

## ARP spoofing and poisoning

## TRAFFIC TRICKS

Any user on a LAN can sniff and manipulate local traffic. ARP spoofing and poisoning techniques give an attacker an easy way

in. **BY THOMAS DEMUTH, ACHIM LEITNER**

**C**uriosity, revenge, industrial espionage are all reasons why insiders attack systems on their own network. Statistics show that 70 to 80 percent of all attacks originate on the internal network [1]. Admins have a hard time preventing these internal attacks because protecting the internal network is a lot more difficult than protecting against external attack.

One of the most formidable forms of internal attack is known as ARP spoofing. ARP spoofing puts an attacker in a

position to sniff and manipulate local traffic. So-called man-in-the-middle attacks are easy to perform, and thanks to sophisticated software, even attackers with little knowledge of networking stand a good chance of succeeding.

### How ARP Works

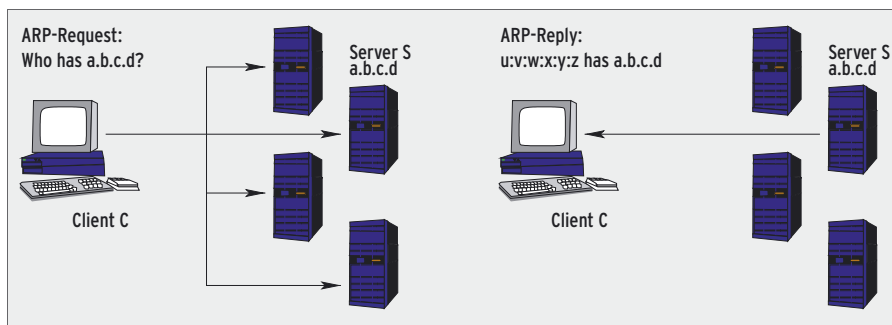
The ARP protocol was published in November 1982 by David C. Plummer as RFC 826 [2]. As IT security was not an important factor back in 1982, the aim was simply to provide functionality. ARP

maps IP addresses to MAC addresses. If client C needs to send a packet to server S, it needs to know the MAC address of S if both machines are on the same subnet. Even if S resides in a different network, C still needs a MAC address – in this case, the address of the next router that will forward the packet. The router takes care of everything else.

To ascertain the MAC address, C broadcasts an ARP request to all the machines on the local network, asking “Who has the IP address a.b.c.d.?” The computer with the matching number replies and tells the client its MAC address (Figure 1).

As shown in Figure 2, an ARP packet is carried as the payload in an Ethernet frame. To allow this to happen, a value of 0x8006 is set in the frame header type field – this tells the target to expect an ARP packet.

As it would be far too expensive to broadcast an ARP request and wait for the response before sending data, each IP stack has an ARP table, also known as an ARP cache (Figure 3). The cache con-



**Figure 1:** The client uses ARP to ascertain the MAC address of the server on the LAN before sending a packet to that server. The “Who has...” request is broadcast to all machines on the LAN. The node with the requested address responds directly to the querying machine.

tains a table with IP addresses and corresponding MAC addresses. The table can hold static entries (i.e., those generated by the user) and dynamic entries (those learned from the ARP protocol). Dynamic entries are often valid for a short period only, typically just a few minutes.

## Addressing Attacks on the LAN

As ARP makes no attempt to protect itself against spoofed packets, it is vulnerable to a series of attacks. The most common types are MAC spoofing, MAC flooding, and ARP spoofing.

MAC spoofing involves the attacker using a spoofed MAC source address. This technique makes sense if privileges are linked to a MAC address. Many WLAN (Wireless LAN) operators put the MAC addresses of authorized users in an access control list. This is a weak security measure that is easy to avoid. The attacker only needs to know and spoof a privileged address while the machine used by the legitimate user with this address is down. MAC spoofing is useful for attackers who want to protect their identity.

