Global Information Assurance Certification Paper

Address Resolution Protocol Cache Poisoning

And

Defense

Dan McDaniel
GIAC-GSEC Practical
Version 2

9/15/2004

Abstract

This paper is intended to define the ARP protocol and ARP Cache. What their function in the Ethernet environment is and how they facilitate communication on the network. What ARP Cache poisoning is and the inherent vulnerabilities and compromises that a network might be put into by the malicious use of ARP Cache poisoning. What a network administrator should be aware of and some tools that can be used to both create ARP Cache poisoning packets, detect and defend against these attacks.

Table of Contents

What is ARP??

ARP stands for Address Resolution Protocol. The RFC (Request for Comments) defines ARP in RFC 826 [1] as "The purpose of this RFC is to present a method of Converting Protocol Addresses (e.g., IP addresses) to Local Network Addresses (e.g., Ethernet addresses)."

Communication on an Ethernet network requires 2 addresses on each device. The Machines physical address called a MAC or Ethernet Address and the Internet Protocol Address called the IP address. The Address Resolution Protocol resolves these 2 addresses to each device on the network so that programs can send and receive information from the correct devices.

A MAC address is a unique Hexadecimal number that is burned into the NIC (Network Interface Card) and installed into all devices attached to an Ethernet network. The NIC address defined by IANA[2] (Internet Assigned Numbers Authority) "Is an Ethernet hardware address that is 48 bits, expressed as 12 hexadecimal digits (0-9, plus A-F, capitalized). These 12 hex digits consist of the first/left 6 digits (which should match the vendor of the Ethernet interface within the station) and the last/right 6 digits which specify the interface serial number for that interface vendor."

The IP address assigned to a device can vary and is not permanently associated to it. The IP address as defined by IANA[3] "IP Addresses are numbers. Currently there are two types of IP addresses in active use: IP version 4 (IPv4) and IP version 6 (IPv6). IPv4 was initially deployed on 1 January 1983 and is still the most commonly used version.

IPv4 addresses are 32-bit numbers that range from 0 to 4,294,967,295. They are almost never used or seen in that form. Instead, they are usually represented by the familiar "dotted decimal" notation, with four decimal numbers separated by periods, for example, 192.168.0.1 or 10.0.250.252. Each individual number in a dotted decimal ranges in value from 0 to 255, and thus, in dotted decimal notation, IP addresses range from 0.0.0.0 to 255.255.255.255."

IP addresses can be changed to any value in the networks IP range that is needed on an Ethernet network. But the MAC address on the NIC card is set at the factory and cannot be changed.

[1] http://rfc.net/rfc826.html
[2] Internet Assigned Numbers Authority at http://www.iana.org/assignments/ethernet-numbers
[3] Internet Assigned Numbers Authority at http://www.iana.org/faqs/abuse-faq.htm#StructureofIPAddresses

In order to understand how ARP works we need to have a basic understanding of the Ethernet frame that the data moves in on an Ethernet network. Below is a datagram of the Ethernet frame and its components:

Ethernet frames

When a device wants to communicate on a network it sends its IP packet onto the network in an Ethernet frame.

Ethernet frame datagram IEEE 802.3 standard[4]:

7 bytes	1 byte	6 bytes	6 bytes	2 bytes	46-1500 bytes	0-p	2-4 bytes
Preamble	Start frame	Destination address	Source address	Length/Type field	Data/Payload	Pad	CRC

Preamble:

A sequence of 56 bits (one byte is 8 bits so 8*7 = 56 bits) with alternating 1 and 0 values that the network uses to synchronize the communication. Consider this as the handshake or greeting portion of the transmission.

Start frame:

This is one byte of data that tells the receiving Ethernet card that the data frame is starting. This byte basically tells the receiving machine….get ready here comes the information!

Destination and Source addresses:

The destination address is the address of the NIC card the packet is meant to be delivered to. If the destination address doesn't match the packet is dropped. (We will discuss this later regarding promiscuous mode NIC cards). The source address is of course, the source of the packet.

Length/Type field:[5]

If the value of this field is less than or equal to 1500 bits, then the field indicates the number of bytes in the subsequent MAC Data/Payload field. If the value of this field is

[4] http://standards.ieee.org/getieee802/802.3.html
[5] http://standards.ieee.org/regauth/ethertype/type-pub.html

greater than or equal to 1536, then the Length/Type field indicates the nature of the MAC client protocol (protocol type).

