Network Security

1. TCP/IP Basics
2. Spoofing
3. TCP Session Hijacking
4. Packet Fragmentation
5. Denial of Service Attacks
6. IPv6 Security Changes
7. Port Scanning
8. Firewalls
9. VPNs

TCP/IP

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>HTTP, FTP, telnet</td>
</tr>
<tr>
<td>Transport</td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>Network</td>
<td>IP, ICMP, IGMP</td>
</tr>
<tr>
<td>Network Access</td>
<td>PPP, 802.11, Ethernet</td>
</tr>
</tbody>
</table>
**Application Layer**

Applications have their own protocols.
- ex: FTP, HTTP, IRC, POP, SMTP, ssh, telnet

Request/response pattern
- Client requests data from the server.
- Server sends a response to client.

**Transport Layer**

Two main protocols: TCP and UDP

TCP guarantees delivery of data across network
- Error detection and recovery
- Sequence numbers and ACKs

**Network Layer**

IP transmits data across the network.
- Addressing
- Routing
Network Access Layer

Physically connects one computer to another. Common protocols
- Ethernet
- PPP
- 802.11

Encapsulation

Network Sniffing

- All ethernet frames to or from any locally connected host are seen by all hosts.
- NIC normally filters out frames that are not addressed to its MAC address.
- In promiscuous mode, NIC processes all ethernet frames, not just ones addressed to it.
  - Requires administrative access on most OSes.
ARP Spoofing

ARP Spoofing/Cache Poisoning:
- Send spoofed MAC address in response to sender’s ARP request
- Sender will cache response
- May need to stop response from correct host

Man-in-the-Middle Attack
- Send your MAC address in response to both Alice's and Bob's ARP responses
- Intercept and forward all traffic

Tools: ettercap, parasite

ARP Spoofing Defences

Enable switch MAC binding
- MAC address for a port is set once
Create static ARP table for local LAN
Arpwatch
- Builds table of IP/MAC bindings for LAN
- Sends notifications of any changes

IP: Internet Protocol

Unreliable, connectionless datagram service
- Packets may arrived damaged, out of order, duplicated or not at all.
- Transport/Application layers provide reliability.
IPv4 underlies Internet.
- 32-bit addresses in dotted-quad: 10.17.0.90.
- IPv6 is successor with 128-bit addresses.
Complexities: addressing, routing
IP Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
</tr>
<tr>
<td>Header Length</td>
<td>12</td>
</tr>
<tr>
<td>Type of Service</td>
<td>8</td>
</tr>
<tr>
<td>Total Length</td>
<td>16</td>
</tr>
<tr>
<td>Identification</td>
<td>12</td>
</tr>
<tr>
<td>Flags</td>
<td>3</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>16</td>
</tr>
<tr>
<td>Time to Live</td>
<td>8</td>
</tr>
<tr>
<td>Protocol</td>
<td>8</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>16</td>
</tr>
<tr>
<td>Source Address</td>
<td>32</td>
</tr>
<tr>
<td>Destination Address</td>
<td>32</td>
</tr>
<tr>
<td>Options</td>
<td>0-40</td>
</tr>
<tr>
<td>Data</td>
<td>Variable</td>
</tr>
</tbody>
</table>

IP Spoofing

Forging IP address of packets
- Spoofer must bypass TCP/IP stack by writing data directly to data link layer (raw sockets)

Attacks
- Conceal identity of attacker
- Misidentification: finger another IP as attacker
- Feints: hide real attack within flood of forged packets
- Authentication: bypass IP-based ACLs
- Denial of Service

Types of Spoofing
- Non-Blind
- Blind

Non-blind Spoofing

Spoofing on a network which you can sniff
- local network
- compromised network
- redirected traffic via ARP spoofing

Easier to attack
- Can see responses
- Can see TCP sequence numbers
Blind Attacks

Attacker A sends packets to victim host V using spoofed IP address of trusted host T

- V will send responses to T
- T will discard responses as replies to packets that it never send
- A cannot see any of the reply packets
- A must be able to ignore or predict responses

Source Routing

- Use source route option to ensure that attacker host A receives responses instead of trusted host T, whose IP address was spoofed
- Well-known attack type
  - Most routers drop source routed packets

IP Spoofing Defences

Packet filtering gateway
- Disallow incoming packets with source IPs that belong to your internal networks
- Drop source routed packets

ISP packet filtering
- Disallow outgoing packets with source IPs that don't belong to ISP
- Drop source routed packets
### TCP 3-way Handshake

![TCP 3-way Handshake Diagram]

