**HTTP Redirection using ARP Spoofing**

**-**by Ravish Thalesar, Rushabh Shah and Rohan Suri.

Introduction

What is ARP?

**Address Resolution Protocol** (**ARP**) is a [telecommunication](http://en.wikipedia.org/wiki/Telecommunication) protocol used for resolution of [network layer](http://en.wikipedia.org/wiki/Network_layer) addresses into [link layer](http://en.wikipedia.org/wiki/Link_layer) addresses.

ARP is used to convert a network address (e.g. an [IPv4 address](http://en.wikipedia.org/wiki/IPv4_address)) to a physical address such as an [Ethernet address](http://en.wikipedia.org/wiki/Ethernet_address)(also known as a [MAC address](http://en.wikipedia.org/wiki/MAC_address)). A [host](http://www.webopedia.com/TERM/H/host.html) wishing to obtain a physical address [broadcasts](http://www.webopedia.com/TERM/B/broadcast.html) an ARP request onto the TCP/IP network. The host on the network that has the IP address in the request then replies with its physical hardware address.

What is ARP cache?

In simplest terms, an ARP cache is a stored mapping of IP addresses with link layer addresses. An ARP cache obviates the need for an ARP request/reply conversation for each IP packet exchanged. Naturally, this efficiency comes with a price. Each host maintains its own ARP cache, which can become outdated when a host is replaced, or an IP address moves from one host to another. The ARP cache is also known as the neighbor table.

What is ARP Spoofing?

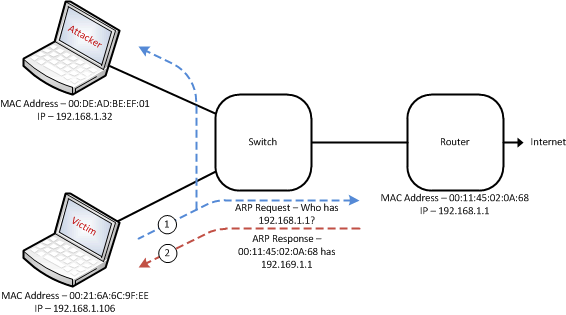
**ARP spoofing**, **ARP cache poisoning**, or **ARP poison routing**, is a technique by which an attacker sends ([spoofed](http://en.wikipedia.org/wiki/Spoofing_attack)) [Address Resolution Protocol](http://en.wikipedia.org/wiki/Address_Resolution_Protocol) (ARP) messages onto a [local area network](http://en.wikipedia.org/wiki/Local_area_network). Generally, the aim is to associate the attacker's [MAC address](http://en.wikipedia.org/wiki/MAC_address) with the [IP address](http://en.wikipedia.org/wiki/IP_address) of another [host](http://en.wikipedia.org/wiki/Host_(network)), such as the[default gateway](http://en.wikipedia.org/wiki/Default_gateway), causing any traffic meant for that IP address to be sent to the attacker instead. ARP spoofing may allow an attacker to intercept [data frames](http://en.wikipedia.org/wiki/Data_frame) on a network, modify the traffic, or stop all traffic. Often the attack is used as an opening for other attacks, such as [denial of service](http://en.wikipedia.org/wiki/Denial_of_service), [man in the middle](http://en.wikipedia.org/wiki/Man-in-the-middle_attack), or[session hijacking](http://en.wikipedia.org/wiki/Session_hijacking) attacks.

Address Resolution Protocol’s vulnerabilities

The reason for such attacks being possible are because of the stateless nature of this protocol. a **stateless protocol** is a [communications protocol](http://en.wikipedia.org/wiki/Communications_protocol) that treats each request as an independent transaction that is unrelated to any previous request so that the communication consists of independent pairs of [*request and response*](http://en.wikipedia.org/wiki/Request%E2%80%93response). Network hosts will automatically [cache](http://en.wikipedia.org/wiki/Cache_(computing)) any ARP replies they receive, regardless of whether Network hosts requested them. Even ARP entries which have not yet expired will be overwritten when a new ARP reply packet is received. There is no method in the ARP protocol by which a host can[authenticate](http://en.wikipedia.org/wiki/Authenticate) the peer from which the packet originated. This behavior is the vulnerability which allows ARP spoofing to occur.

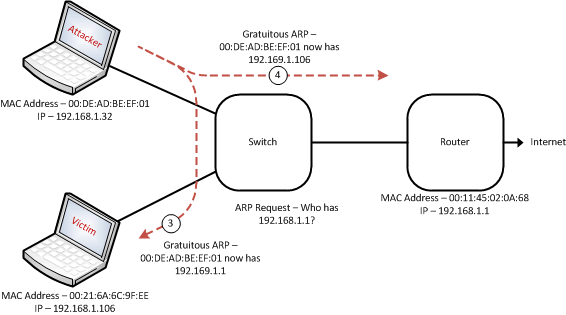
ARP spoofing explaining with an example

In the example below, we see the host 192.168.1.106 coming online and issuing an ARP Request for the MAC Address of the default gateway 192.168.1.1.  This ARP Request is broadcasted to all hosts on the network.  In this example, the network has not been compromised yet, so the proper ARP Response is issued by the router and sent towards the requesting host.  Through this mechanism, the requesting host now associates the Layer 2 MAC Address 00:11:45:02:0A:68 with the Layer 3 IP Address 192.168.1.1.