Data/Payload:

This field holds the data that is being transferred from the source device to the destination device. The minimum size of the field is 46 bytes and the maximum size of the field is 1500 bytes. If the field is smaller than 46 bytes than the "Pad" field is necessary to make the frame size equal to the minimum frame size. This is where the info goes that is being sent from one program to another. Consider this the cargo area of the frame.

Pad:

If the Data/Payload field size is smaller than 46 bytes than this field is used to bring the frame up to its minimum size. A minimum Ethernet frame size is 64 bytes from the Destination MAC Address field through the CRC field.

CRC:

This is the cyclical redundancy check (CRC) values used for error checking the frame. When the source device assembles the frame is calculates the bits in the frame from destination field to the payload field and stores the value in the CRC field. Consider this the caboose of the frame. It is also the verifier of the frame so the destination device knows it got all the data in the frame that the sending device sent.

I've always viewed frames this way. They are kind of like the freight trains of the network world. They carry freight (IP packets in this case) to their destination then unload them. This is why ARP exists. It acts like the ticket agent for the Ethernet frame loading the freight onto the frame going to the correct device.

ARP Cache

Before a device transmits a frame of data onto a network it needs to know the correct MAC address associated with the IP address in the frame. So each device builds a temporary table of associated IP and MAC addresses called an ARP cache. This cache is created and updated as the device sends requests for addresses and receives responses then stores them in its table.

To view the ARP Cache on a Windows 2000 machine open up a command prompt and type ARP –a[6].

6

http://www.microsoft.com/windows2000/en/server/help/default.asp?url=/windows2000/en/server/help/sag_TCPIP_pro_ArpCache.htm

ARP Cache on a Linux[7] machine is not as structured as it is not a proprietary operating system. Arp –na at root shell will show the ARP table but Linux has many more configuration options for the user to use at the kernel level. The ARP protocol functions the same for both Windows and Linux on an Ethernet network.

There are 4 basic ARP network communications:

1. **An ARP Request**. Computer "A" puts a request onto the network to find out what MAC address has a certain IP address. "Who has IP XXX.XXX.XXX.XXX? Tell computer "A""

2. **An ARP Reply**. Computer "B" replies to Computer "A"s request, "I have that IP. My MAC address is [XX:XX:XX:XX:XX:XX]."

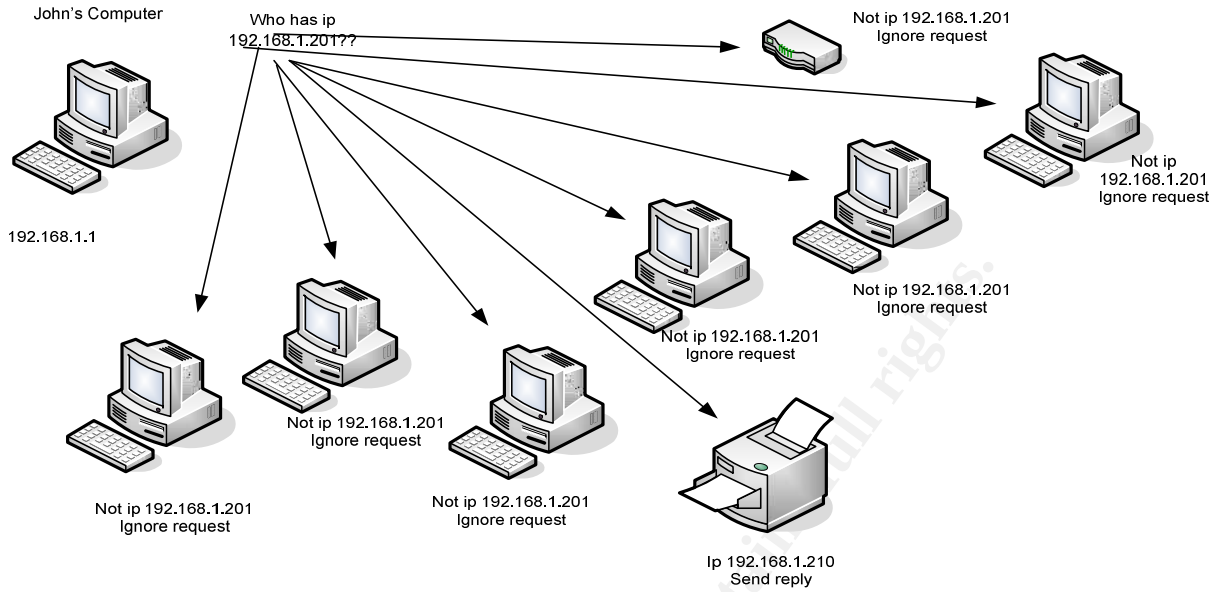
3. **A Reverse ARP Request (RARP)**. Same concept as an ARP request, but Computer "A" asks, "Who has this MAC address [XX:XX:XX:XX:XX:XX:]?"

