

Network Attacks

CSC 348-648



Spring 2013

Computer Networks and Security

- Ensuring that two systems can effectively and efficiently communicate
 - Internet (TCP/IP protocols) is the most successful example
- Want to provide integrity, confidentiality, and availability

Is this possible with the current Internet?



- In “Security problems in the TCP/IP protocol suite” Bellovin describes why IP and security are actually at odds...

Network Attacks

- Attacks can occur at every layer of the network model

| | Model Layer | Example Attacks |
|-----|-------------|--------------------------------------|
| 7 | Application | DNS poisson, XSS, ... |
| 4 | Transport | session hijack, port scan, SYN flood |
| 3 | Network | route changes, reflection, ICMP |
| 1-2 | MAC | promiscuous NIC, ARP poisson, DoS |

- Consider sniffing traffic
 - At layers 1-2 use promiscuous NIC, at layer 3 change routing tables, at layer 4 hijack TCP session, or at layer 7 use DNS poisson
- Consider DoS
 - Layers 1-2 use ARP, at layer 3 use reflection, or at layer 4 use SYN flooding

Layer 2 Attacks

- Medium Access Control (MAC) layer concerns local communications
 - Assumes computers are *local*, can be contacted directly
 - No *message passing* or routing
- Example technologies includes Ethernet, 802.11, etc...

Layer 2 Sniffing

- In a broadcast network, frames are sent to all *local* machines
 - The NIC will hear all the frames, but will normally only respond to frames with its MAC as the destination address (*only passes frames with its MAC address to layer 3*)
 - Setting the NIC to *promiscuous*, all frames transmitted will be obtained (*all are sent to layer 3*)
 - Often referred to as *passive sniffing*
- *What kind of information can be obtained?*
 - All the information, just need to interpret
 - telnet, ftp, and POP3 transmit passwords in plaintext

How can we prevent this attack?

Defeating *Older* Ethernet Switches

- Switched Ethernet decreases, if not, eliminates sniffing
 - Typically each machine connects directly to the *switch*
- Smarter than a hub, switch reads frames (destination MAC address) and forwards it to the correct machine **only**
 - Therefore, the switch must keep a table

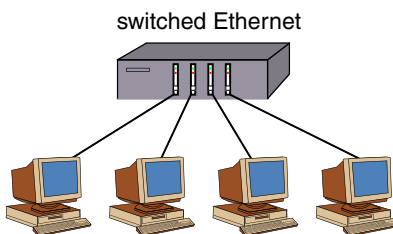


Table contents would be?

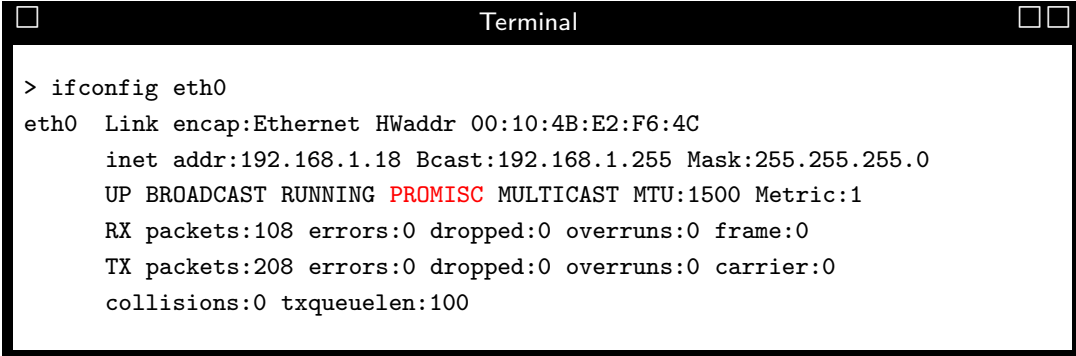
- How the switch creates the MAC table
 - When a machine connects, it will send an ARP messages
 - ARP message states “my MAC address is x ”
 - Switch stores the MAC and the line (port) it is associated with
- The problem with *most* switches
 - Tables have a finite size
 - An attacker can send a *flood* of fake ARP messages, each with a unique random MAC address (use program `macof`)
 - Switch add entries until the table is full
 - Once the table is full, older switches will begin forwarding frames to **all** computers
 - Now attacker can sniff traffic, the switch is just a hub

Dsniff

- Tools for network auditing and testing
- Passively monitor the network for *interesting* data
 - `dsniff`, `filesnarf`, `msgsnarf`, `urlsnarf`, and `webspay`
- Help in the interception of network traffic
 - `arp spoof`, `dnsspoof`, and `macof`
- Help in man-in-the-middle attacks against SSH and HTTPS
 - `sshmitm`, `webmitm`, and `nettturd`

Detecting Local Sniffers

- Sniffers are typically passive programs
 - Places the NIC in *promiscuous* mode
- Can be detected in Unix environments with `ifconfig`

A terminal window titled "Terminal" with a black background and white text. It shows the output of the command `ifconfig eth0`. The output includes details about the network interface eth0, such as its link type (Ethernet), hardware address (00:10:4B:E2:F6:4C), IP address (192.168.1.18), broadcast address (192.168.1.255), netmask (255.255.255.0), and its current state (UP, BROADCAST, RUNNING, PROMISC, MULTICAST). It also shows statistics for RX and TX packets, errors, dropped packets, overruns, frame drops, carrier drops, collisions, and txqueuelen.

```
> ifconfig eth0
eth0  Link encap:Ethernet HWaddr 00:10:4B:E2:F6:4C
       inet addr:192.168.1.18 Bcast:192.168.1.255 Mask:255.255.255.0
       UP BROADCAST RUNNING PROMISC MULTICAST MTU:1500 Metric:1
       RX packets:108 errors:0 dropped:0 overruns:0 frame:0
       TX packets:208 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:100
```

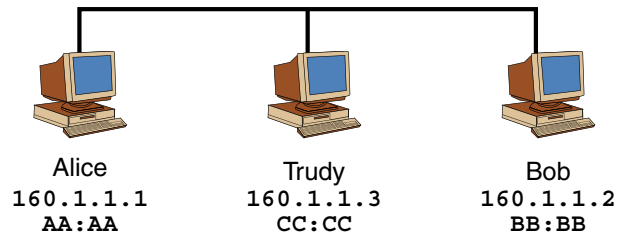
What if a rootkit has been installed?