There is a good way of preventing this on wired networks: many switches enable port security. The switch only learns each MAC address once, and then stores this address permanently. From

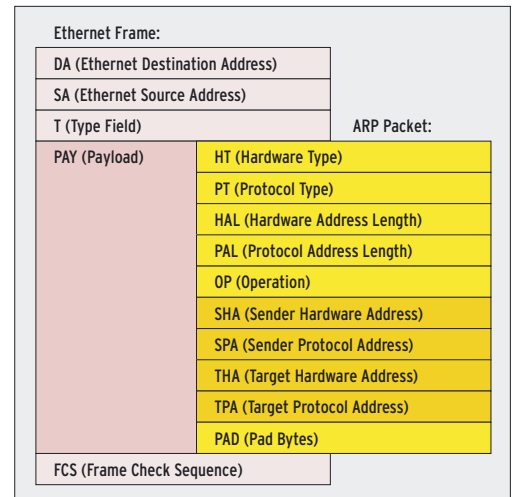
this point onward, the switch will not accept any other source MAC address on the mapped port. This mechanism is effective against MAC spoofing attacks. On the downside, admins need to reconfigure the switch whenever they change the network.

Port security can also protect your network against another kind of attack. MAC flooding attacks are designed to take down a switch's port security mappings. In contrast to hubs, switches use CAM (Content Addressable Memory) tables, which specify the port behind each active MAC address on the switch. The switch will only send packets via the port that leads to the target machine.

Attackers can disable this function by flooding the switch with addresses – the CAM table can only hold a limited number of entries. If the attack succeeds, the switch is reduced to working like a hub, and that makes communication visible across all ports.

## ARP Table Poisoning

The third attack is not as easy to detect, and there are no simple countermeasures. The attack is based on ARP spoofing, where the attacker deliberately transmits fake ARP packets. ARP poisoning is a specific type of ARP spoofing



**Figure 2:** An ARP packet is transmitted as the payload of the Ethernet frame. The fields with the type and length of the addresses in each packet are followed by the source and target data.

that aims to manipulate (poison) the ARP tables on other machines.

As operating systems tend not to check if an ARP reply really is the answer to an ARP request sent previously, the address information from the reply is cached. On Windows systems attackers can even modify entries explicitly declared as static by users.

Doing so allows an attacker to monitor the dialog between a client and a server, and, as the man in the middle, to manipulate that dialog. The man in the middle manipulates the server entry in the cli-

## Addresses on the LAN: Basics

If two computers on a network want to talk, they need a way of identifying each other uniquely. Ethernet uses a 48-bit (6 byte) number, which is assigned by the manufacturer. The so-called MAC address (Media Access Control) is unique world wide. This allows users to add (more or less) as many Ethernet adapters as they like to a LAN. Without switches or bridges Ethernet uses broadcasting; that is, every packet on the wire is sent to every node on the network segment. But only the intended recipient will actually accept the packet, whereas all other nodes will ignore it.

This approach is amazingly easy, but it does not scale well. Everyone attached to the shared medium shares the transmission bandwidth. Bridges and switches mitigate the situation by dividing the network into multiple segments and learning which MAC addresses are available via which ports (CAM table,

Content Addressable Memory). This allows these devices to transmit packets only to the segment where the recipient lives. Within each segment, network nodes can send each other packets without interfering with communications in other segments.

This principle is unsuitable for a world wide network. Each switch needs to know the whereabouts of each target machine. To handle this, the founders of the Internet introduced an addressing scheme based on IP addresses. The IP address has a length of 32 bits (4 bytes) and comprises a network and a host section. The network mask tells you which part of the address refers to the network and which part identifies the host.

The individual networks that make up the Internet are connected by routers. Routers only need to know network addresses to send packets in the right direction. While routing relies on IP

addresses, the LAN continues to use only MAC addresses. But it would be inconvenient for each program to need to know both the IP address and the MAC address. This is where ARP (Address Resolution Protocol) can help by providing the matching MAC address for an IP address. The admin does not need to configure this – that is, there is no need to set up matching pairs of IP/MAC addresses. On the downside, automation leads to a big security issue, which we will be discussing in more detail in this article.

