoiding ARP attacks is to encrypt all data passing over the network. While this is a possibility, it is not commonly employed due to the processing overhead and complexity of setup.

Summary

ARP spoofing is one of the most useful techniques of gaining access to privileged data that a hacker has. It provides a hacker full control over the data passing between a target computer and a gateway device, or a local server on the network. While there are ways to detect it, stopping it from occurring is practically impossible due to the importance of ARP in a network. The most important lesson that can be learned from this is that your data is never as safe as you may think it is.