ARP Games - Playing Man In The Middle Or Knocking Off With DoS

Muhammad Farooq-i-Azam
CHASE-2006
Lahore

Overview

- Theory
- Existing Sniffers – in action
- Switched Environment
- ARP Protocol and Exploitation
- Develop it yourself

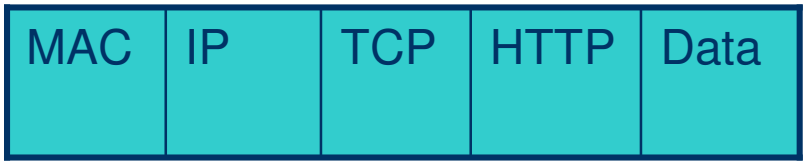
Network Traffic

- Computers and network devices communicate with each other by sending or receiving information over tiny bundles of electronic signals called packets.

- Flow of different packets to and from different computers over the network is said to constitute network traffic.

Packet Types and Structures

- Data Packets – Visible
  - Not all contents of the packet are seen by the end user.
  - e. g. HTTP packets



Packet Types and Structures

- Control Packets – Invisible
  - e. g. ARP packets

Ethernet – Our Domain of Experiment

- Non-Switched Hub
  - Single broadcast domain
  - Seldom used now a days
- Switched Hub
  - Multiple broadcast domains

Promiscuous Mode

- Normal Operation of NIC
  - Receive packets of own address only
- Operation Under Promiscuous Mode
  - Receive all packets regardless of destination address
  - Non-switched Ethernet LAN
    - Capture traffic of neighbors
  - Switched Ethernet LAN
    - Still gets own traffic
    - Further techniques in addition to mere promiscuous mode

Packet Sniffer

- A piece of software that captures all the traffic flowing in and out of a computer.
- Non-Promiscuous Mode Sniffing
  - Own packets for both switched and non-switched LANs.
- Promiscuous Mode Sniffing
  - Non-switched LAN
    - Entire neighbor traffic
  - Switched LAN
    - Own traffic only
    - Additional technique for capturing neighbor traffic
    - ARP exploits

Some Theory

Network Layers

Ethernet and ARP Protocols

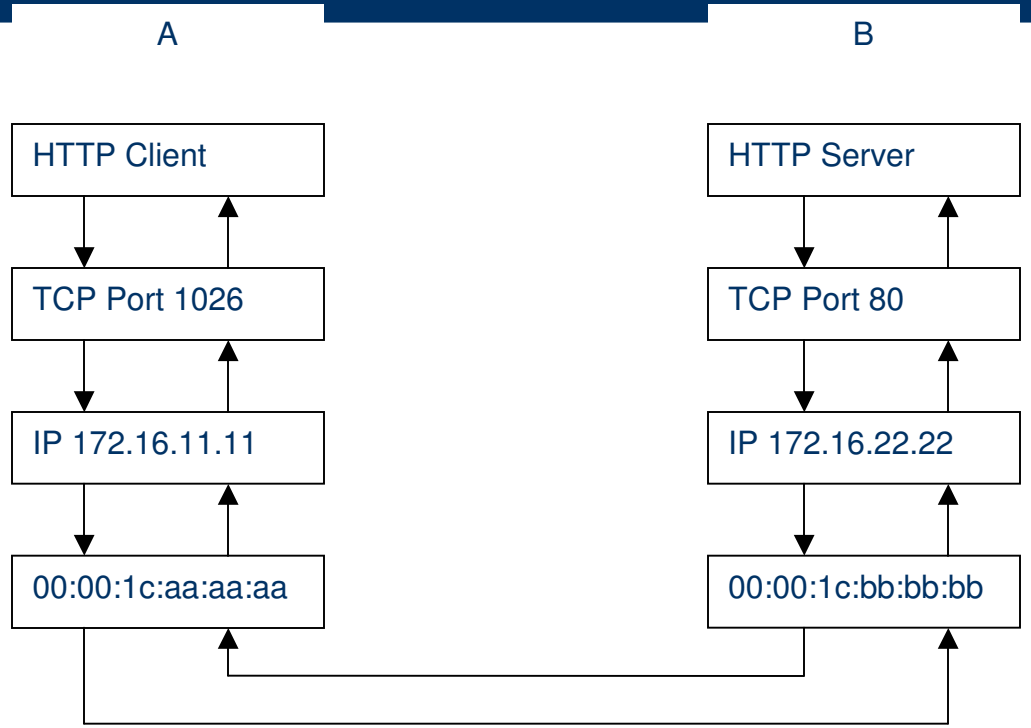
Physical and Logical

- Node to node communication uses physical addresses i.e. MAC addresses.
- Applications use logical addresses i.e. IP addresses.
- Each host on an Ethernet LAN has
  - IP address
  - MAC address