Besides ARP, there is also RARP (Reverse ARP, [3]). In a similar way to DHCP, a RARP server assigns an IP address to a machine based on knowledge of that machine's MAC address. As RARP does not pass any other parameters (name server, gateway address, network mask), it is very rarely used nowadays.

```

arp -n
Table -- # entries
  IP address      HW address      Flags Match  I/Face
-----
192.168.1.100    08:00:27:00:00:00  C           0         eth0
192.168.1.101    08:00:27:00:00:00  C           0         eth0
192.168.1.102    08:00:27:00:00:00  C           0         eth0
192.168.1.103    08:00:27:00:00:00  C           0         eth0
192.168.1.104    08:00:27:00:00:00  C           0         eth0
Data -- #

```

**Figure 3: The ARP table on a Linux system with one incomplete entry, one static entry, and two dynamic entries (Flag C: complete, M: static).**

ent's ARP cache, making the client believe that the attacker's own MAC address is actually the server address. The same trick is also played on the server.

If the client wants to talk to the server, it will check its poisoned ARP table and send the packet to the attacker's MAC address. This allows the attacker to read and modify the packet before forwarding it to the server. The server then assumes that the packet was sent directly by the client. The server reply again goes to the attacker, who forwards it to the client. If the server resides on a different subnet, the attacker can simply launch the attack against the router.

Of course, it goes without saying that an attacker could cause a denial of service by simply discarding any re-routed packets. To manipulate data, the attacker simply needs to forward different data than he or she receives. Attackers can trivially collect passwords, as the port number will allow them to guess the protocol used and identify the user credentials based on this knowledge.

## Caution even with SSL and SSH

Encrypted connections are not automatically immune, as various ARP tools demonstrate. These programs are now available for various operating systems (see the box titled "ARP Exploit Tools"). Besides ARP poisoning functionality, they include client and server implemen-

tations for SSL (Secure Socket Layer), TLS (Transport Layer Security), SSH (Secure Shell), or PPTP (Point to Point Tunneling Protocol).

On accessing the SSL web server, the

browser warns the user that something is wrong with the certificate for this connection. But many users do not understand the significance of the warning and just ignore it. The fact that many web servers use a self generated or elapsed certificate means that the warning is quite familiar to users; in fact, it just tends to provoke a click and ignore effect. A bug in some Internet Explorer versions makes it possible to attack SSL connections without the browser even displaying a warning.

The attack on SSH follows a similar pattern (Figure 4). If the client already knows the server-side host key, it will issue a clear warning (Figure 5). But many users and admins will still ignore the warning, assuming that someone has changed the SSH key on the server. Few protocols or implementations are immune. (IPsec is one exception. IPsec refuses to work if something goes wrong with the authentication process.)

Because of this problem, almost any kind of internal communication is vulnerable. There are even script kiddie tools that can grab passwords for over 50 protocols. As these attacks run at ARP level and typically only IP access is logged, today's attackers can feel quite safe that nobody will notice what they are up to.

## Preventing ARP Attacks

One approach for preventing ARP

attacks would be to prevent downloading and execution of external software, although this rule is extremely difficult to enforce. Admins would need to restrict the

use of the Internet connection. HTTP, HTTPS, FTP, and email make it easy for an attacker to smuggle malware onto the internal network. Admins would also need to outlaw the use of removable media such as floppies or CDs, as well as mobile devices such as notebooks or PDAs. Due to the serious usability restrictions, this solution is typically impracticable.

If you use Linux on your internal network and do not allow your users root privileges, you can avoid most attacks from the outset: users need root privileges to send spoofed ARP packets. However, as an admin, you have no real way of preventing users from booting their machines from CD or attaching their laptops to the network.

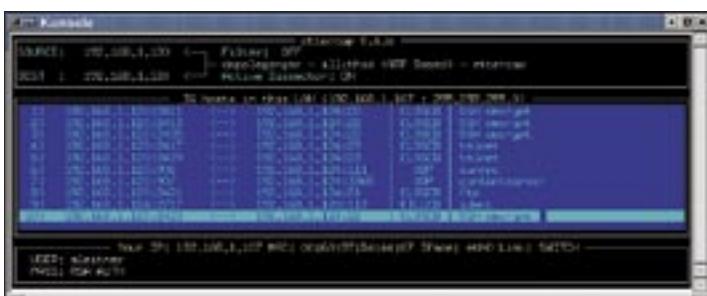
Static ARP entries can help to prevent ARP attacks, but most admins will want to avoid the enormous effort involved in manually assigning addresses for all but the most critical machines (routers and servers). And Microsoft operating systems allow attackers to poison even manually assigned ARP entries, removing any protection this might give you.

