# “TCP Hijacking”

Invenia mvia mautfacia m

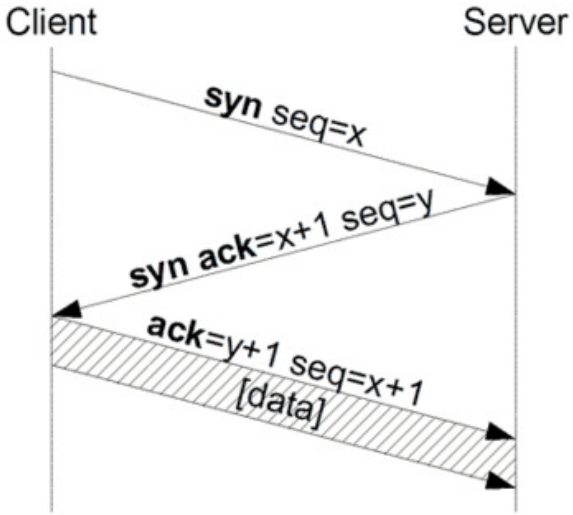
# “What is TCP”

- TCP = Transmission Control Protocol
- Part of the IP network protocol suite
- It is a connection-based protocol
- It is a point-to-point protocol
- It is used for data transfer across network between two points
- Defined in the RFC 793

Inveniamviamautfaciam

# “TCP calling …”

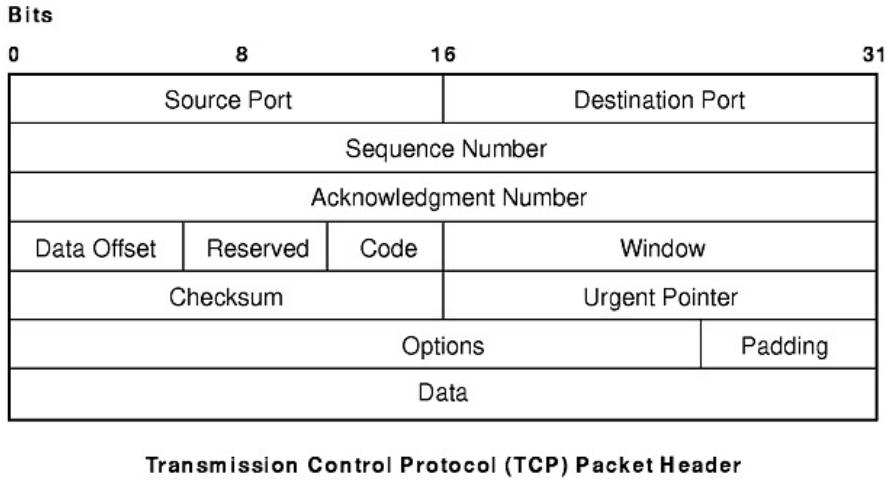
- TCP requires ARP (Address Resolution Protocol)
- ARP maps MAC addresses to IP addresses
- ARP is a broadcast protocol using “request” and “reply” packets
- To cut down on traffic each machine keeps an ARP cache
- Once a machine has another’s IP address TCP can start
- A 3-way “handshake” is started






# “TCP calling … (cont.)”

- As part of the “handshake” a sequence number is given
- A sequence number (SN) is used to help order the packets
- For each transmitted byte the sequence number increments by one
- A window size is also given which states a SN range
- Window scaling is used for high latency/bandwidth connections
- Thus any TCP connection is made unique through 5 parameters. An attacker needs to know all 5;
  - Source IP address
  - Destination IP address
  - Source Port
  - Destination Port
  - Sequence Number




Transmission Control Protocol (TCP) Packet Header



# “The Problem with Sequence Numbers”

- A sequence number should be 1 of $2^{32}$ possible values

- First problem is that the SN is incremented with each transmission

- While the window range means that anything out of range is discarded, it means anything in range is accepted

- Different OS’s and applications have different window sizes (ie; Windows XP has a window size of 65535, which equates to a possible 65537 possible sequence numbers).

- Windows scaling (RFC 1323) makes this worse because for certain applications the window size can be scaled up by a factor of 16384

- All of this makes it easier to guess or brute-force the sequence number

Inveniamviamautfaciam

# “Any port in a storm?”

• Now lets move onto ports. Getting a destination port is easy

• Guessing a source port –without sniffing- should be difficult as it should be randomly assigned ( 1 of 65535 options), but…

• Port 1024 and below are reserved

• Different OS’s increment the source port differently. BSD is truly random while windows XP starts at 1050 and then increments consecutively

• If the attacker can sniff the TCP connection they can get it that way

• This all allows the attacker to guess the 5$^{th}$ parameter



# “Blind TCP Hijacking”

• Now that an attacker has all 5 of the parameters, so what?

