Review: Sniffing & Spoofing
A & B can see everything each other sends if they’re on the same open or WPA-Personal WiFi network.
Because of this, B can spoof DHCP offers to A, and vice versa. But no one else can, because the requests stay within A’s subnet.
R can see anything A, B or C send out to the Internet ... and any replies sent back to them
Thus, R can do successful TCP or DNS injection on them … … other than for local traffic such as between A & B
...since R can't see what A sends to B or vice versa
C can’t see any of A or B’s traffic since C is on a different subnet. C likewise can’t see R’s traffic.
D can’t see E’s traffic nor any traffic from the Berkeley Network unless it happens to be directed to D.
Like all Internet hosts, D can spoof whatever packet fields D desires … BUT
BUT it’s a separate question whether those spoofs will succeed. The use of randomized fields in TCP & DNS make this very hard.
Controlling Networks Using Firewalls
Controlling Networks … On The Cheap

• Motivation: How do you harden a set of systems against external attack?
  – Key Observation:
    • The more network services your machines run, the greater the risk
  – Due to larger attack surface

• One approach: on each system, turn off unnecessary network services
  – But you have to know all the services that are running
  – And sometimes some trusted remote users still require access

• Plus key question of scaling
  – What happens when you have to secure 100s/1000s of systems?
  – Which may have different OSs, hardware & users …
  – Which may in fact not all even be identified …
Taming Management Complexity

- Possibly more scalable defense: Reduce risk by blocking *in the network* outsiders from having unwanted access to your network services
  - Interpose a **firewall** that traffic to/from the outside must traverse
  - **Chokepoint** can cover 1000s of hosts
Selecting a Security Policy

• Effectiveness of firewall relies on deciding what policy it should implement:
  – Who is allowed to talk to whom, accessing what service?

• Distinguish between inbound & outbound connections
  – Inbound: attempts by external users to connect to services on internal machines
  – Outbound: internal users to external services
  – Why? Because fits with a common threat model
Selecting a Security Policy

- Effectiveness of firewall relies on deciding what policy it should implement:
  - Who is allowed to talk to whom, accessing what service?

- Distinguish between **inbound** & **outbound** connections
  - Inbound: attempts by external users to connect to services on internal machines
  - Outbound: internal users to external services
  - Why? Because fits with a common *threat model*

- Conceptually simple *access control policy*:
  - Permit inside users to connect to any service
  - External users restricted:
    - Permit connections to services meant to be externally visible
    - Deny connections to services not meant for external access
How To Treat Traffic Not Mentioned in Policy?

• **Default Allow**: start off permitting external access to services
  – Shut them off as problems recognized
How To Treat Traffic Not Mentioned in Policy?

- **Default Allow**: start off permitting external access to services
  - Shut them off as problems recognized

- **Default Deny**: start off permitting just a few known, well-secured services
  - Add more when users complain (and mgt. approves)
How To Treat Traffic Not Mentioned in Policy?

- **Default Allow**: start off permitting external access to services
  - Shut them off as problems recognized
- **Default Deny**: start off permitting just a few known, well-secured services
  - Add more when users complain (and mgt. approves)

- Pros & Cons?  
  - Flexibility vs. conservative design
  - Flaws in Default Deny get noticed more quickly / less painfully

*In general, use Default Deny*
Packet Filters

• Most basic kind of firewall is a *packet filter*
  – Router with list of *access control rules*
  – Router checks each received packet against security rules to decide to *forward* or *drop* it
  – Each rule specifies which packets it applies to based on a packet’s header fields (*stateless*)
    • Specify source and destination IP addresses, port numbers, and protocol names, or *wild cards*
IP Header

- 4-bit Version
- 4-bit Header Length
- 8-bit Type of Service (TOS)
- 16-bit Total Length (Bytes)
- 16-bit Identification
- 3-bit Flags
- 13-bit Fragment Offset
- 8-bit Time to Live (TTL)
- 8-bit Protocol
- 16-bit Header Checksum
- 32-bit Source IP Address
- 32-bit Destination IP Address

TCP Header

- Source port
- Destination port
- Sequence number
- Acknowledgment
- HdrLen
- 0
- Flags
- Advertised window
- Checksum
- Urgent pointer