This approach only makes sense in small networks, as the number of ARP entries grows proportionally to the square on the number of network adapters. In other words, you would need 9900 entries for a hundred systems (99 for each system). This involves enormous administrative effort, especially if you need to troubleshoot problems on the network.

## Keeping a Watchful Eye

Arpwatch [4] is an open source tool for Unix platforms that monitors unusual ARP activities. The computer running Arpwatch reads the address information stored in each ARP packet that it sees and stores this information in a database. If the data fails to match previously stored entries, Arpwatch mails an alert to the administrator. The authors claim that the tool supports SNMP, although we were unable to confirm this in our lab.

Today, many networks use dynamic IP addresses assigned by DHCP (Dynamic Host Configuration Protocol). In this kind of environment, Arpwatch will return large numbers of false positives as it notifies users of any changed IP/MAC mappings.



**Figure 4: Ettercap waiting for a connection between 192.168.1.120 and 192.168.1.124 (source and destination, top left). The tool can trivially sniff telnet and FTP. It uses a man in the middle attack on SSHv1 to decrypt the connection.**

```

odo@odo:~$ ssh -i bashir
Warning: Remote host ID:192.168.1.100 has changed!
It is possible that someone is doing something nasty!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the host's host key has just been changed.
The fingerprint for the host's key does not match what you saved in
/etc/ssh/ssh_known_hosts (192.168.1.100).
Please contact your system administrator.
Add correct host key to /home/odo/.ssh/known_hosts to get rid of this message.
Offending host key in /home/odo/.ssh/known_hosts:10
Warning: Permanently added host key to avoid future warnings.
Warning: Permanently added host key to avoid future warnings.
odo@odo:~$

```

**Figure 5:** During the Ettercap attack (Figure 4), the client (odo in this case) receives a fake host key from the server. The key is from the attacker and not from the requested server (bashir). If the user chooses to ignore the warning, the connection can be sniffed.

ARP-Guard [5], a fairly new product by ISL, works within the framework of a sensor management architecture. Multiple sensors monitor ARP information and forward this information to the management system, which analyzes messages and alerts administrators in case of an attack. The architecture means that ARP-Guard will scale well from small to large networks, and the web-based management interface is something that command-line challenged admins will appreciate.

Intrusion Detection Systems (see the box titled “Snort and ARP”) are also capable of detecting ARP attacks, but these systems are more typically deployed at network borders. For many businesses, it is simply not worth deploying an IDS on the internal network. Additionally, employees might resent the IDS system administrator playing Big Brother on the network. The administrator can see all the traffic on the network and thus monitor access by

staff. The usefulness of this approach is also questionable, as many IDS systems simply ignore the ARP protocol. And finally, the whole system would collapse faced with ARP poisoning attacks in combination with dynamically assigned IP addresses.

ARP-Guard has LAN and SNMP sensors. The LAN sensor works just like Arpwatch or any IDS system, analyzing any ARP packets that the sensor sees. In contrast to this, the SNMP sensor uses SNMP to connect to existing devices and query their ARP tables.

## Cryptography Can Help

Wherever cryptographic protocols (IPsec above all) ensure the confidentiality, authenticity and integrity of data, ARP attacks are reduced to mere denial of service attacks. Any attempt to sniff or manipulate data will fail. However, it may be a long while before IPsec and other crypto-protocols are set up and correctly configured on internal networks.

One group of researchers suggests replacing ARP with a more secure version [7]. S-ARP relies on cryptography, a CA (Certification Authority), and digitally signed ARP messages. However, it is questionable whether the effort is really worthwhile: IPsec provides far more protection with a similar overhead, whereas

S-ARP only protects ARP. The only thing S-ARP has going in its favor is that it means less CPU load on the systems.