4. **A RARP Reply**. Computer B tells Computer A, "I have that MAC address. My IP address is [XXX.XXX.XXX.XXX.]"

The following is an example of an ARP request on a network. John in accounting wants to print an excel spreadsheet from the network Hewlitt Packard printer. John's computer sends the print packet from his Excel program to its Ethernet card addressed to the HP Printer. John's computer then looks it up in its ARP Cache to see if it already has the HP printers IP address associated with a MAC address in it. Since it is the first time today he is sending a packet to the HP Printer it is not in the ARP Cache so John's computer needs to get the HP printer's MAC address so it knows where to send the print job. To do this it broadcasts out an ARP request "Who has IP address 192.168.1.201??" see figure1.

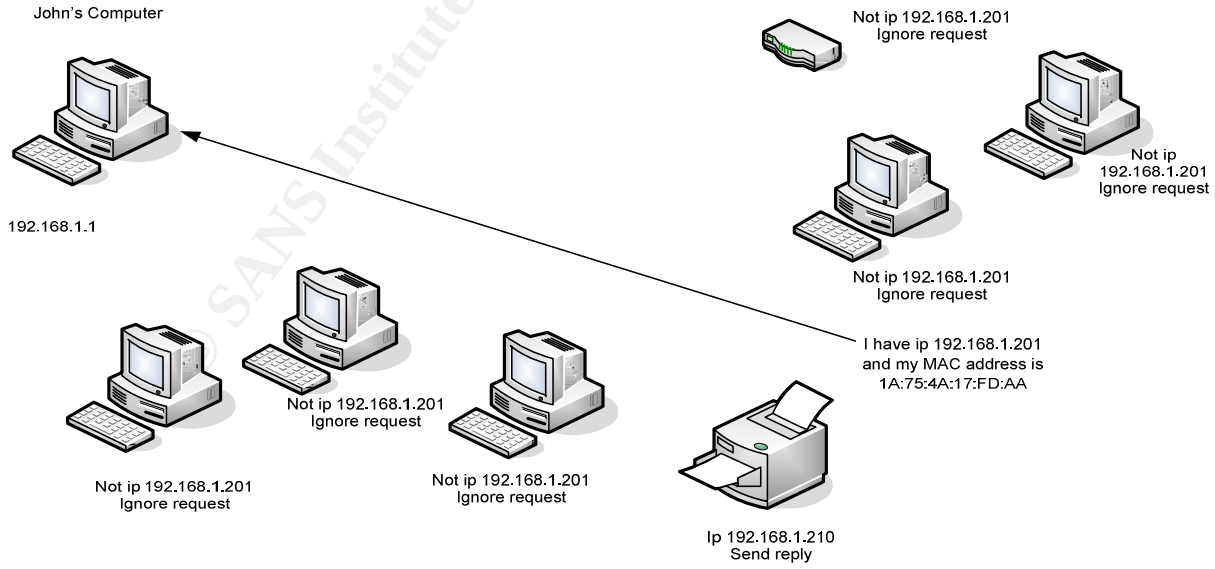
[7] http://www.die.net/doc/linux/man/man7/arp.7.html

Figure 1



All the network devices receive the request. All the devices except the one that has the IP address John's computer requested drop the packet and ignore it. But the HP printer recognizes the request of the IP address 192.168.1.201 as its own and responds to John's computer "My MAC address is 1A:75:4A:17:FD:AA" See Figure 2.

Figure 2



Now that John's computer has the HP printers MAC address it can send the print job to the correct printer. John's computer also updates its ARP cache with both the HP

printer's MAC address and its I.P address. So the next time John wants to print to the HP printer, John's computer will not have to make another ARP broadcast, it will simply send the print job to the HP printer using the MAC and IP address information stored in its ARP cache. Using the ARP cache saves a lot of bandwidth on a network by reducing the number of request packets and helps keep a network running faster.

A simpler way of looking at ARP would be like this. A houses address could be 1125 Main St. But the occupant might change at the house. So the address might be John Doe at 1125 Main St. Or the address might be Jane Doe at 1125 Main St. The house address, like the machine address, never changes. It is constant.