Why Logical Needs to Get Physical

- Host A → Host B
- Application on A→Application on Host B
- They talk to each other using services provided by protocols on lower layers

Layer Talks to Layer



Frame Transmission

- MAC header of the frame should have destination's physical address
- Sending machine should know the receiving machine's physical address

Destination MAC	Source MAC	Proto Type

Finding Destination

- Knowing the Destination Physical Address
  - Host maintains a list of IP – MAC mappings
  - Uses ARP for hosts not in the list
  - Updates the list of mappings

Typical IP – MAC Mappings

- arp -a
- Interface: 172.16.30.30 on Interface 0x9000003

Internet Address	Physical Address	Type
172.25.30.1	00-50-8b-66-86-33	dynamic
172.25.30.2	00-07-e7-47-b3-cb	dynamic
172.25.30.3	00-50-fc-71-38-94	dynamic
172.25.30.4	00-07-e9-5b-3f-e6	dynamic
172.25.30.5	00-30-6e-c9-af-13	dynamic

ARP Overview

- Host A wants to send a frame to host B
- Host A broadcasts an ARP query to find out physical address for the given IP of B
- Every host on LAN receives query
- Host B with the given IP sends back its physical address in a unicast reply.
- Host A now sends the frame to B using the destination address

ARP Frame Format

48.bit: Ethernet address of destination
48.bit: Ethernet address of sender
16.bit: Protocol type

Ethernet packet data:

16.bit: Hardware address space (e.g., Ethernet)
16.bit: Protocol address space.
8.bit: byte length of each hardware address
8.bit: byte length of each protocol address
16.bit: opcode [REQUEST | REPLY]
nbytes: Hardware address of sender
mbytes: Protocol address of sender
nbytes: Hardware address of target of this packet (if known).
mbytes: Protocol address of target.

Example Packet – ARP Request

Destination:	**ff:ff:ff:ff:ff:ff**
Source:	**00:11:11:25:4a:d2**
Type:	**0x0806 (ARP)**
Hardware type:	**0x0001 (Ethernet)**
Protocol type:	**0x0800 (IP)**
Hardware size:	**6**
Protocol size:	**4**
Opcode:	**0x0001 (Request)**
Sender MAC address:	**00:11:11:25:4a:d2**
Sender IP address:	**172.16.30.27**
Target MAC address:	**00:00:00:00:00:00**
Target IP address:	**172.16.36.21**

Example Packet – ARP Reply

Destination:	**00:11:11:25:4a:d2**
Source:	**00:03:ba:68:7b:b2**
Type:	**0x0806 (ARP)**
Hardware type:	**0x0001 (Ethernet)**
Protocol type:	**0x0800 (IP)**
Hardware size:	**6**
Protocol size:	**4**
Opcode:	**0x0002 (Reply)**
Sender MAC address:	**00:03:ba:68:7b:b2**
Sender IP address:	**172.16.36.21**
Target MAC address:	**00:11:11:25:4a:d2**
Target IP address:	**172.16.30.27**

ARP Exploitation

- Fabricated/Spoofed ARP packets
  - ARP request packets
  - False source IP – MAC binding
  - Poison target ARP cache
  - Unicast source-spoofed packets to a single host
    - Interrupt communication between spoofed and target hosts
    - Intercept communication between spoofed and target hosts
  - Broadcast source-spoofed packets
    - Isolate target host from everyone else
    - Intercept communication between target and the rest