Other Ways to Detect a Sniffer

- Suspicious DNS queries
 - Sniffer will attempt to resolve names with IP addresses
 - To detect, generate a connection from a fake IP address at another computer and watch if other computers attempts to resolve
- Higher network latency
 - If the NIC is promiscuous then it will resolve every packet
 - To detect, use `ping` to detect the response time
- Weird kernel behavior
 - Linux accepts packets with wrong Ethernet address but correct IP
 - Windoze, only the first octet is checked for broadcast addresses, so `ff:00:00:00:00:00` will be accepted
- Tools for detecting sniffers use these principles

Spoofing

- The idea behind spoofing is to masquerade as another computer
 - Allows a *(wo)man-in-the-middle* attack
- Why not set the IP address as another machine?



- Trudy can reset her IP address and masquerade as Bob

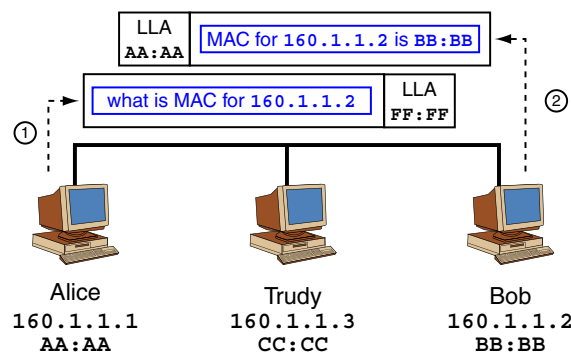
```
ifconfig eth0 160.1.1.2 up
```

- However, this will not work due to ARP

ARP Review

ARP seeks to determine a MAC address given an IP address

- Computer would broadcast “If your IP address is $w.x.y.z$, then please send me your MAC address”
- Computers receive request and the *correct* computer **replies**



How do you know the request is broadcasted?

- Once an ARP reply is received it is added to table (cache)
 - Cache increases performance

```

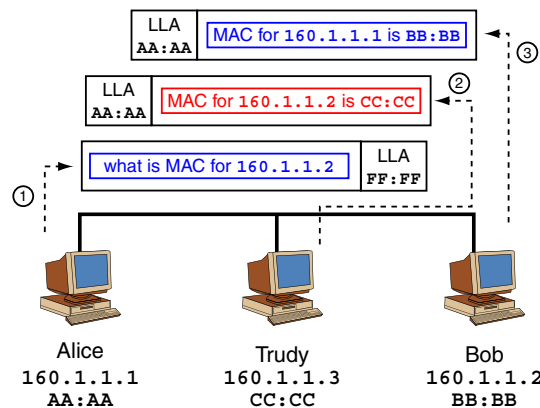
Terminal
> arp -a
Interface Address      Physical Address
-----
152.17.1.7            00:00:0e:01:01:0f
152.17.8.14           00:00:5e:00:01:0f
152.17.140.92         00:02:fc:47:5f:de

```

- Since it is a cache, entries expire
- For this reason, ARP is considered a *stateless* protocol

Why not keep all ARP entries forever?

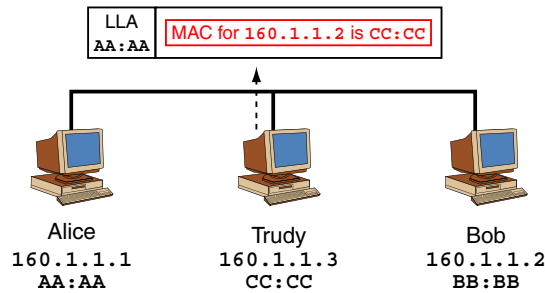
- If Trudy resets her IP address to Bob's, a race condition results
 - Trudy and Bob reply to Alice's ARP request



- Last ARP reply is stored by Alice (never certain which)
- *System admin should be looking for this behavior...*

ARP Spoofing

- A simple way of preventing the ARP *race* condition
 - Trudy could send an *unsolicited* ARP reply to Alice



- Alice would store the entry in her ARP table (*ARP poison*)
- Anytime Alice would send to Bob, she would send to Trudy

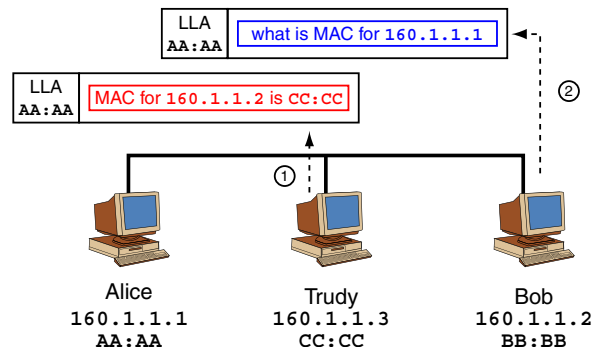
Won't Bob see the ARP spoof from Trudy?

Hello, are you there? ARP

- Many systems (including Linux) try to update ARP tables
 - System will occasionally go through the ARP table and send ARP requests for each entry

Why is this advantageous?