The following is a snippet from an Ethereal[8] file capture showing actual ARP requests on a network:

No.	Time	source	Destination	Protocol	Protocol Info
3	3.583642	192.168.1.4	Broadcast	ARP	Who has 192.168.1.48? Tell 192.168.1.4
4	6.755983	192.168.1.3	Broadcast	ARP	Who has 192.168.1.178? Tell 192.168.1.3
5	8.574922	00000000.0060b0793129	00000000.Broadcast	IPX SAP	General Response
6	13.405968	192.168.1.31	Broadcast	ARP	Who has 192.168.1.218? Tell 192.168.1.31
7	14.154717	192.168.1.3	Broadcast	ARP	Who has 192.168.1.22? Tell 192.168.1.3
8	21.098804	192.168.1.3	Broadcast	ARP	Who has 192.168.1.22? Tell 192.168.1.3

This is only a partial summary of a captured session but lets look at it and see what is going on before we go any further. The Destination of these packets is "Broadcast". This is done by the source NIC putting ff:ff:ff:ff:ff:ff in the destination portion of the Ethernet frame. All devices on the network accepts the packet and then checks the IP request and if it matches will respond, if not it drops the packet.

The ARP requests are pretty much up front and fairly easy to follow the way Ethereal displays them. What I find interesting is the language that Ethereal uses. In the above example the protocol info says "Who has XXX.XXX.XXX.XXX?? Tell XXX.XXX.XXX.XXX". The "Who" is implying that the MAC's are the stable devices and the IP addresses are the ephemeral identifiers. Although we constantly hear about IP addresses, being aware of the MAC addresses on an Ethernet network is just as important.

ARP and Authentication

Did you notice that when John's machine received the reply from the HP printer there was no verification that this was indeed the correct device?? This is because ARP does not require any authentication. When networking was in its infancy and the protocols were being established, the creators were more concerned about networks working as efficiently as possible and not too concerned about security. This may be why ARP does not have any authentication methods in place.

[8] www.Ethereal.com

We also need to remember that when ARP was being implemented it was at a time when hackers were not as sophisticated and the ARP protocol being a low level protocol and seemingly benign to the security of any data as the protocols higher up on the OSI[9] model would secure the information. Networks were also relatively small and users were not very sophisticated in their knowledge of networking and for the most part just worked within the parameters they were given.

Whenever a network device sends an ARP request onto the network it blindly accepts whatever reply that comes back and updates it's Cache with the information never checking to verify if the information it receives is indeed valid. As the protocols standards were being created and networking was growing NIC cards used to act the way they were intended, they always sent out the MAC address imbedded in it to the resources on the network. But once NIC manufacturers began making NIC's that could be set into promiscuous mode and hackers were able to discover how to turn on promiscuous mode, which allows their NIC to receive all ethernet frames that the NIC receives. Where a non promiscuous mode NIC will drop all packets if the destination MAC is not their own.

Many operating systems trust ARP to such a degree that they allow network devices to accept ARP replies without ever sending out an ARP request. This means that if a device has a valid ARP address for another device, a malicious unsolicited ARP reply can be sent to it and it would update its ARP table to make the malicious addresses the one it would use to execute on. The ARP table uses the most recent entry to direct its communication.

So now that it is common knowledge that the ARP protocol will accept any ARP reply or request without verifying the validity of the request. So what?? Creating and sending an ARP package is not something a normal person or user would be able to do right? Wrong. How in the world would anyone be able to recreate such a packet? This is what I think the creators of the protocol were thinking as well as the Operating system designers that sheer complex at the binary level would be beyond the end user.

This is where Ethernet has its drawback. If one person was able to write a program that would create malicious ARP packets to trick devices it indeed wouldn't be much of an issue for the whole IT industry to be concerned about. But what if that person put that same program out on the internet for anyone that wanted it could download and use it?? Now the seemingly small problem has become a very real and potentially damaging situation.

This is exactly what is happening. There are programs that are free to download from the internet such as Cain and Abel [10] and Ettercap[11]. Ettercap requires the user to compile the program and is a bit more difficult to get up and running. Cain and Abel is

[9] http://computer.howstuffworks.com/osi1.htm
[10] http://www.oxid.it/cain.html
[11] http://ettercap.sourceforge.net/

ready to be used right after the program is downloaded and I find it a much easier program to use. Among other mischievous things these programs will allow someone to do, they are mainly ARP cache poisoning tools. ARP cache poisoning is when an ARP cache is malicious altered, or in other words someone tricks an ARP cache into routing packets sent to an IP addresses to the incorrect MAC address.