ARP Exploitation - Example

- Three hosts on a LAN
  - Principal host A
    - IP 172.16.11.11
    - MAC 00:00:c1:aa:aa:aa
  - Principal host B
    - IP 172.16.22.22
    - MAC 00:00:c1:bb:bb:bb
  - Man in the middle host C
    - IP 172.16.33.33
    - MAC 00:00:c1:cc:cc:cc

ARP Exploitation - Example

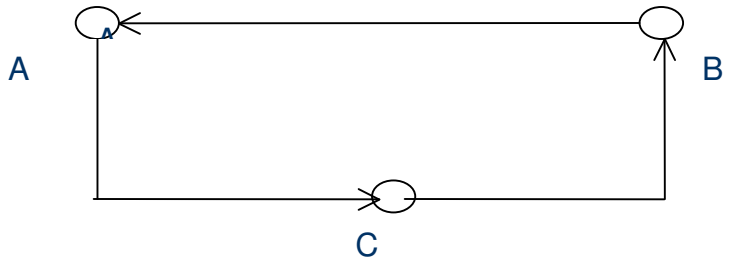
- Normal IP – MAC Bindings
  - Host A
    - 172.16.22.22 00:00:c1:bb:bb:bb
  - Host B
    - 172.16.11.11 00:00:c1:aa:aa:aa

ARP Exploitation - Example

- Interception By Host C
  - One-way interception
    - All communication from A to B OR
    - All communication from B to A
  - Two-way interception
    - All communication between A and B

ARP Exploitation - Example

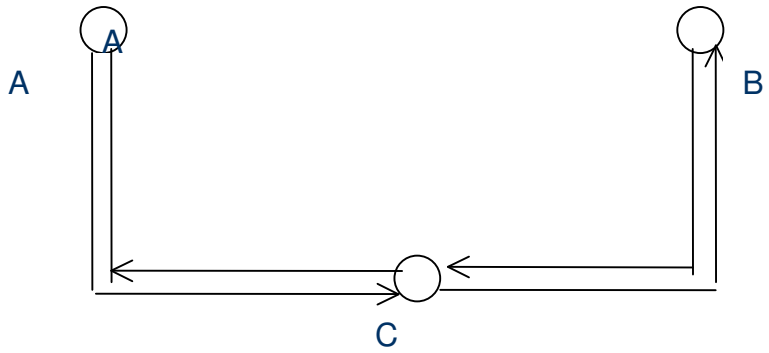
- One-way interception
  - Make A believe that B has MAC of C
    - Change IP – MAC mappings in host A to be
    - 172.16.22.22 00:00:1c:cc:cc:cc
    - C forwards packets with destination IP of B to B
      - Else communication from A do not reach B i.e. partial DoS



ARP Exploitation – Example

- Two-way interception
  - Make A believe that B has MAC of C
    - Change IP – MAC mappings in host A to be
    - 172.16.22.22 00:00:1c:cc:cc:cc
    - C forwards packets received from A to B
  - Make B believe that A has MAC of C
    - Change IP – MAC mappings in host B to be
    - 172.16.11.11 00:00:1c:cc:cc:cc
    - C forwards packets received from B to A
  - If no packet forwarding, then interruption i.e. DoS

ARP Exploitation - Example



How to Impersonate

- Spoofed ARP request packets
  - Desired IP – MAC binding in the sender field
  - Example: C impersonating B
    - Send an ARP request packet to A with following data
      - Sender MAC Address: 00:00:1c:cc:cc:cc
      - Sender IP Address: 172.16.22.22
    - IP – MAC binding in A is now:
      - 172.16.22.22 00:00:1c:cc:cc:cc

ARP Exploitation - Example

- **ARP Source-Spoofed Request Packet**

Destination:	00:00:c1:aa:aa:aa
Source:	00:00:c1:cc:cc:cc
Type:	0x0806 (ARP)
Hardware type:	0x0001 (Ethernet)
Protocol type:	0x0800 (IP)
Hardware size:	6
Protocol size:	4
Opcode:	0x0001 (Request)
Sender MAC address:	00:00:c1:cc:cc:cc
Sender IP address:	172.16.22.22
Target MAC address:	00:00:00:00:00:00
Target IP address:	172.16.66.66