Data
Packet Filters

• Most basic kind of firewall is a *packet filter*
  – Router with list of *access control rules*
  – Router checks each received packet against security rules to decide to forward or drop it
  – Each rule specifies which packets it applies to based on a packet’s header fields (stateless)
    • Specify source and destination IP addresses, port numbers, and protocol names, or wild cards
    • Each rule specifies the *action* for matching packets: ALLOW or DROP (aka DENY)
      
      <ACTION>  <PROTO>  <SRC:PORT>  ->  <DST:PORT>

  – First listed rule has *precedence*
Examples of Packet Filter Rules

allow tcp 4.5.5.4:1025 -> 3.1.1.2:80

• States that the firewall should *permit* any TCP packet that’s:
  – from Internet address 4.5.5.4 *and*
  – using a source port of 1025 *and*
  – destined to port 80 of Internet address 3.1.1.2

deny tcp 4.5.5.4:* -> 3.1.1.2:80

• States that the firewall should *drop* any TCP packet like the above, regardless of source port
Examples of Packet Filter Rules

\[
\begin{align*}
\text{deny tcp 4.5.5.4:*} & \rightarrow 3.1.1.2:80 \\
\text{allow tcp 4.5.5.4:1025} & \rightarrow 3.1.1.2:80
\end{align*}
\]

- \textit{In this order}, the rules won’t allow \textit{any} TCP packets from 4.5.5.4 to port 80 of 3.1.1.2

\[
\begin{align*}
\text{allow tcp 4.5.5.4:1025} & \rightarrow 3.1.1.2:80 \\
\text{deny tcp 4.5.5.4:*} & \rightarrow 3.1.1.2:80
\end{align*}
\]

- \textit{In this order}, the rules allow TCP packets from 4.5.5.4 to port 80 of 3.1.1.2 \textit{only} if they come from source port 1025
Firewall Considerations

• Firewalls can have 1000s of filtering rules like these
  – Easy to introduce subtle errors 😞

• Provide not only security but also policy enforcement
  – E.g. do not allow company systems to access file-sharing sites

• Modern firewalls operate in a stateful fashion
  – Make Yes/No decisions upon establishment of a connection/flow
    • For Yes decisions, add 4-tuple to a connection table consulted for future traffic
    • Drop arriving non-establishment packet if not in table

• An important example of a reference monitor
Security Principle: Reference Monitors

- Firewalls embody useful principles that are applicable elsewhere in computer security
  - Optimized for enforcing particular kind of access control policy
  - Chokepoint notion makes enforcement possible
- A reference monitor examines every request to access a controlled resource (an object) and determines whether to allow request
Reference Monitor Security Properties

- **Always invoked**
  - *Complete mediation* property: all security-relevant operations must be mediated by RM
  - RM should be invoked on every operation controlled by access control policy
- **Tamper-resistant**
  - Maintain RM integrity (no code/state tampering)
- **Verifiable**
  - Can verify RM operation (correctly enforces desired access control policy)
    - Requires extremely simple RM
    - We find we can’t verify correctness for systems with any appreciable degree of complexity
Considering Firewalls as Reference Monitors

• Always invoked?
  – Place Packet Filter as an *in-path* element on *chokepoint* link for all internal-external communications
  – Packets only forwarded across link if firewall *explicitly decides* to do so after inspection
Potential Problems?

• What if a user hooks up an unsecured wireless access point to their internal machine?
• Anyone who drives by with wireless-enabled laptop can gain access to internal network
  – Bypasses packet filter!
• Or: what if user brings an infected device onto the premises?
• To use a firewall safely, must ensure we’ve covered all links between internal and external/untrusted networks with firewalls
  – Set of links known as the security perimeter
RM Property: *Tamper-Resistant*

- Will this hold?

- Do not allow management access to firewall other than from specific hosts
  - I.e., firewall itself needs firewalling

- Protect firewall’s physical security

- Must also secure storage & propagation of configuration data
RM Property: Verifiable

• Will this hold?
• Current practice:
  – Packet filter software too complex for feasible systematic verification …
  – … and rulesets with 1,000s (!) of rules
• Result:
  – **Bugs** that allowed attackers to defeat intended security policy by sending unexpected packets that packet filter doesn’t handle as desired
• In addition: challenging to ensure network topology does not allow internal access by untrusted devices
Why Have Firewalls Been Successful?

• **Central control** – *easy administration and update*
  – Single point of control: update one config to change security policies
  – Potentially allows rapid response

• **Easy to deploy** – *transparent to end users*
  – Easy incremental/total deployment to protect 1,000’s

• **Addresses an important problem**
  – Security vulnerabilities in network services are rampant
  – Easier to use firewall than to directly secure code …
Firewall Disadvantages?