ARP cache poisoning:

As we just covered, ARP is a very insecure protocol. Any network device can update another devices ARP Cache without any authentication what so ever. This means that chaos on the network is not to hard to make happen.

There are 3 main types of attacks that ARP Cache poisoning is used for:

1. Denial of Service

With one of the many ARP cache poisoning tools a hacker can download from the internet he can use the program to capture all network traffic between two of the devices on a network and execute a denial of service attack.

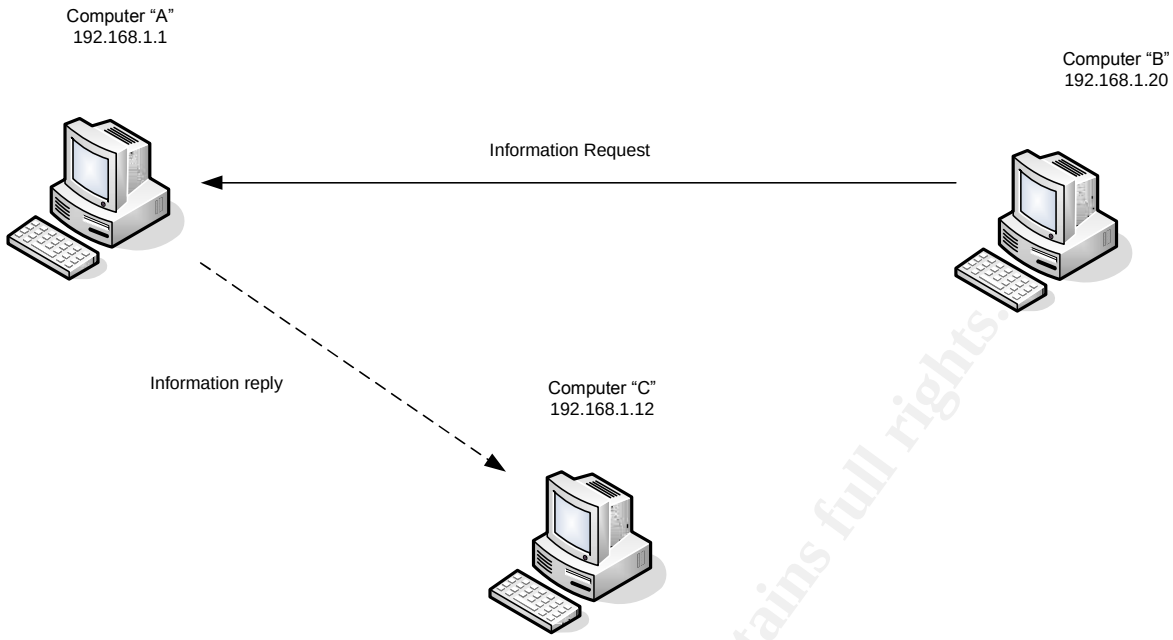
When a hacker executes a denial of service attack using ARP what he does is trick one machine, usually a server or router but not always, to send all data meant for one machine to a machine he is using or has access to.

Figure 3 shows an example of an ARP denial of service attack. Computer "B" sends a request to computer "A" asking for a response. When computer "A" receives the request is wants to return the requested information back to computer "B". But computer "C", the hackers computer, keeps telling computer "A" that computer "B"'s MAC address is actually his. Since there is no authentication and computer "B" has no reason to remind computer "A" what it's MAC address is. Computer "A" sends the information to the latest entry in its ARP cache which is computer "C"'s IP and MAC information.

This means that for every request that computer "B" makes to computer "A" the reply will go to computer "C" as long as the hacker keeps updating computer "A"'s ARP cache with the poisoned information. In fact, computer "C" is denying all packets from computer "A" to computer "B" which is why this attack is called a denial of service.

Let's say computer "A" is the router and computer "B" is requesting information from a website. All the website information will be routed to computer "C" because the router's ARP table is poisoned to think computer "C" is computer "B" and computer "B" is denied any if the information it requested.

Figure 3.



