Assignment 4

Man-in-the-middle Attacks - ARP Spoofing/TCP Hijacking

Introduction

In this assignment you will examine some of the security flaws in existing network protocols. Our currently used suite of network protocols, commonly implemented in the TCP/IP protocol stack on contemporary computer systems, was designed in an environment of trust. Communicating network machines were trusted not to lie about who they are or subvert the communications on the network. In our current global internet environment this trust does not hold because the attackers are in control of some of the machines on the network. In this assignment we will see how the attacker can manipulate packets and lie about who he is. Manipulation of the network can occur at any of the network layers of the OSI model. In this assignment we will manipulate information in the link layer, internet layer, and the transport layer. These manipulations will allow us to eavesdrop, redirect communications and hijack TCP sessions. A common hacking tool will be used to perform the packet manipulations and you will capture and analyze the resulting traffic in order to better understand how the network vulnerabilities are being exploited.

Part 1 - ARP Spoofing and TCP Hijacking

The Address Resolution Protocol (ARP)

In addition to specifically asking for a MAC/IP mapping when a machine needs to send a network packet, it also listens to all the other ARP traffic on the network. By "listening-out" to ARP traffic in this way a machine can come to know even more MAC/IP address mappings and store them in its ARP cache. In this way the machine may not have to delay a network communication while it makes an ARP request; it already discovered the mapping by listening-out. This scheme is called "Gratuitous ARP". These updates are done even for IP addresses that are already in the ARP cache.

You can see that in order to populate its ARP cache a machine trusts the senders of the ARP messages not to lie. An attacker can craft false ARP packets and cause a machine to alter the mappings in its ARP cache. In this way the attacker can convince the target machine to send packets anywhere he wishes to. This is called "ARP spoofing" or "ARP Cache Poisoning."

TCP Hijacking

If an attacker can monitor an ongoing TCP connection he has access to all the information in the IP and TCP headers and to the packet payloads. Of course the attacker can eavesdrop on the connection, but by gathering the right information the attacker can actually take over the connection (hijack it). To do this the attacker needs to know the IP addresses and port numbers the connection is using. He also needs to know the sequence numbers in the TCP headers that are being used to control the error correction mechanisms in the TCP stream communication. Recall that there is an incrementing series of sequence numbers (a different set of numbers for each direction of communication) that are used to make sure: that all packets are being received at the destination, that the packets are reassembled in the correct order, and that duplicated packets are only used once. During synchronization (the initial TCP 3-way handshake) a random sequence number is chosen as the starting sequence number for each direction of the communication. As data is passed on the TCP connection the sequence numbers reported in each packet are incremented to indicate the number of bytes of data passed on the connection up to the point of that packet's transmission. The sequence numbers are used to sort out the order of packets (and discover missing packets) at the other end.

Since the attacker is eavesdropping on the network he has all the information he needs to craft a false packet that looks like it is the next packet in the TCP connection. For example if a user is running a telnet session and logged into a remote computer the attacker can craft packets to send his own commands to the remote machine. The packets will look like, and be executed as if they came from the legitimate user. This is a "TCP Simple Hijack."

A TCP simple hijack has some problems. It works fine for the first hijacking packet but not for more. When the attacker sends his packet he increments the sequence numbers in the normal way so the packet is accepted properly at the destination. Unfortunately the original user's machine is not aware of the hijacking data that was falsely sent and its sequence numbers are still at the old values. When the destination machine acknowledges the false packet the original user's machine sees an acknowledgment for data that it never sent (the data was sent by the attacker). The sequence numbers are now out of synchronization between the legitimate user's machine and the remote host (destination machine). Different operating systems deal with this in different ways but there is danger of a continuous steam of ACK packets being passed back and forth between the machines as they hopelessly try to resynchronize. This is called an "ACK Storm" and can hang the systems.

Now, if we combine this TCP hijacking technique with the ARP spoofing techniques in the previous section we can solve the ACK Storm problems and inject characters into the TCP stream, or take total control of the TCP connection. The attacker uses ARP spoofing to have the legitimate user's machine and the remote host both send their packets to his attacking machine, instead of their correct destinations. The attacker is now a man-in-the-middle. He controls all the packet flow between machines. Initially, before the hijack, the attacker can simply forward the messages to their correct destination.

However, when he wants to inject packets the attacker can send crafted packets to one of the target machines. The crafted packets will spoof the correct sequence numbers so they will be

accepted by the target machine. After this point the sequence numbers in the original two target machines will be out of sync when they communicate; one machine has received more data than the other machine has sent. Therefore the attacking machine as man-in-the-middle must modify the sequence numbers on-the-fly as they pass through to adjust them and keep the target machines in synchronization.

To completely take over the connection, the attacker stops forwarding packets for the legitimate user's machine associated with the hijacked TCP connection. The attacker now fully takes the place of the legitimate user's machine and the remote target host is not aware of the switch. To the legitimate user's machine the connection has "gone-dead". The legitimate user's connections will eventually time-out, or the attacker may try a more sophisticated resynchronization technique when he is done with the hijack.

What is expected?

(i) You must detect an ARP man-in-middle attack
(ii) There can be multiple such attacks you have to detect all of them.
(iii) Try to hijack a TCP connection, you can use ARP poison in this.
(iv) Implement on active network, using packet capture (libpcap), not on traffic dumps.

Part 2 – Defend against ARP Spoofing and TCP hijacking

To defend against ARP spoofing and other man-in-middle attacks, what methods do you take? Can you design any solution? Explain the feasibity and technical details

What is expected?

(i) Write in detail about the methods
(ii) What is the feasibility and how safe are the methods that you suggest?

References:


```
http://www.tildefrugal.net/tech/arp.php
http://www.omnisecu.com/tcpip/address-resolution-protocol-arp.htm
http://www.irongeek.com/i.php?page=security/arpspoof
http://systhread.net/texts/200805lpcap1.php
http://www.tcpdump.org/pcap.html
http://yuba.stanford.edu/~casado/pcap/section1.html
```


If you have doubts, please email to ramki@cse.iitd.ernet.in with subject: Assignement4