• **Functionality loss** – *less connectivity, less risk*
  – May reduce network’s usefulness
  – Some applications don’t work with firewalls
    • Two peer-to-peer users behind different firewalls
• **The malicious insider problem**
  – Deployment assumes insiders are trusted
    • Malicious insider (or *anyone gaining control of internal machine*) can wreak havoc
• **Firewalls establish a security perimeter**
  – Like *Eskimo Pies:* “hard crunchy exterior, soft creamy center”
  – Threat from travelers with laptops, cell phones, …
5 Minute Break

Questions Before We Proceed?
Getting Around Firewalls
Subverting Firewalls

• Along with possible bugs, packet filters have a fundamentally limited semantic model
  – They lack a full understanding of the meaning of the traffic they carry
    • In part because operate only at layers 3 & 4; not 7

• How can a local user who wants to get around their site’s firewall exploit this?
  – *(Note: we’re not talking about how an external attacker can escape a firewall’s restrictions)*

• One method of subversion: abuse ports
  – Who says that e.g. port 53/udp = DNS?
    • Why couldn’t it be say Skype or BitTorrent?
    • Just requires that client & server agree on application protocol
The Internet
## Packet Sent to Halo Server

### IP Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td>4-bits</td>
</tr>
<tr>
<td>4-bit Header Length</td>
<td>4-bits</td>
</tr>
<tr>
<td>8-bit Type of Service (TOS)</td>
<td>8-bits</td>
</tr>
<tr>
<td>16-bit Total Length (Bytes)</td>
<td>16-bits</td>
</tr>
<tr>
<td>16-bit Identification</td>
<td>16-bits</td>
</tr>
<tr>
<td>3-bit Flags</td>
<td>3-bits</td>
</tr>
<tr>
<td>13-bit Fragment Offset</td>
<td>13-bits</td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td>8-bits</td>
</tr>
<tr>
<td>8-bit Protocol</td>
<td>8-bits</td>
</tr>
<tr>
<td>16-bit Header Checksum</td>
<td>16-bits</td>
</tr>
</tbody>
</table>

### Source IP Address

32-bit Source IP Address

### Destination IP Address

32-bit Destination IP Address

### IP Payload

---
## Packet Sent to Halo Server

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4-bit</td>
<td>Major version number</td>
</tr>
<tr>
<td>Header Length</td>
<td>4-bit</td>
<td>Total length of header</td>
</tr>
<tr>
<td>Type of Service (TOS)</td>
<td>8-bit</td>
<td>Type of Service</td>
</tr>
<tr>
<td>Total Length (Bytes)</td>
<td>16-bit</td>
<td>Total length of packet</td>
</tr>
<tr>
<td>Identification</td>
<td>16-bit</td>
<td>Unique identifier</td>
</tr>
<tr>
<td>Flags</td>
<td>3-bit</td>
<td>Flags for fragment control</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>13-bit</td>
<td>Offset for fragment</td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td>8-bit</td>
<td>Time remaining for packet</td>
</tr>
<tr>
<td>Protocol</td>
<td>8-bit</td>
<td>Protocol type</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>16-bit</td>
<td>Checksum of header</td>
</tr>
</tbody>
</table>

### Source (SRC)
- 7.7.8.2

### Destination (DST)
- 14.6.1.7
### Packet Sent to Halo Server

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IP Header</strong></td>
<td></td>
</tr>
<tr>
<td>4-bit Version</td>
<td></td>
</tr>
<tr>
<td>4-bit Header Length</td>
<td></td>
</tr>
<tr>
<td>8-bit Type of Service (TOS)</td>
<td></td>
</tr>
<tr>
<td>16-bit Total Length (Bytes)</td>
<td></td>
</tr>
<tr>
<td>16-bit Identification</td>
<td>SRC=7.7.8.2</td>
</tr>
<tr>
<td>3-bit Flags</td>
<td></td>
</tr>
<tr>
<td>13-bit Fragment Offset</td>
<td>DST=14.6.1.7</td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>DST=2302</td>
</tr>
<tr>
<td>16-bit Header Checksum</td>
<td></td>
</tr>
<tr>
<td><strong>UDP Header</strong></td>
<td></td>
</tr>
<tr>
<td>SRC=arbitrary</td>
<td></td>
</tr>
<tr>
<td>DST=2302</td>
<td></td>
</tr>
<tr>
<td><strong>UDP Payload</strong></td>
<td></td>
</tr>
<tr>
<td>Halo Data</td>
<td></td>
</tr>
</tbody>
</table>
The Internet