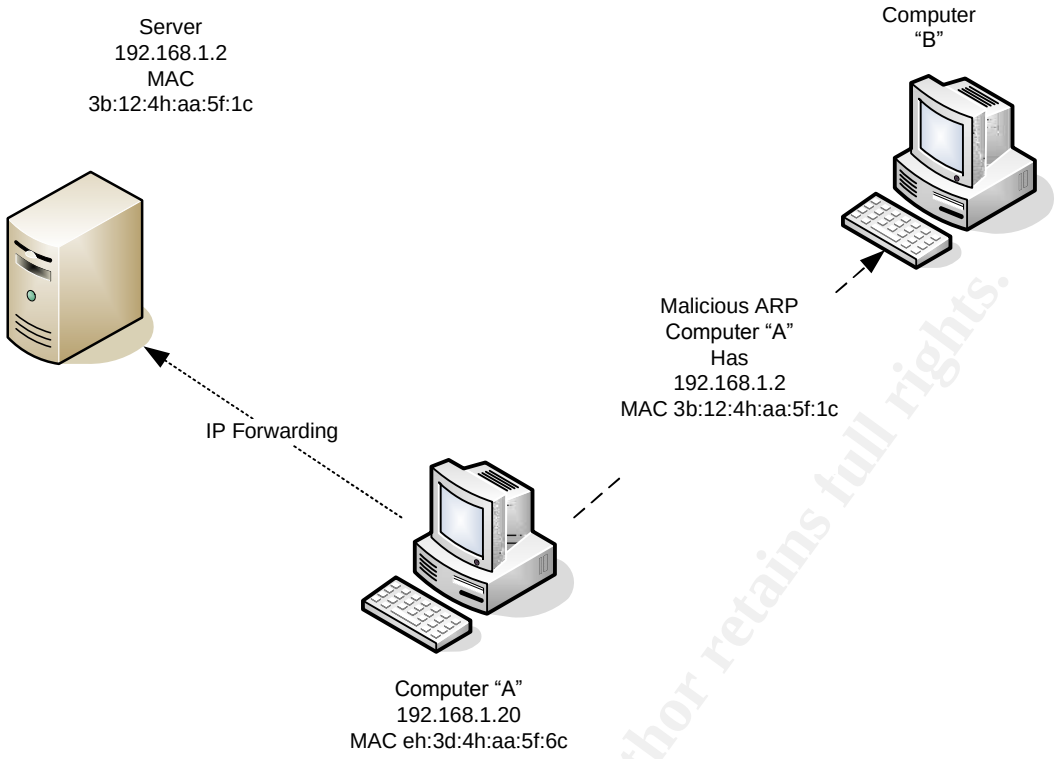
2. Man in the middle attack

The man in the middle attack is also known as Monkey in the middle or MiM is exactly what the name indicates. A hacker takes advantage of two different things to do this.

1. Is ARP cache poisoning. Where the hacker sends a malicious packet to a device to think his machine is another machine. Exactly like the denial of service attack mentioned above.
2. Is IP forwarding. The hacker uses an OS feature called IP forwarding. This allows the hackers machine to forward the IP packets to another machine specified by the hacker.

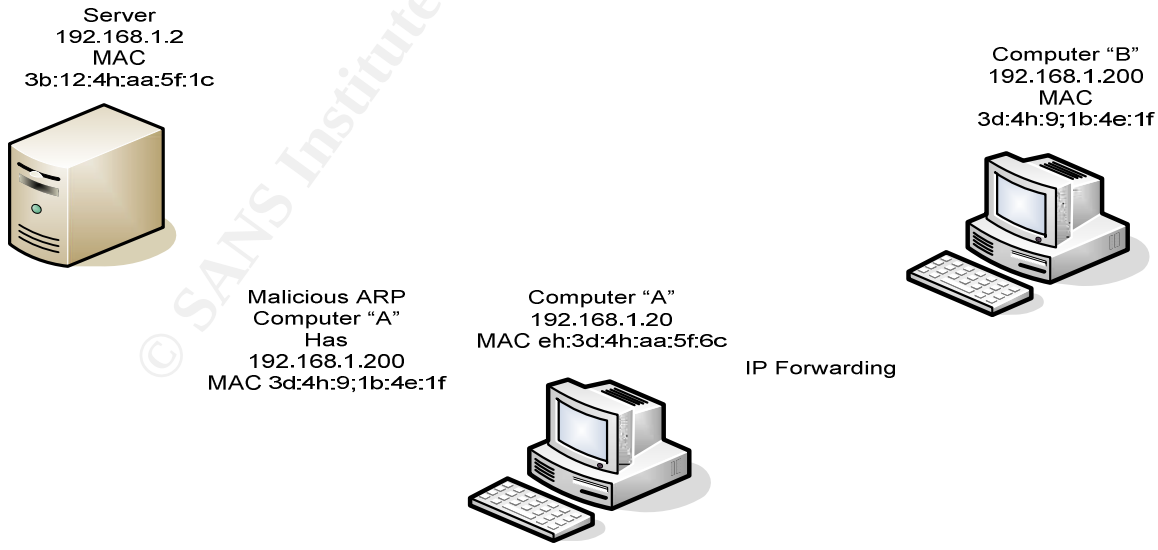
So let's take a look at a typical Man in the middle attack in Figure 4. Computer "A" sends a malicious ARP reply to computer "B" so that it thinks Computer "A" is the server.