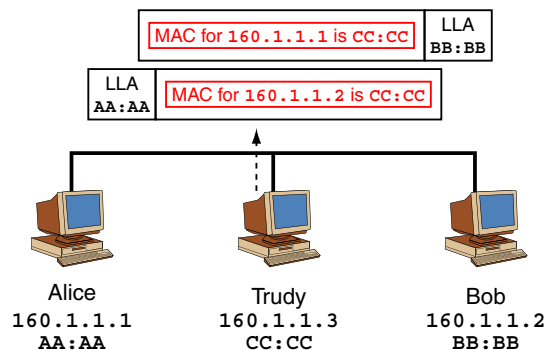
- So, Bob would send an ARP request to Alice for her MAC



- Once Alice receives the ARP request, she would reply
 - One of two scenarios could occur
 1. Bob would never receive a reply, thinking Alice (or he) is disconnected he calls the admin
What happened to the reply?
 2. Alice receives the request and updates her table with Bob's real MAC (removing the poison)
 - In either case, the spoof is over...

Feeding Both Sides Poison

- To keep both sides quite, Trudy should *feed* both sides
 - Trudy tells Alice Bob's MAC is CC:CC
 - Trudy tells Bob Alice's MAC is also CC:CC



- Of course since these are caches, Trudy must constantly send
 - For example, send ARP poison every 40 seconds

More ARP Damage

- ARP spoofing is very dangerous
 - It is a layer 2/3 protocol
 - Independent of platform (everyone is susceptible)
- ARP can be used for more than Man-in-the-Middle (MiM) attacks
 - Constantly sending a victim a nonexistent MAC address
“could have a quite a spectacular effect on one's [the victim's] mental health” - Yuri Volobuev
 - Broadcast the wrong address of an important computer
What is the effect? Can you give an example?

Anyway to prevent these attacks?

Static ARP Tables

- One method of preventing ARP attacks are static tables
 - Make certain (or all) ARP table entries *static*
 - Enter the important IP/MAC addresses manually

Will this work for the general campus at WFU?

What else can be done to limit or eliminate ARP attacks?

MAC Layer DoS

- Consider a 802.11b network
 - Uses virtual channel sensing before sending
- Several methods for causing DoS
 - Frequency jamming, *trivial...*
 - Set NAV field (15 bits) to 32767
 - Send forged de-authentication frame to AP

Layer 3 Attacks

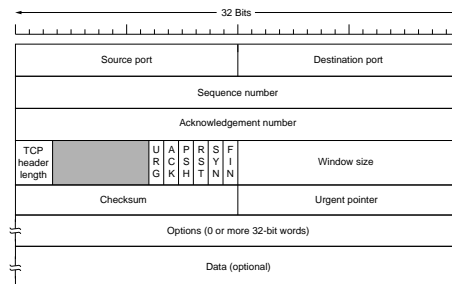
- Network layer concerns forwarding packets from one LAN to another
 - Packets are forwarded from device to device
 - Routing is the primary responsibility of this layer
- Example implementations include IPv4 and IPv6
 - We'll also consider ICMP since it helps manage layer 3

Constructing Custom Packets

- BSD sockets library commonly is used for network communications
 - Can create TCP connections, send TCP/UDP packets, etc...
 - However we never considered packet creation, *is it possible to set arbitrary addresses for source and destination?, Can we arbitrarily set packet header flags?, Will anyone hire Hobo?, ...*
Is there a need?
- libnet-dev is a framework for low-level network packet construction
 - Possible to create and inject *arbitrary* packets

Fragmentation

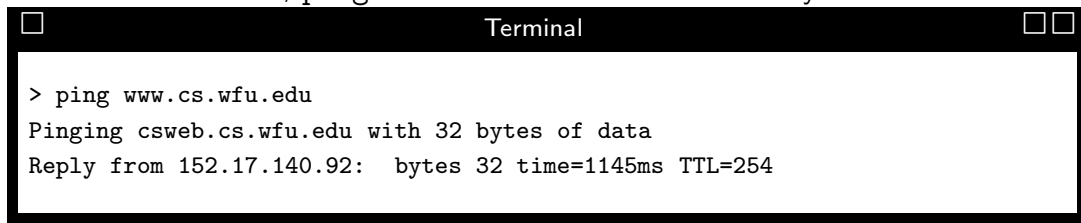
- A packet encapsulated in a frame might require fragmentation
 - Occurs when packet is larger than what a frame can carry (MTU)
 - Can occur at any hop in the network
 - A packet can request “*do not fragment*”, but if fragmentation is necessary the packet is dropped (ICMP message sent to source)



- If packet can be fragmented then, header copied to each fragment, more fragment bit set and fragment offset value set

Ping O' Death

- As we have seen, ping is used to determine connectivity



```
Terminal
> ping www.cs.wfu.edu
Pinging csweb.cs.wfu.edu with 32 bytes of data:
Reply from 152.17.140.92:  bytes=32 time=1145ms TTL=254
```

- ping command also allows you to specify the data size
 - Helpful for determining delays and fragmentation
- Some *older* (pre 1996) systems would crash if a ping message greater than x bytes was received



```
Terminal
> ping -l 65510 gavin.borg.com
```

Fragmentation and Evasion

- Firewalls and IDS analyze packet headers and payloads
 - Compare headers to a list, inspect packet payloads, etc...
- Fragmentation can be used to evade the firewall
 - Some firewalls make a decision based on the first fragment

So what, the packet header is copied?

"The idea is to split up the TCP header over several packets to make it harder for packet filters, intrusion detection systems, and other annoyances to detect what you are doing" (nmap guide)

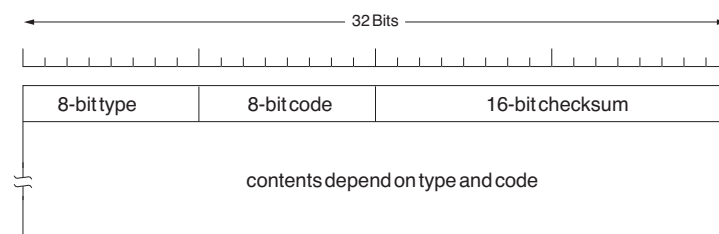
- Possible (*with overlapping fragments*) to rewrite ports
- Given fragments, an IDS might try to not reassemble
 - As a result, a threat might be missed

What?