Enterprise network

router

7.7.8.2

14.6.1.7
2302/udp
7.7.8.2

Enterprise network

firewall

14.6.1.7
2302/udp

The Internet
The Internet

Enterprise network

firewall

deny udp *:* /int -> *:2302/ext

7.7.8.2

14.6.1.7
2302/udp
Hey *Halo* guys can you help me out and run on a different port?
Sure! Let’s see them try to block this!

```
deny udp *:*/int -> *:2302/ext
```
Sure! Let’s see them try to block *this*.
Enterprise network

firewall

... deny udp *:* /int -> *:2302 /ext
deny udp *:* /int -> *:53 /ext
...

7.7.8.2

14.6.1.7
53/udp

The Internet
Rule is impractical because it will block all legitimate DNS too!

\[\text{deny udp } *:* /int \rightarrow *:2302 /ext\]
\[\text{deny udp } *:* /int \rightarrow *:53 /ext\]
Enterprise network

The Internet

firewall

deny udp *:*/* -> *:2302
deny udp *:* -> 14.6.1.7/*

7.7.8.2

14.6.1.7
53/udp

HALO
COMBAT EVOLVED

Halo
The Internet

Enterprise network

user’s remote relay
9.9.1.1, 8822/udp

deny udp *:* /int -> *:2302/ext
deny udp *:* -> 14.6.1.7/*

firewall

7.7.8.2

14.6.1.7 53/udp

YODODYNE MONKEY WORKS

The Future Begins tomorrow!*
Mahler Falco-Nashville, TN 2013

SINCE 2013

HALO: Combat Evolved

...
Packet Sent to Halo Server

<table>
<thead>
<tr>
<th>IP Header</th>
<th>UDP Header</th>
<th>UDP Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td>4-bit Header Length</td>
<td>8-bit Time to Live (TTL)</td>
</tr>
<tr>
<td>8-bit Type of Service (TOS)</td>
<td>16-bit Total Length (Bytes)</td>
<td>UDP</td>
</tr>
<tr>
<td>16-bit Identification</td>
<td>3-bit Flags</td>
<td>13-bit Fragment Offset</td>
</tr>
<tr>
<td>SRC=7.7.8.2</td>
<td>DST=14.6.1.7</td>
<td>DST=53</td>
</tr>
<tr>
<td>SRC=arbitrary</td>
<td>checksum</td>
<td>length</td>
</tr>
<tr>
<td>Halo Data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Packet Sent to Remote Relay

<table>
<thead>
<tr>
<th>IP Header</th>
<th>UDP Header</th>
<th>UDP Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td>4-bit Header Length</td>
<td>8-bit Identification</td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td>UDP</td>
<td>16-bit Header Checksum</td>
</tr>
<tr>
<td>8-bit Type of Service (TOS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit Total Length (Bytes)</td>
<td>16-bit Identification</td>
<td>3-bit Flags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13-bit Fragment Offset</td>
</tr>
<tr>
<td>SRC=7.7.8.2</td>
<td>DST=9.9.1.1</td>
<td></td>
</tr>
<tr>
<td>SRC=arbitrary</td>
<td>DST=8822</td>
<td></td>
</tr>
<tr>
<td>checksum</td>
<td>length</td>
<td>“Please send appended to 14.6.1.7, UDP port 53”</td>
</tr>
<tr>
<td>Halo Data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Packet Sent by Remote Relay

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IP Header</strong></td>
<td><strong>UDP Header</strong></td>
</tr>
<tr>
<td>4-bit Version</td>
<td><strong>SRC=9.9.1.1</strong></td>
</tr>
<tr>
<td>4-bit Header Length</td>
<td><strong>DST=14.6.1.7</strong></td>
</tr>
<tr>
<td>8-bit Type of Service (TOS)</td>
<td><strong>SRC=arbitrary</strong></td>
</tr>
<tr>
<td>16-bit Total Length (Bytes)</td>
<td><strong>DST=53</strong></td>
</tr>
<tr>
<td>16-bit Identification</td>
<td><strong>checksum</strong></td>
</tr>
<tr>
<td>3-bit Flags</td>
<td><strong>length</strong></td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td><strong>UDP Payload</strong></td>
</tr>
<tr>
<td>13-bit Fragment Offset</td>
<td><strong>Halo Data</strong></td>
</tr>
</tbody>
</table>
Enterprise network