Figure 4.



Then the hacker sends a malicious ARP packet to the Server so it thinks that Computer “A” is computer “B”. See figure 5

Figure 5



By poisoning both machines ARP cache. Computer “B” sends a request to computer “A” thinking it is the server and the server replies to computer “A” thinking it is computer “B”.

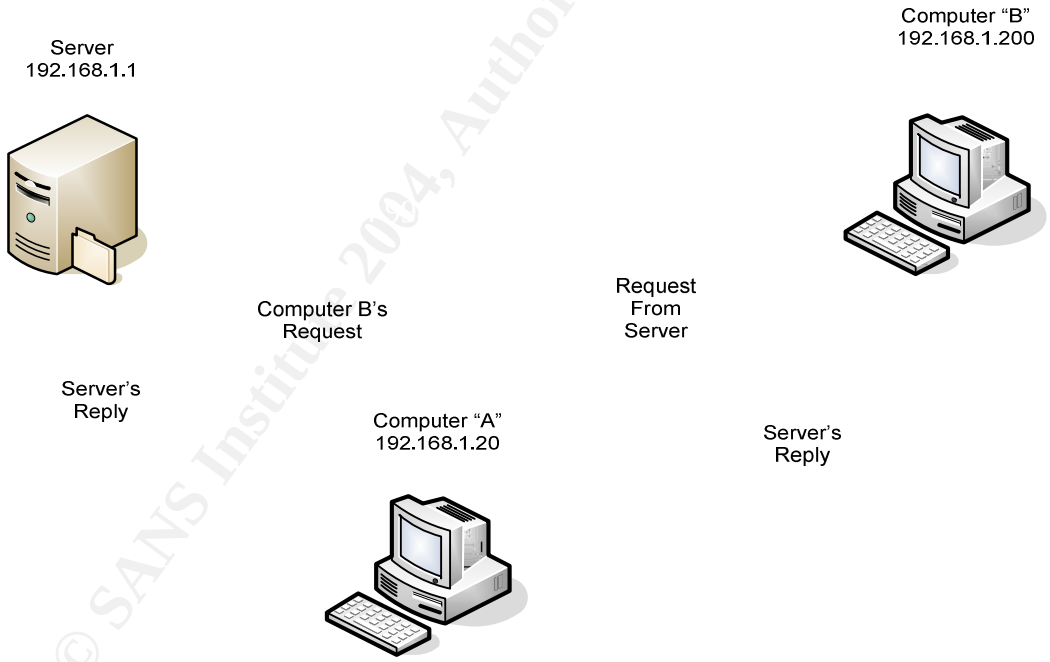
Computer “A” turns on IP forwarding so that the server requests are forwarded to the Server and so that all traffic that comes from computer “B” to computer “A” that is intended for the server will be forwarded from computer “A” to the server.

The same applies to the Server’s response to computer “B”’s request. Because it’s ARP cache is also poisoned it sends the reply to computer “A” thinking it is computer “B”. Then computer “A” uses IP forwarding to forward the packet to computer “B”.

All the while the hacker sitting at computer “A” can read and or save any and all packets of information going between the Server and computer “B”. With many of the free software programs that are available many will put the packets together into their correct content to make it even easier for hacker to read the data.

As this is going on neither the Server nor computer “B” has a clue this is happening as the requests and the replies being sent are both being received and they do not notice anything is amiss. See figure 6.

Figure 6.



3. MAC Flooding

MAC flooding is used to disarm Switches [12] so that hackers can get the Switch to change into HUB mode. This is done because network sniffers have a hard time

[12] http://www.linuxgazette.com/issue42/tag/6.html

gathering any data from a switch as a switch directs packet traffic directly to intended devices using, you guessed it, ARP. Where as a hub broadcasts all data to all devices connected to it waiting for the matching address to accept the data intended for it.

Since a Switch sends packet directly to each machine instead of broadcasting the packets it uses ARP to keep track of what IP request or reply goes with what MAC address and sends the packets directly to the correct MAC address. Once a malicious machine starts inundating a switch with a flood of MAC's some switches will change from switch mode to hub mode just to get the data out to the network.