- If the IDS will reassemble the packet then an attacker could
 - Create fragments that cannot be reassembled (missing pieces)
 - IDS would store the fragments until...

Internet Control Message Protocol

- Operation of the Internet is monitored by routers
 - If something unexpected occurs, event is reported to **ICMP** (Internet Control Message Protocol) RFC[792]
 - ICMP messages are acted on by the IP layer or transport layer
- ICMP messages are transmitted within IP datagrams



- ICMP can also be used to make requests and responses
- Unfortunately no security mechanisms included

ICMP Redirect

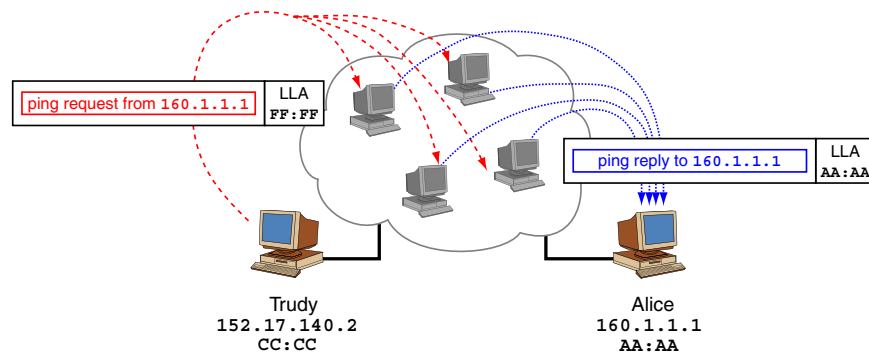
- Redirect message is used to change a network route
 - Can be sent to hosts or routers
 - Verification: the new router (next hop) must be directly connected, redirect cannot tell host to use itself, route being modified must be an indirect route

"RFC1122 states that redirects should only be sent by gateways and should not be sent by Internet hosts."

- Attacker can create spoofed messages to reroute traffic
 - Attacker can hijack traffic or perform DoS attack

Smurf Attacks

- Also called *broadcast attacks*
 - Rely on directed broadcast to victim
 - Creates a flood of traffic to a victim
- Smurf attack
 - Broadcast a ping request to a LAN and give the source address as the victim
 - Machines in the LAN will send a ping reply back to the victim
 - The victim will be overwhelmed with ping replies
 - The LAN is considered a **smurf amplifier**



- Trudy initiates a smurf attack against Alice
 - Forged ping request from Alice to all computers in LAN
 - Every computer responds back to Alice
- Defense against...as system admin
 - Be certain your network is **not** an amplifier
 - Attack can be traced back to amplifier

Route Hijacking

- YouTube address space is 208.65.152.0/22
 - `11010000.01000001.10011000.00000000`
 - For example youtube.com is 208.65.153.238, ...
- In February 2008 Pakistan telecom advertised new route
 - Route was for 208.65.153.0/24
 - `11010000.01000001.10011001.00000000`
 - So what?*
 - Internet thought 208.65.153.238 is part of the ISP
- Created an outage, although resolved in two hours
 - Can this easily happen again?*

Reflector Attacks

- A reflector is a component that responds to packets
 - Attack sends forged packets to reflectors
 - Reflectors send response to victim
- Examples include ICMP, TCP, and DNS

Have we seen this before?

Layer 4 Attacks

- Transport layer concerned with end to end communication
 - Only implemented at the source and destination
 - Performs error control, flow control, etc...
- Examples include TCP and UDP

TCP Review

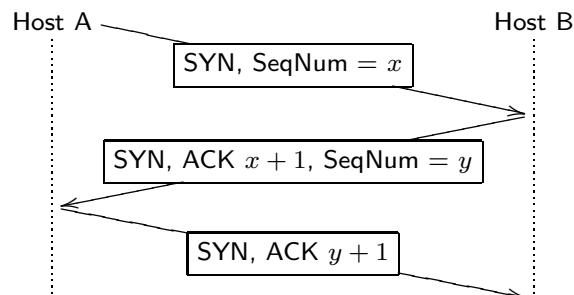
- Remember, the Internet has two basic types of services
 - UDP provides unreliable connectionless service
 - TCP provides reliable connection-oriented service
- *What does TCP reliable connection-oriented mean?*
 - Deliver all data without error and proper order
 - Must include **sequence numbers** with packets

Why sequence numbers?

 - Requires three-way handshake to establish connection
- Three-way handshake helps initiate the sequence numbers
 - Helps answer the question “*what number will you send first?*”

TCP Handshake

- Assume A and B wish to communicate using TCP
 1. A sends SYN to B giving its starting sequence number x
 2. B acknowledges (ACK) and sends its sequence number y
 3. A ACKs the sequence number of B



- At this point both sides agree on sequence numbers, regular TCP transmission can begin

TCP Problems

- Each TCP connection has an associated state
 - IP addresses, port numbers, and sequence numbers
 - Client and server have this information
 - Unfortunately, it is *typically* easy to guess this information
So what?
- Another problem is when a program authenticates
 - Applications usually only authenticate only once
 - The authentication is done at the beginning of the session
 - For example, telnet or reading email

TCP Hijack

- End users (applications) of a TCP connection will trust packets from each other if the sequence numbers are correct
 - Initial sequence numbers should be *random*
 - If attacker guesses correctly, then can possibly *hijack/spoof*
- If attacker connected to the same LAN or is a hop on the route
 - Sniff the sequence numbers (*no guessing required*)
 - Use ARP poisson to spoof the destination
 - Continue communications with the original source

What about a remote (non-local) session?