### TCP Spoofing

1. Select trusted host to impersonate
2. Guess TCP ISN of victim host
3. Use a DOS attack to silence trusted host
4. Send SYN packet to victim host with spoofed IP address of trusted host
5. Trusted host can't respond to SYN+ACK
6. Send ACK packet to victim host with spoofed IP address and guessed ISN+1

### TCP Session Number Guessing

- Create test TCP connections to target host to examine ISNs and discover algorithms
- Typical algorithm
  - Increment TCP SN by 128,000/sec
  - Increment TCP SN by 64,000/connection
- Calculate round trip time of packets to host
  - Time to host is typically RTT/2
- Send TCP segment with calculated SN
Possible Results

1. Correct Guess
2. Too Low
   - Segment dropped silently
3. Too High, but within window
   - Segment held, pending arrival of intermediate segments
4. Too High, but outside window
   - Segment dropped
   - Host sends a segment back (to spoofed IP) with expected SN

TCP Session Killing

RST
- Need one valid TCP sequence number
- Send RST segment with spoofed IP address and valid sequence number
- May need to send multiple RST's in case host receives TCP segment with your chosen sequence number before your RST segment

FIN
- Need valid TCP sequence + ACK numbers
- Send FIN+ACK segment with spoofed IP address to terminate session
- Receive FIN packet in response, verifying kill if successful

Desynchronized TCP State

- TCP connection in established state
- No data is being sent
- Server SN != Client ACK
- Client SN != Server ACK
- Once data is sent:
  - If Client SN within server window, packet accepted for future use, pending receipt of packet with correct SN
  - If Client SN not within window, discarded
Early desynchronization

1. Listen for SYN+ACK from server in setup
2. Send server RST packet, then SYN packet with exactly same parameters (but diff sequence number) of client SYN
3. Server will close first connection on RST
4. Server will re-open new one on same port with different seq number on receipt of SYN
5. Attacker sends expected ACK response, completing session establishment

Null data desynchronization

1. Attacker sends large amount of null data (data that will not affect session, i.e. telnet NOP) to server, increasing server's ACK number to be out of sequence with client
2. Attacker sends null data to client, forcing it out of sequence with the server

TCP Session Hijacking

1. Guess TCP sequence numbers used in current session between two hosts
2. Create desynchronized state so neither side of connection can talk to the other
3. Send packet with correct SN + ACK with spoofed client IP address to server, containing attack
Session Hijacking Attack

- rlogin can be configured to allow access from an IP address without password
  - ~/.rhosts or /etc/hosts.equiv
- Plan of Attack
  - Hijack telnet connection from V to T
  - Send target host T commands
    - echo "+ + >>~/.rhosts"
  - Use rlogin to access account without password

ACK Storm

- Noisy side effect of TCP session hijacking
- Both client and server ACK unacceptable packets with expected sequence number
- Each ACK is also unacceptable and generates another ACK response
- If network drops packet, no response made
- ACK storms create network congestion, leading to many dropped packets

TCP Defences

- Random ISNs
  - If attacker can't guess sequence numbers of a connection, session can't be hijacked
  - Adding a random number to previous ISN insufficient
  - Some "random" schemes can be statistically attacked
- Cryptographically Secure Protocols
  - Connections reject packets that aren't correctly encrypted as part of the application stream
  - Still vulnerable to RST sniping
UDP Attacks: NIS

- Network Information Service (NIS) used by clients to obtain authentication information, including users, hosts, and netgroups (ACLs) from server
  - RPC service using UDP packets
  - Attacker host listens on client subnet
    - Must respond to requests before real NIS server
  - Attacker attempts to login to client w/ fake user
    - Client asks for fake user’s information from NIS
    - Attacker host responds with a forged password entry
    - Attacker successfully logs in with forged account

Packet Fragmentation

- Occurs at IP layer
- Each fragment has own IP header
- Characteristics:
  - Each fragment of a packet has same identification field
  - More Fragments flag set (except on final frag)
  - Fragment Offset is offset (8-byte units) of fragment from beginning of original datagram
  - Total Length field is length of fragment

Fragment Security Issues

- Large Datagrams
  - Use multiple fragments that will be re-assembled into a packet larger than the maximum IP packet size of 64KB
  - example: ping of death
- Tiny Fragments
  - Artificially small fragments break up TCP header into multiple packets, preventing firewalls/NIDS from seeing header data
  - Minimum fragment size is 68 bytes, which would put only 8 bytes of TCP header (src + dest ports) in first fragment, while flags like SYN and ACK, which indicate connection initiation would be in second
Fragment Security Issues