user's remote relay 9.9.1.1, 8822/udp

firewall

deny udp *:*/* - > *:2302/ext

deny udp *:* - > 14.6.1.7/*

The Internet

7.7.8.2

hdr1
data for server

14.6.1.7
53/udp
The Internet

Enterprise network

firewall

deny udp *::*/int -> *:2302/ext
deny udp *::* -> 14.6.1.7/*

user's remote relay 9.9.1.1, 8822/udp

data for server

hdr2

7.7.8.2

14.6.1.7 53/udp

HALO COMBAT EVOLVED
Enterprise network

firewall

user’s remote relay
9.9.1.1, 8822/udp

The Internet

deny udp *:*/* -> *:2302/udp

deny udp *:* -> 14.6.1.7/*

...
Enterprise network

firewall

7.7.8.2

Enterprise network

firewall

user's remote relay 9.9.1.1, 8822/udp

data for client 7.7.8.2

The Internet

deny udp *:*/* -> *:2302/ext

deny udp *:* -> 14.6.1.7/*

...
Hiding on Other Ports

- **Method #1**: use port allocated to another service (how can this be detected?)
- **Method #2**: tunneling
  - **Encapsulate** one protocol inside another
  - Receiver of “outer” protocol *decapsulates* interior tunneled protocol to recover it
  - Pretty much any protocol can be tunneled over another (with enough effort)
- E.g., tunneling IP over **SMTP** (email)
  - Just need a way to code an IP datagram as an email message (either mail body or just headers)
Example: Tunneling IP over Email

From: halo-nut@yoyodyne.com
To: my-buddy@tunnel-decapsulators.R.us
Subject: Here’s my IP datagram

IP-header-version: 4
IP-header-len: 5
IP,proto: 17 (UDP)
IP-src: 7.7.8.2
IP-dst: 14.6.1.7
IP-payload: 0xa144bf2c0102...

This operator of this email server has chosen to cooperate with the email sender to help them tunnel.

Remote email server receives this legal email, builds an IP packet corresponding to description in email body ... ... and injects it into the network.

How can a firewall detect this??
Network Control & Tunneling

• **Tunneling** = embedding one protocol inside another
  – Sender and receiver at each side of the tunnel both **cooperate** (so it’s **not useful for initial attacks**)

• Traffic takes on properties of outer protocol
  – Including for **firewall inspection**, which generally can’t analyze inner protocol (due to complexity)

• Tunneling has **legitimate** uses
  – E.g., Virtual Private Networks (VPNs)
    • Tunnel server relays remote client’s packets
    • Makes remote machine look like it’s **local** to its home network
    • Tunnel **encrypts** traffic for privacy & to prevent meddling
Other Ways of Securing Network Access
Secure External Access to Inside Machines

- Often need to provide secure remote access to a network protected by a firewall
  - Remote access, telecommuting, branch offices, …
- Create **secure channel** (*Virtual Private Network*, or *VPN*) to tunnel traffic from outside host/network to inside network
  - Provides **Authentication**, **Confidentiality**, **Integrity**
    - Requires some form of key management to set up
  - However, also raises *perimeter issues*
    (Try it yourself at http://www.net.berkeley.edu/vpn/)
Application Proxies

• Can more directly control applications by requiring them to go through a proxy for external access
  – Proxy doesn’t simply forward, but acts as an application-level middleman

• Example: SSH gateway
  – Require all SSH in/out of site to go through gateway
  – Gateway logs authentication, inspects decrypted text
  – Site’s firewall configured to prohibit any other SSH access
SSH Gateway Example

host-to-gateway SSH session

host-to-gateway host SSH session

gateway-to-remote

1.3.5.7

firewall

allow
<port=22, host=1.3.5.7>

drop <port=22>
Application Proxies

- Can more directly control applications by requiring them to go through a proxy for external access
  - Proxy doesn’t simply forward, but acts as an application-level middleman

- Example: SSH gateway
  - Require all SSH in/out of site to go through gateway
  - Gateway logs authentication, inspects decrypted text
  - Site’s firewall configured to prohibit any other SSH access

- Provides a powerful degree of monitoring/control

- Costs?
  - Need to run extra server(s) per app (possible bottleneck)
  - Each server requires careful hardening