Remote Hijack

- Remote hijacking is more difficult, (called *blind spoofing*)

Why is this attack more difficult?

- Steps in the attack
 - Guess the initial sequence number
 - DoS the destination (who Trudy pretends to be), will also prevent the real destination from sending an RST
 - Spoof the IP address of the destination and send a SYN
 - Send packets with the data you want, for example
`cat "pluf.com" >> ~/.rhost`

Three requirements for a successful TCP hijack/spoof?

TCP DoS

- The previous spoofing method can be used as a Denial of Service
- Suppose attacker can guess SN for an existing connection
 - Attacker can send Reset (RST) packet to close connection
 - This results in a Denial of Service (DoS)
 - Naively, success prob is $\frac{1}{2^{32}}$ (32-bit SN)
- Most systems allow for a large window of acceptable SN
 - As a result, much higher success prob of successful attack

Therefore are long connections more susceptible?

Window scaling make this better or worse?

Choosing TCP Sequence Numbers

- RFC defines ways to improve sequence numbers
 - Sadly some implementations still don't
- “Strange Attractors and TCP/IP Sequence Number Analysis” and its update “One Year Later” by Zalewski
 - Measured the randomness of TCP sequence for different OS
 - Values were depicted in a 3-D graph, if s_i is the i^{th} random value of the sequence

$$x = s_i - s_{i-1}$$

$$y = s_{i-1} - s_{i-2}$$

$$z = s_{i-3} - s_{i-4}$$

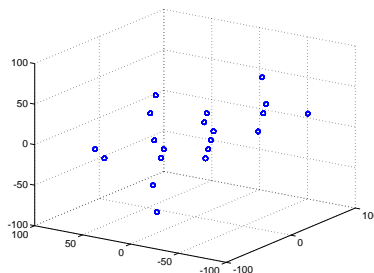
- When graphed, want to see points randomly distributed

Not So Random Random Values

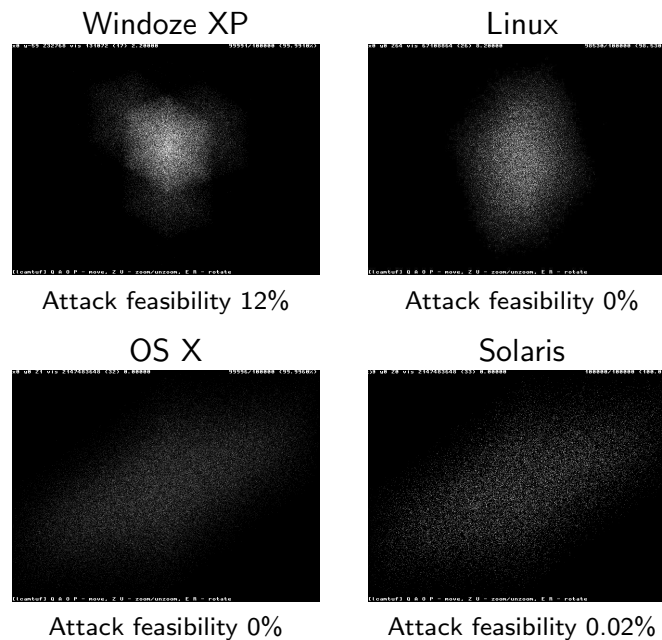
- The follow was generated using a bad linear congruential generator

$$s_{i+1} = (a \cdot s_i + c) \mod m$$

```
18 41 12 79 38 81 92 39 58 21 72 99 78 61 52 59 98 1 32 19 18 41 12 79 38 81 92 39 58 21 72 99 78
61 52 59 98 1 32 19 18 41 12 79 38 81 92 39 58 21 72 99 78 61 52 59 98 1 32 19 18 41 12 79 38 81
92 39 58 21 72 99 78 61 52 59 98 1 32 19 18 41 12 79 38 81 92 39 58 21 72 99 78 61 52 59 98 1 32
19 18 41 12 79 38 81 92 39 58 21 72 99 78 61 52 59 98 1 32 19 18 41 12 79 38 81 92 39 58 21 72 99
78 61 52 59 98 1 32 19 18 41 12 79 38 81 92 39 58 21 72 99 78 61 52 59 98 1 ...
```

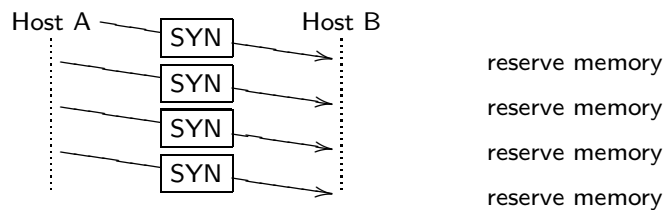


TCP Sequence Numbers for Different OS



Half Open Connection

- *September 6, 1996, Mail service for an ISP in NY, was shut down by a simple SYN flood. A week later the story was covered by the Wall Street Journal, the Washington Post, and many others*
 - An early example of a SYN-ACK attack
- To start a TCP connection must have three-way handshake
 - Source sends a SYN packet
 - Destination sends SYNACK packet back
 - *What if the source does not reply?* (last part of handshake)
 - This is called a **half opened connection**



- Every received SYN packet requires resources to be allocated
 - Memory reserved to store connection information

What type of information?

SYN Flood Attack

- Flood the destination with SYN packets, but never reply
 - Every SYN packet reserves memory, the reservation will remain until a time-out occurs (could be several minutes)
 - Legitimate connections will be refused, since no memory
 - Linux 1.2.x log is 10 entries and timeout is 3 minutes...
- This is a popular attack against web and email servers
 - HTTP and SMTP use TCP connections *Why?*
- It is very easy to trace this attack
 - Should always use a *forged* source address
 - However the forged address should be non-existent

Why? Any defense? More connections? Shorter timeouts?