When and if a switch does change to hub mode the hacker can then begin to sniff the packets on the network freely. It is easier for a hacker to sniff a hub than to manage all the computers IP and MAC addresses and then forward all the traffic.

What can be done??

As we have seen ARP Cache poisoning can be very detrimental to a network. But the good thing is that it is limited to networks only. Since routers do not forward MAC addresses with internet request, ARP is a local network issue only. What that means is that the hacker must either have physical access to your network or have remote access to a machine attached to your network. At least that is good news.

ARP Cache poisoning is possible due to a lack of security in ARP which is a required protocol layer for TCP/IP networking. Since ARP is required on a TCP/IP network there is nothing we can do about the security issue with the protocol other than to be aware of what ARP cache poisoning is and how it works.

If you are using a smaller Microsoft network you can implement the ARP –s [13]command. Using the –s switch will make the ARP cache on the machine static. This means the cache will not change until the machine is turned off. Though that would make the ARP cache more secure it also would make updating it far more difficult. You can write a login script that will add the static ARPs to each machine as they boot but maintaining it can be difficult. If you are using a DHCP[14] server on the network to distribute and manage your IP addresses or if the network devices ever change this will cause more havoc than it is worth.

If you have a network that uses DHCP or devices change such as laptops or vendors coming and going, then from what I have found to be the best tool is a program called ARPWATCH.
(http://www.securityfocus.com/tools/142). This is a Unix/Linux utility that monitors IP/Ethernet mappings and will let you know when changes out of the ordinary are being

[13] http://www.microsoft.com/windows2000/en/server/help/default.asp?url=/windows2000/en/server/help/sag_TCPIP_pro_ArpCache.htm

[14] http://www.dhcp.org/

made. Since ARP is at the Network layer in the OSI model a Unix/Linux machine on your network running Arpwatch is very effect regardless of the NOS. If the network uses a switch instead of hubs check to see if the switch allows for "Port Security". If the switch has Port Security, then the switch can be set to allow only one MAC address for each physical device on each port. This will stop devices from spoofing other devices MAC addresses on the network, but if a hacker is obsessed with using ARP Cache poisoning you may see MAC flooding take place to get beyond the limits of the "Port Security" feature so that is something to watch out for as well.

The most important thing is to know your network and monitor the traffic on it at all times. Be aware of increased traffic and patterns that may appear such as increased ARP requests between 8-10 A.M. This is when most people are logging into a network and if the hacker is after usernames and passwords that would be the ideal time for them to do so.

Also, it is imperative that when using Ethereal or other packet capturing programs that they are monitored and reviewed on a consistent basis. You will need a few weeks of logs to set up a historical baseline of activity to see whether or not you have a spike in ARP activity.

Summary

In order to be successful at reducing the inherent exploitability of the ARP protocol it is important that defense in depth be applied on networks. The ARP protocol and the ARP Cache is part of the functionality of an Ethernet network, it will be part of networking for a long time to come. But understanding how Ethernet frames work and the purpose of ARP to coordinate the IP and MAC addresses will help with detecting ARP cache poisoning. Using the tools described in this paper to monitor and track network traffic and knowing what vulnerabilities to look for on a network can make a network more secure.

References:

RFC 826 ARP Definition: http://rfc.net/rfc826.html

IANA NIC Assignment: http://www.iana.org/assignments/ethernet-numbers

IANA IP Definition: http://www.iana.org/faqs/abuse-faq.htm#StructureofIPAddresses

Windows ARP Cache:
http://www.microsoft.com/windows2000/en/server/help/default.asp?url=/windows2000/en/server/help/sag_TCPIP_pro_ArpCache.htm

Linux ARP Cache: http://linux.about.com/library/cmd/blcmdl7_arp.htm

OSI Model: http://computer.howstuffworks.com/osi1.htm

Switch: http://www.linuxgazette.com/issue42/tag/6.html

IEEE 802.3 Standard: http://standards.ieee.org/getieee802/802.3.html

Length/Type Field: http://standards.ieee.org/regauth/ethertype/type-pub.html

ARP Cache Lifetime:
http://www.microsoft.com/windows2000/en/server/help/default.asp?url=/windows2000/en/server/help/sag_TCPIP_und_arp.htm

DHCP: http://www.dhcp.org/

TOOLS:

Ethereal: www.Ethereal.com

Cain and Abel: www.oxid.it/cain.html

Ettercap: http://ettercap.sourceforge.net

Arpwatch: http://www.securityfocus.com/tools/142