## Other Prevention Techniques

Some firewalls and router manufacturers maintain that their products are capable of detecting ARP spoofing attacks, but this is not strictly true as these systems can only detect and log modifications to their own ARP tables and have no way of knowing whether the change has a legitimate cause.

Dividing the network into a large number of subnets and assigning a small

number of users to each subnet can help to limit exposure to ARP attacks. Manageable switches, which allow administrators to manage network traffic, give protection against ARP attacks and serve as a boon to traffic management as well. On the downside, a manageable switch is expensive, adds administrative overhead, and may have the effect of breaking some applications.

Some developers attempt to add protection to the IP stack on the terminal devices. The Antidote patch [8] tells a Linux machine to send a request to the previous MAC address before changing an ARP entry. The machine will only

change the entry if the request to the previous address is not answered. Again, this approach does not give you any real protection against sabotage. The attacker simply needs to ensure that the attack occurs when the machine with the previous MAC address is down or unreachable. In the case of many high availability or load balancing solutions, the patch can actually cause communication to these systems to fail.

Another approach to protecting yourself against ARP poisoning is to prevent reassignment of existing MAC-IP mappings. The Anticap patch [9] implements this behavior for Linux, FreeBSD, and NetBSD. Solaris has a similar option, which requires a timer to expire before applying a change. This behavior is freely configurable, however, a solution such as the Anticap patch only protects systems that are permanently up – attackers would have no trouble spoofing new entries once the cached entries expire.

Linux with kernel 2.4 or newer no longer reacts to unsolicited ARP replies. Unfortunately, this mechanism is easy to circumvent, as the Ettercap readme file explains. The kernel always has to process ARP requests. As the kernel is passed a combination of IP address and MAC address (for the source), it proactively adds this data to its ARP cache. So the attacker only needs to send a spoofed ARP request. Ettercap sends a combined ARP request and reply, and

## ARP Exploit Tools

Following are some programs that allow attackers to exploit ARP vulnerabilities. Admins can use these tools to test their own networks. The tools are quite useful for demonstrating the severity of ARP attacks. The real security problem, of course, is not the fact these tools exist, but that ARP is inherently insecure.

**ARP-SK:** The programmers describe their tool as the Swiss Army Knife for ARP; it is available in Unix and Windows versions. The program can manipulate ARP tables on various devices. <http://www.arp-sk.org>

**Arpoc and WCI:** This program for Unix and Windows performs a man in the middle attack on a LAN. It replies to each ARP request that reaches the machine with a spoofed ARP replay and forwards any packets for non-local delivery to the appropriate router. <http://www.phenoelit.de/arpoc/>

**Arpoison:** A command line tool that creates spoof ARP packets. The user can specify the source and target IP/MAC addresses. <http://arpoison.sourceforge.net>

**Brian:** This extremely simple tool (comprising a single C file) uses ARP poisoning to disable switching on a LAN. This allows an attacker to sniff all the traffic on the network. <http://www.bournemouthbynight.co.uk/tools/>

**Cain & Abel:** This sophisticated Windows software started life as a password recovery tool. It sniffs the network and uses a variety of techniques to decipher encrypted and obfuscated passwords. Version 2.5 of the tool was the first to introduce ARP poisoning, which allows the attacker to sniff IP traffic off a switched LAN. The program attacks SSH and HTTPS connections. <http://www.oxid.it/cain.html>

[oxid.it/cain.html](http://www.oxid.it/cain.html)

**Dsniff:** The individual programs in this suite of tools fulfill various tasks. Dsniff, Filesnarf, Mailsnarf, Msgsnarf, Urlsnarf and Webspny sniff the network and grab interesting data (such as passwords, email and files). Arpspoof, Dnsspoof, and Macof allow admins and attackers to access data that a switch would normally protect. Sshmitm and Webmitm support man in the middle attacks on SSH and HTTPS (although the author refers to them as Monkey in the Middle attacks). <http://naughty.monkey.org/~dugsong/dsniff/>