DoS Or Intercept

- DoS
  - Just Capture and do not forward
- Intercept
  - Capture and forward
    - # echo 1 > /proc/sys/net/ipv4/ip_forward

Packet Sniffer Tools

- ipgrab
  - Command line
  - Distributed with Debian Linux
  - For Unix based systems
- tcpdump
  - Command line
  - Distributed with various platforms
  - Classical
- ethereal
  - Both command line and gui
  - Both windows and Unix based systems

Existing Tools

- Switched Network
  - dsniff
  - ettercap
  - arpspoof

Example Trace – An Exchange of Packets

172.16.22.22 -> Broadcast ARP Who has 172.16.11.11? Tell 172.16.22.22

172.16.11.11 -> 172.16.22.22 ARP 172.16.11.11 is at 00:00:1c:aa:aa:aa

172.16.22.22 -> 172.16.11.11 TCP 1291 > 8080 [SYN]

172.16.11.11 -> 172.16.22.22 TCP 8080 > 1291 [SYN, ACK]

172.16.22.22 -> 172.16.11.11 TCP 1291 > 8080 [ACK]

172.16.22.22 -> 172.16.11.11 HTTP GET http://www.msn.com/ HTTP/1.0

Example Trace – Single Packet

```
Ethernet II, Src: 00:00:1c:bb:bb:bb, Dst: 00:00:1c:aa:aa:aa
   Destination:   00:00:1c:aa:aa:aa
   Source:                        00:00:1c:bb:bb:bb
   Type:                          IP (0x0800)
Internet Protocol, Src Addr: 172.16.22.22 (172.16.22.22), Dst Addr: 172.16.11.11 (172.16.11.11)
   Version:                       4
   Header length:                 20 bytes
   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
                                  0000 00.. = Differentiated Services Codepoint: Default (0x00)
                  .... ..0. = ECN-Capable Transport (ECT): 0
                  .... ...0 = ECN-CE: 0
   Total Length: 413
   Identification:                0x22bd (8893)
   Flags:                         0x04 (Don't Fragment)
                  0... = Reserved bit: Not set
                  .1.. = Don't fragment: Set
                  ..0. = More fragments: Not set
   Fragment offset:               0
   Time to live:  128
   Protocol:                      TCP (0x06)
   Header checksum:               0x505f (correct)
   Source:                        172.16.22.22 (172.16.22.22)
   Destination:   172.16.11.11 (172.16.11.11)
```

Example Trace – Single Packet (continued)

Transmission Control Protocol, Src Port: 1291 (1291), Dst Port: 8080 (8080), Seq: 1, Ack: 1, Len: 373

Source port:	1291 (1291)
Destination port:	8080 (8080)
Sequence number:	1 (relative sequence number)
Next sequence number: 374	(relative sequence number)
Acknowledgement number:	1 (relative ack number)
Header length:	20 bytes
Flags:	0x0018 (PSH, ACK)
	0... .... = Congestion Window Reduced (CWR): Not set
	.0.. .... = ECN-Echo: Not set
	..0. .... = Urgent: Not set
	...1 .... = Acknowledgment: Set
	.... 1... = Push: Set
	.... .0.. = Reset: Not set
	.... ..0. = Syn: Not set
	....0 = Fin: Not set
Window size:	17520
Checksum:	0x40a4 (correct)

Example Trace – Single Packet (continued)

Hypertext Transfer Protocol
GET http://www.msn.com/ HTTP/1.0\r\n
Request Method: GET
Request URI: http://www.msn.com/
Request Version: HTTP/1.0
Accept: */*\r\n
Accept-Language: en-us\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;.NET CLR 1.1.4322)\r\n
Host: www.msn.com\r\n
Proxy-Connection: Keep-Alive\r\n
\r\n

Packet Sniffer Development

- Kernel dependent packet capture
  - OS dependent code
  - No portability – new application for each OS
  - E.g.
    - Berkeley Packet Filter (BPF) BSD
    - Data Link Provider Interface (DLPI) Solaris
    - SOCK_PACKET Linux
- Kernel-independent packet capture
  - Libpcap Unix based systems
  - Winpcap Microsoft Windows
  - Portable

Source Code

- ARP Packet Crafting
- Packet Sniffer

Thank You

- Questions?