Now lets see the scenario in case of the attack.

In the diagram below, you can see that the attacker has sent an ARP Reply to both the router as well as to the victim.  In the case of the ARP Reply to the router, we are telling the router that traffic destined towards the victim should now be sent to the attacker.  Similarly, in the case of the ARP Reply to the victim, we are telling the victim that traffic destined towards the router should likewise be sent to the attacker as well.



Http redirection

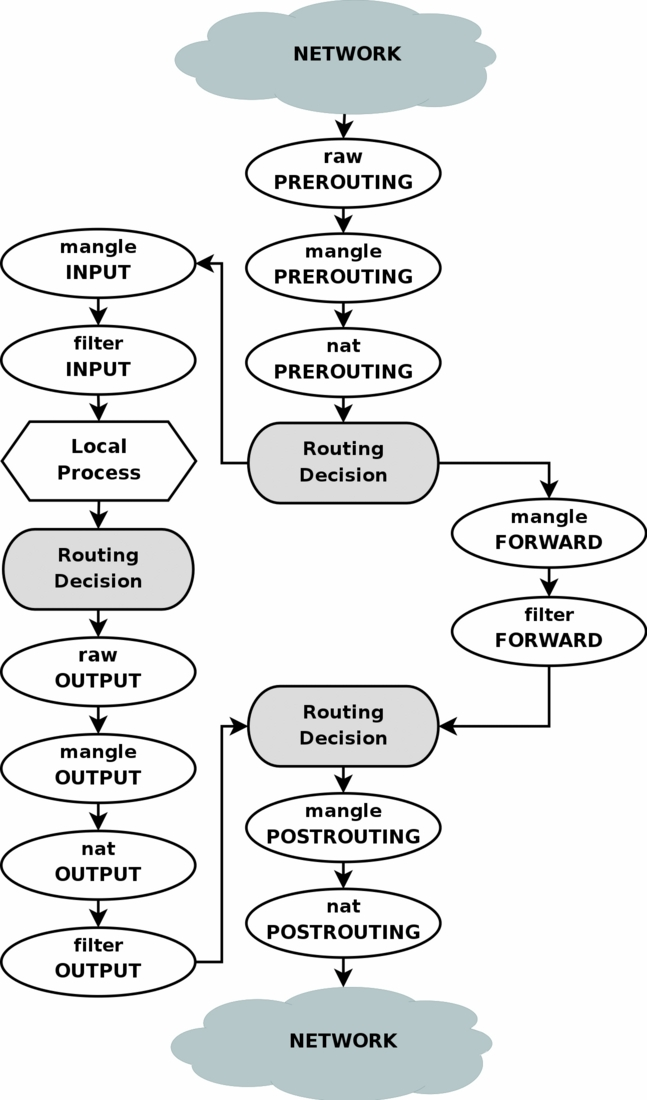
This is what we want to achieve using ARP spoofing. The attacker gets all packets sent by the victim once we have spoofed. Next, we simply forward all other requests to the router except the ones which are http requests. We modify such http requests to a preset http request and send it to the router! Hence the returned response is of some other webpage than desired by the victim!

Project requirements

* Python
* Scapy
* Tkinter
* Linux distro

Implementation

* We enable ip\_forwarding in the attacker’s system. So that it doesn’t drop any packets that it didn’t request for.
* Next we define rules for the Linux Netfiler system using the command *iptables .*
* iptables allows the [system administrator](http://en.wikipedia.org/wiki/System_administrator) to define *tables* containing *chains* of *rules* for the treatment of packets. Each table is associated with a [different kind of packet processing](http://en.wikipedia.org/wiki/Netfilter#iptables). Packets are processed by sequentially traversing the rules in chains. A rule in a chain can cause a goto or jump to another chain, and this can be repeated to whatever level of nesting is desired. (A jump is like a “call”, i.e. the point that was jumped from is remembered.) Every network packet arriving at or leaving from the computer traverses at least one chain.
* Let us see how a packet gets processed



* The rules that we defined for our project are
  + iptables –t nat –A PREROUTING –i eth1 –p tcp –dport 80 –j DNAT --to-destination 192.168.1.3:8080

Append a prerouting rule to the nat table such that for any packets on eth1 interface that use TCP/IP protocol and port 80 must jump to DNAT target specified by the new destination address and port.

Port 443 may also be specified to redirect any https requests (–dport 80,443)

* + iptables -A FORWARD --in-interface eth1 -j ACCEPT

This chain is entered after the prerouting decision has been taken. Its target is to accept the packet and now forward it to the gateway. This is defined because we want certain packets (other than port 80 ones) to bypass NAT.

* + iptables -t nat --append POSTROUTING --out-interface eth1 -j MASQUERADE

The source address of outbound packets is replaced with the static IP address of the attacker. When outside computers respond, they will set the destination address to the IP address of the attacker, and the attacker will intercept those packets, change their destination addresses to the victim’s IP address and forward them.

References

<http://webstersprodigy.net/2012/07/06/some-practical-arp-poison-attacks-with-scapy-iptables-and-burp/>

<http://www.control.aau.dk/~jdn/edu/courses/14-2/socket-iptables/littt/Oreilly.Linux.iptables.Pocket.Reference.Aug.2004.pdf>