**Ettercap:** An extremely powerful program with a sharp text-based interface (see Figure 4); the latest version also has a Gtk interface. Actions are performed automatically, with the tool listing potential targets in a window. Besides Sniffing, ARP attacks, and automatic password grabbing, Ettercap can also manipulate data within a connection. The program also attacks SSHv1 and SSL connections (again using man in the middle techniques). <http://ettercap.sourceforge.net>

**Hunt:** Gatecrashes connections, sniffs data, and hijacks sessions. The tool uses ARP spoofing and other techniques. <http://packetstormsecurity.nl/sniffers/hunt/>

**Juggernaut:** In 1997, Phrack Magazine published Juggernaut, a predecessor to many of today's sniffers with ARP cache poisoning capabilities. <http://www.phrack.org/show.php?p=50&a=6>

**Parasite:** The Parasite daemon sniffs the LAN and responds to ARP requests with spoofed ARP replies. The tool gradually allows the host machine to establish itself as a man in the middle for any communications on the LAN. <http://www.thc.org/releases.php>

## INFO

- [1] KPMG survey: <http://www.kpmg.com/about/press.asp?cid=469>
- [2] Address Resolution Protocol, RFC 826: <http://www.ietf.org/rfc/rfc826.txt>
- [3] Reverse APR, RFC 903: <http://www.ietf.org/rfc/rfc903.txt>
- [4] Arpwatch: <http://www.nrg.ee.lbl.gov> und <http://www.securityfocus.com/tools/142>
- [5] ARP-Guard: <https://www.arp-guard.com>
- [6] Snort: <http://www.snort.org>
- [7] Secure ARP: <http://security.dico.unimi.it/research.en.html#sarpd> and <http://www.acsac.org/2003/papers/111.pdf>
- [8] Antidote patch: <http://www.securityfocus.com/archive/1/299929>
- [9] Anticap patch: <http://cvs.antifork.org/cvsweb.cgi/anticap/>

any system will respond to one of these techniques.

Protection built into the IP stack is powerless to prevent ARP spoofing. If an attacker responds to an ARP request more quickly than the machine to which the request is addressed, the attacker

wins the race and the attacker's address is added to the ARP table.

## No Protection

Today's techniques can't give you complete protection against ARP attacks, but you can arm yourself with IDS and spe-

cialized ARP manipulation sensors to detect most manipulation attempts. To be completely secure, you need to deploy IPsec consistently on your network. Ignoring the issue is not a good option unless you can genuinely trust every user with access to your LAN. ■

## Snort and ARP

Snort [6] is a prominent example of a network IDS. The Intrusion Detection System helps administrators detect attacks on the network at an early stage so they can launch countermeasures. Snort has an Arpspoof preprocessor with four detection mechanisms:

- For each ARP request it detects, the Arpspoof preprocessor validates the source address in the Ethernet frame against the source address in the ARP packet. If these two addresses do not match, it issues a warning. ARP poisoning does not imply using different addresses in these fields, which means that the attack would go unnoticed.
- For ARP replies, a comparison of the source and target addresses is per-

formed. If one of these address pairs does not match, Snort issues a warning. Again, this will not detect ARP poisoning, although it does catch Proxy ARP – on the other hand, this technique is typically legitimate and involves one machine answering ARP requests on behalf of another machine.

- The system alerts on ARP requests that are sent to unicast addresses rather than broadcast. Although this behavior does not comply with the (now 20 year old) standard, there are good reasons for it. However, a genuine ARP attack does not need to unicast ARP requests, so again, this mechanism would fail to detect an ARP poisoning attack.

- Snort checks all ARP packets based on a list of IP addresses and MAC addresses supplied by the administrator. If the source IP address is on the list, the IDS will read the corresponding MAC address from the list and compare it with the source MAC address from the ARP packet and Ethernet frame. In case of discrepancy, Snort issues a warning. This mechanism is only useful for small networks, as the configuration effort is too high in any other case. There is no way to use this functionality sensibly when faced with dynamic IP address assignments (DHCP).

In other words, Snort's ability to detect ARP poisoning is limited, as is the case for all Intrusion Detection Systems.