• Remember that a packet within the window would be accepted..

• Sending a “valid” packet with a RST flag would drop the connection. This would cause the connection to drop straightaway.

• Sending a “valid” packet with a SYN flag would reset the connection. This would drop the connection after the host responds to the packet.

• Both of the above can be very nasty as DOS attacks.

• Sending a packet with a valid payload would get the host to execute a command. For example inserting a command into a ftp connection.

• All of these are “blind” because they are non-interactive exploits but they are very effective and easy (one could use *nemesis, tcpdump* and *gawk*), and can be difficult to trace.

Invenia m via ama utfaciam

# “Network Session Hijacking”

• Blind Hijacking works because the attacker does not worry about a response

• But if we do, then we can indulge in session hijacking

• In this respect we either take over a session or we inject characters into it

• In order to do this the attacker would have to insert itself into the process

• This is called a man-in-the-middle attack (MITM) and can be accomplished by many tools (*arpspoof, hunt, webmitm, ettercap*, etc) using arp-poisoning or dns-poisoning

• This causes the client to view the attacker as the server, and the server to view the attacker as the client. It could also cause ACK storms due to ACK negotiations

• Once this happens any data travelling between the client and server is compromised

• This completely bypasses authentication

Invenia man via man au t face ia m

# “Does it work? -1”

- Lets watch, first we check the normal ARP tables..

**HOST A:**

*# arp -n*

| *Address* | *HWtype* | *HWaddress* | *Flags Mask* | *Iface* |
|---|---|---|---|---|
| *19x.x.x.11* | *ether* | *00:08:9B:1C:67:78* | *C* | *eth0* |

**HOST B:**

*# arp -n*

| *Address* | *HWtype* | *HWaddress* | *Flags Mask* | *Iface* |
|---|---|---|---|---|
| *19y.y.y.12* | *ether* | *00:04:23:1E:BC:28* | *C* | *eth1* |



# “Does it work? -2”

- Now we use *hunt* to poison the ARP caches of our two machines..

**ATTACKER MACHINE:**

*--- arpspoof daemon --- rcvpkt 107, free/alloc 63/64 ------*

s/k) start/stop relayer daemon

l/L) list arp spoof database

a) add host to host arp spoof i/I) insert single/range arp spoof

d) delete host to host arp spoof r/R) remove single/range arp spoof

t/T) test if arp spoof successed y) relay database

x) return

*-arps> a*

*src/dst host1 to arp spoof> 19y.y.y.12*

*host1 fake mac [EA:1A:DE:AD:BE:01]>*

*src/dst host2 to arp spoof> 19x.x.x.11*

*host2 fake mac [EA:1A:DE:AD:BE:02]>*

*refresh interval sec [0]> 1*

*--- arpspoof daemon --- rcvpkt 1774, free/alloc 63/64 ------*

Inveniamviamautfaciam

# “Does it work? -3”

- Now we recheck the ARP tables..

**HOST A:**

*#arp -n*

| *Address* | *HWtype* | *HWaddress* | *Flags Mask* | *Iface* |
|---|---|---|---|---|
| *19x.x.x.11* | *ether* | *EA:1A:DE:AD:BE:02* | *C* | *eth0* |

**HOST B:**

*# arp -n*

| *Address* | *HWtype* | *HWaddress* | *Flags Mask* | *Iface* |
|---|---|---|---|---|
| *19y.y.y.12* | *ether* | *EA:1A:DE:AD:BE:01* | *C* | *eth1* |

- You can see the result of a simple operation.



# “What does it mean?”

• Currently there is no foolproof way to protect against these threats on a local area network.

• An attacker can still use arpspoofing across networks by poisoning the router caches

• Blind TCP Hijacking is very difficult to trace and can be devastating as a DOS attack

• As long as we use TCP in it’s current form, this will be a threat.

• Coutermeasures include..
  - Snort with the arp spoofing modules
  - Static ARP tables
  - Use strong encryption on protocols
  - Use patches to ensure smaller windows and random source ports
  - Use proper ingress and egress filtering

• New variants also include Host Session Hijacking and Wireless Hijacking

Invenia m via m au t fac ia m

# Thank you for your attention


http://www.unet.univie.ac.at

http://upload.wikimedia.org

http://www.ibiblio.org/pub/docs/rfc/rfc793.txt

http://www.ietf.org/rfc/rfc1323.txt

http://nemesis.sourceforge.net/

http://www.tcpdump.org/

http://linux.maruhn.com/sec/hunt.html

http://monkey.org/~dugsong/dsniff/

http://ettercap.sourceforge.net/


Inveniamviamautfaciam