SYN-ACK Cookies

- When the table of pending connections is full

- Server responds to client with *SYN-ACK cookie*

$$SN = h(s_1, s_{ip}, s_p, d_{ip}, d_p, s_1) + n + 2^{24}t + [h(s_2, s_{ip}, s_p, d_{ip}, d_p, s_2) \bmod 2^{24}]$$

“ h is a hash function, s_1 and s_2 are random secrets, s_A and s_p are addresses and ports, t is a counter incremented every minute, and n is the sequence number of the packet...”

- Sends the SYN-ACK with the SN above

Does the server need to maintain state?

- Client responds with ACK(SN + 1)

- Server checks response by recalculating the SN

- If correct, establish connection and reserve memory

Why use a formula? Can the attacker send the ACK(SN + 1)?

Other TCP DoS Approaches

- Sending a RST closes a connection and can be used as a DoS

- *Are there other flags that have the same result?*

- Consider a SYN flag, used to initiate a connection

- If received during a connection, considered an error

- Receiver closes the connection and sends RST [RFC 793]

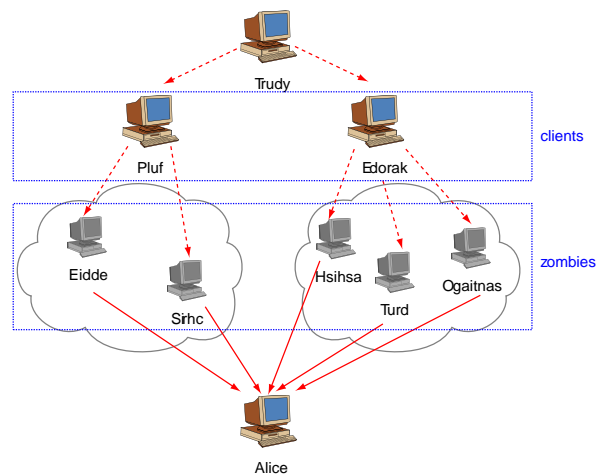
- This is referred to as *reflecting the packet*

Network reflection is actually another attack, how?

- N.B. network reflection attack is against the availability of a resource (DoS), however the term *reflection attack* also refers to an attack against authentication

Distributed Denial of Service

- Magnitude of DoS attacks
 - SYN flood - one machine attacks another machine
 - Smurf - one network attacks another machine
 - Distributed DoS - multiple networks attack another machine
- Architecture of Distributed DoS (DDoS)
 - Attacker gains access to a remote machine, installs and runs an attack program (continually running *background* process)
 - The compromised machine (with the process) is now a **zombie**, awaiting the command to attack
 - Attacker sends orders to zombies to attack a victim
 - Attack can be, SYN flood, smurf attack, etc...




- Trudy gains access to multiple machines
 - Trudy starts attack by signaling clients (distributed control)
 - Clients signal zombies, who flood Alice with traffic

- *What type of signaling?*
 - Clients and zombies can be signaled with ping
 - The signal may include an *unusual* address
 - Or zombie process may open a port and listen...
- Once DDoS occurs it is difficult to trace
 - Control is distributed and attacker never signals zombies
 - In addition, only one zombie per network is needed
 - This is an area of research, current methods attempt to profile traffic, keep state information per router, etc...

Don't the packets indicate where the traffic is from?

Port Scanning

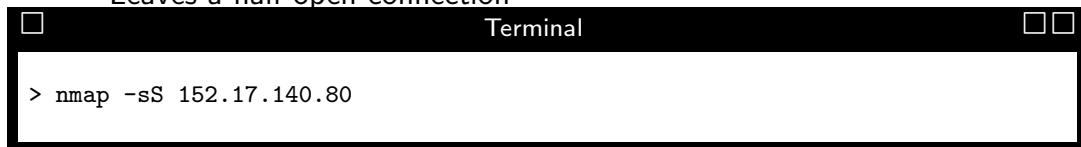
- Used to determine the services running on host
 - Can scan for TCP or UDP services
- TCP connect scan, just try to make a connection
 - Try all 65535 ports, if connection then service running
 - Requires 3 way handshake, so it can take some time

A terminal window titled "Terminal" with a black title bar and window controls. The terminal has a white background and shows the command `> nmap -sT 152.17.140.80` entered at the prompt.

```
> nmap -sT 152.17.140.80
```

- TCP SYN scan, just try to make a *half open* connection
 - Attacker send SYN to each port
 - if server replies with SYNACK then consider service open

- Leaves a half open connection

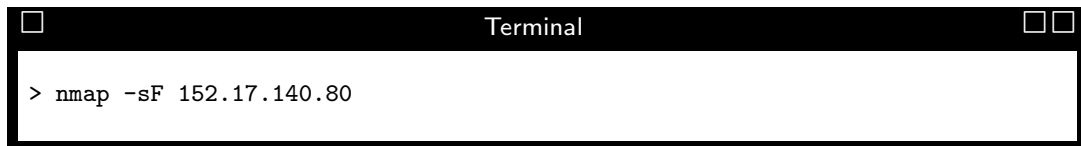


```
Terminal
> nmap -sS 152.17.140.80
```

- TCP FIN scan, attacker sends a FIN (close connection message)

Doesn't the connection need to be open first?

- Server will respond with a RST if the port is closed
- Server will ignore if the port is open
- *Windoze sends RST in all cases... a good feature?*



```
Terminal
> nmap -sF 152.17.140.80
```

What about UDP, there is no connection to establish?

UDP Port Scanning

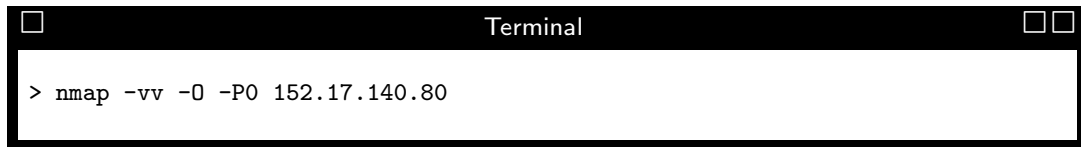
- Send a zero length packet to each port
 - If ICMP message *"port unreachable"* then assume service is closed
- OS might limit the error rate sent (Windoze does not limit)
 - Therefore this type of scan can be slow



```
Terminal
> nmap -sU 152.17.140.80
```

OS Fingerprint

- Determine the OS at the server/host using *crafted* packets
 - The response will help identify the OS
- Some reactions that might occur
 - Wrong answers to FIN packets
 - *Undefined* flags in the crafted packet are copied in the response
 - Selection of the TCP initial sequence number and window size
 - Analysis of ICMP messages (error rate)

A terminal window titled "Terminal" with a black background and white text. The command entered is: > nmap -vv -O -P0 152.17.140.80. The window has standard window control buttons (minimize, maximize, close) in the top right corner.

```
> nmap -vv -O -P0 152.17.140.80
```

- *You can use this information to fake your OS as well...*

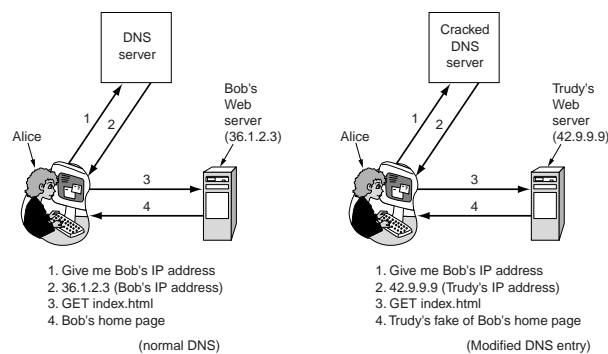
Layer 7 Attacks

- Concerns applications using the network
 - Any application that uses the network operates at layer 7
- Example layer 7 applications include DNS, HTTP, email, etc...

Secure Naming

- Assume Alice wants to visit Bob's web site
 - She types in Bob's URL and receives a web page
 - *Is the page really Bob's?*
- If Trudy can intercept the messages between Alice and Bob
 - Can send her own version of the pages to Alice
 - Could Alice ever really know?*
- This attack requires Trudy to be able to intercept messages
 - *However a simple trick removes this requirement*

DNS Spoofing



- Assume Trudy can break into the DNS system
 - Assigns her IP address to Bob's name
 - When Alice wants Bob's page, DNS will return Trudy's IP
 - Trudy can serve pages as if she was Bob
- Requires breaking into DNS, *but there is an easier way...*

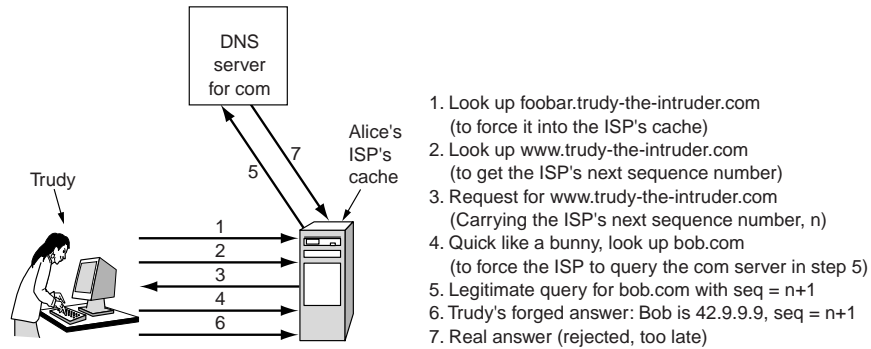
DNS Updates

- Want to cause DNS to give Trudy's IP for Bob's name
 - Unfortunately (*in terms of security*), DNS uses UDP
What is the problem?
 - Trudy can forge a DNS reply with her IP and Bob's name
- How the attack progresses
 1. Trudy asks DNS of Alice's ISP for Bob's (bob.com) IP
 2. Alice's DNS asks the top-level .com DNS
 3. Trudy sends her fake response to DNS of Alice's DNS
 4. The second *real* reply will be discarded
- This is **DNS spoofing** and it utilizes **cache poisson**
 - *Actually the attack is not that easy...*

Realistic DNS Spoofing

- DNS does some checking on the *reply* messages
 - Makes certain that the source address is a valid DNS server
 - Each request has a sequence number, so the server knows which reply goes with which request
 - So Alice needs a valid address and sequence number, *easy...*
- Assume Trudy registers the DNS server dns.trudy.com
 - She makes Alice's DNS aware of the server by asking Alice's DNS,
What is the IP of comp1.trudy.com?
- With dns.trudy.com in Alice's DNS cache, the attack begins
 1. Trudy queries Alice's DNS for comp2.trudy.com

2. Alice's DNS asks Trudy's DNS for the IP address of comp2.trudy.com (she receives a valid sequence number, SN)
3. Trudy then asks Alice's DNS for the IP address of bob.com
4. Trudy forges a reply about bob.com, faking the IP address and using the sequence number (SN + 1)
5. The forged reply is accepted and cached



dig

- dig is a Unix command for DNS queries
 - Can look up IP addresses from hostnames (and vice versa)
- Simple example, *what is the address of www.wfu.edu?*

```

Terminal
> dig www.wfu.edu
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14497
;; flags: qr rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.wfu.edu. IN A
;; ANSWER SECTION:
www.wfu.edu. 600 IN A 152.17.48.78
;; Query time: 46 msec
  
```

- Shorter answer to the previous question

```
Terminal
> dig www.wfu.edu +short
152.17.48.78
```

- Reverse lookup, *What is the hostname for 152.17.49.95?*

```
Terminal
> dig -x 152.17.49.95 +short
inti.cs.wfu.edu
```

- Ask a certain nameserver, *What is the address for www.wfu.edu?*

```
Terminal
> dig @ns1.wfu.edu. www.wfu.edu +short
152.17.48.78
```

- *What are mail servers for wfu.edu?*

```
Terminal
> dig wfu.edu. mx
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22198
;; flags: qr rd; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;wfu.edu. IN MX
;; ANSWER SECTION:
wfu.edu. 3600 IN MX 10 mx3.wfu.edu.
wfu.edu. 3600 IN MX 10 mx1.wfu.edu.
wfu.edu. 3600 IN MX 10 mx2.wfu.edu.
;; Query time: 43 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
```

– Can also find name servers (NS), ...

What about an entire DNS tree traversal?


```
Terminal
dig www.wfu.edu +trace
; <<> DiG 9.6.0-APPLE-P2 <<> www.wfu.edu +trace
;; global options: +cmd
. 386116 IN NS l.root-servers.net.
. 386116 IN NS f.root-servers.net.
. 386116 IN NS m.root-servers.net.
. 386116 IN NS g.root-servers.net.
. 386116 IN NS i.root-servers.net.
. 386116 IN NS d.root-servers.net.
. 386116 IN NS b.root-servers.net.
. 386116 IN NS e.root-servers.net.
. 386116 IN NS c.root-servers.net.
. 386116 IN NS h.root-servers.net.
. 386116 IN NS j.root-servers.net.
. 386116 IN NS k.root-servers.net.
. 386116 IN NS a.root-servers.net.
;; Received from 192.168.1.254#53(192.168.1.254) in 12 ms
edu. 172800 IN NS c.edu-servers.net.
edu. 172800 IN NS a.edu-servers.net.
edu. 172800 IN NS d.edu-servers.net.
edu. 172800 IN NS g.edu-servers.net.
edu. 172800 IN NS f.edu-servers.net.
edu. 172800 IN NS l.edu-servers.net.
;; Received from 192.5.5.241#53(f.root-servers.net) in 96 ms
wfu.edu. 172800 IN NS ncnoc.ncnec.net.
wfu.edu. 172800 IN NS reggae.ncnec.net.
wfu.edu. 172800 IN NS ns1.wfu.edu.
wfu.edu. 172800 IN NS ns3.wfu.edu.
wfu.edu. 172800 IN NS ns2.wfu.edu.
wfu.edu. 172800 IN NS ns4.wfu.edu.
;; Received from 192.41.162.36#53(l.edu-servers.net) in 23 ms
www.wfu.edu. 600 IN A 152.17.48.78
wfu.edu. 600 IN NS reggae.ncnec.net.
wfu.edu. 600 IN NS ns4.wfu.edu.
wfu.edu. 600 IN NS ns3.wfu.edu.
wfu.edu. 600 IN NS ns1.wfu.edu.
wfu.edu. 600 IN NS ns2.wfu.edu.
wfu.edu. 600 IN NS ncnoc.ncnec.net.
;; Received from 128.109.131.3#53(reggae.ncnec.net) in 36 ms
```

DNS Glue

- Suppose authoritative name server for wfu.edu is ns1.wfu.edu
 - Consider resolving www.wfu.edu returns ns1.wfu.edu
 - Attempting to resolve ns1.wfu.edu requires resolving wfu.edu, which is managed by ns1.wfu.edu, which requires resolving wfu.edu, which is managed by ns1.wfu.edu, etc...
- DNS *glue entry* solves a circular dependency, the glue record provides the IP address of authoritative name server
 - In our example DNS would store 152.17.239.1 for ns1.wfu.edu

"Name servers in delegations are identified by name, rather than by IP address. This means that a resolving name server must issue another DNS request to find out the IP address of the server to which it has been referred. If the name given in the delegation is a subdomain of the domain for which the delegation is being provided, there is a circular dependency."

Kaminsky DNS Vulnerability

1. Query a random host in a domain you want to take over
 - For example `torggy.cs.wfu.edu`
 - Request sent to a non-WFU DNS, like the attacker's ISP
2. ISP DNS does not know `torggy.cs.wfu.edu`, must query
3. Spoof responses, but **delegate authority** to some server you own
 - Alter the glue record to be your server, that gives you authority
4. Congrats! you have powned the domain

NFS

- Two steps required for a client to gain access to a file on a server
 - Mount access then file access
- Mount access is achieved by client attempting to attach to server
 - Security for this is provided by the `/etc/exports` file
 - File lists names or IP addresses of those allowed to access
 - Therefore this step is vulnerable to *spoofing*
- File access is a function of normal file access controls on the client
 - User and group permissions on files determine access control

So what? What type of authentication is done?

- Assume, Alice on the server maps to UserID 9999. Alice makes a file on the server that is only accessible by her. A client is allowed to mount the drive where the file is stored. On the client Trudy maps to UserID 9999. So the client user Trudy can access Alice's file that is marked as only accessible by her.

What if you are root on the client machines? Other userIDs?

- There is a solution to clients with userID root
 - Enable root_squash option in NFS
 - Prevents the server from trusting anyone mounting as root

What about the information legitimately sent using NFS?

"When an intruder has access to your network, s/he can make strange commands appear in your .forward or read your mail when /home or /var/mail is NFS exported. For the same reason, you should never access your PGP private key over NFS. Or at least you should know the risk involved."

Other Application Problems

- Post Office Protocol (POP) email retrieval
 - Password sent in cleartext *awesome!*
 - Should use with SSL or SSH, or another mail retrieval protocol
- Simple Mail Transport Protocol (SMTP) transport email
 - No authentication done, therefore easy to send SPAM
 - Information is not encrypted
- File Transport Protocol (FTP) sending data
 - Password sent in cleartext
 - Should use with SSL or just use SSH (sftp)