- Overlapping Fragments
  - Fragment offsets overlap, so during reassembly, second packet is copied over part of TCP header, allowing true header to be hidden in second packet while firewall reads misleading header data from first packet
  - Denial of Service: Teardrop attack uses overlapping fragments to overflow integer in memory copy to crash Windows 95/NT and Linux <2.0.32 hosts
- Tools
  - fragroute, hping

Denial of Service

- Modes of Attack
  - SYN Floods
  - Smurfing
  - Distributed DOS Attacks

Modes of Attack

- Network Connectivity
  - SYN Floods
- Using Your Own Resources Against You
  - echo/chargen spoofing
- Bandwidth Consumption
  - Smurfing
- Other Resource Consumption
  - email bombs
  - disk filling by syslog spoofing/anonymous ftp
**SYN Floods**

- Create many half-open connections to target
  - Send SYN packet
  - Ignore SYN+ACK response
  - (May spoof invalid source IP address for each SYN)
- Target hosts connection table fills up
  - 3 minute timeout for final ACK
  - All new TCP connections refused
- Detection
  - `netstat -a -f inet`
  - Are too many connections in SYN_RECEIVED state?

**SYN Flood Defences**

- Micro-connections: Allocate few resources (~16 bytes) micro-record until ACK received
- RST Cookies:
  - Server sends incorrect SYN+ACK to first client connection request, eliciting RST as response. Thereafter, connections from that client are accepted.
- SYN Cookies: Store state in ISN, not on server.
  - Compute ISN using hash of src + dst IP addresses and ports
  - Valid clients will respond with ISN+1, allowing server to compute connection table entry

**Smurfing**

- Build special ICMP/UDP echo packet
  - Forge IP source address to be that of target
  - Destination address is a broadcast address
- Each host that receives broadcast will respond to the spoofed target address with an echo packet, overwhelming target host
- Most current routers refuse to pass on directed broadcast packets
Distributed DOS Attacks

- Set up DDOS Network
  - Manual compromise by group of crackers
  - Automated compromise by a worm
- Launch Attack
- Victim networks become unresponsive
  - Identification difficult due to router/host failures and lack of logging of packets
- Third party effects
  - Victim responses sent to spoofed IP addresses

DDOS Attack Diagram

Level of Automation

- Manual
  - Manual compromise and attack
- Semi-automated
  - Use of automated exploit to compromise hosts, then DDoS agent (Stacheldraht, TFN2k, Trinoo)
- Automated
  - Worm such as W32/Blaster
Zombie Machines

- Accept commands from master server
  - attack target
  - software updates
- Timer for many worms
- Semi-automatic often use IRC bot
  - IRC bots listen for commands on IRC channel
  - Detect: `netstat -a -n | grep 6667`
- Others use web server or unique UDP server

Types of Attacks

- SYN Flood
- Smurfing
- Bandwidth consumption
  - Many/large packets
  - ICMP flood
  - UDP flood
  - Forge source addresses

Filterable and non-Filterable Attacks

- Filterable Attacks
  - Attack non-essential services (ICMP echo) or ports (random UDP flood)
- Non-filterable Attacks
  - Attack essential services (email or web server ports)
  - Packets may be partially valid for targeted protocol
### Distributed Reflection DOS

- Combine SYN Flood + DDOS attacks
  - Forge target’s IP address in SYN flood on multiple reflection servers
  - Amplification: most OSes send multiple SYN+ACK responses to SYN packet
  - Concealment: packets come from multiple reflection servers, not actual attacker host
  - Difficult to block: attacker may rotate attacks from large pool of reflection servers, many of which may be important hosts which you need to receive traffic from

### DDOS Defences

- Detection
  - DDOSping
  - Zombie Zapper
- Prevention
  - Check for zombie hosts on your networks
  - TCP/IP configuration against specific DDOS attacks like smurfing SYN floods
  - Rate limiting/filtering at border routers or ISP

### IPv6 Security: IPsec

**Encapsulating Security Payload (ESP)**
- End-to-end secret key encryption
- Integrity and data origin authentication
- Anti-replay features
- Confidentiality (padding, dummy packets)

**Authentication Header (AH)**
- ESP features – confidentiality

**IPcomp: IP packet compression**

**IKE (Internet Key Exchange) Protocol**
- Optional: can manually config AH/ESP keys
Port Scanning

Method of discovering exploitable communication channels by probing networked hosts to find which TCP and UDP ports they’re listening on.

1. Port Scanning
2. Stealth Scanning
3. Version Identification
4. OS Fingerprinting
5. Vulnerability Scanning

nmap TCP connect() scan

> nmap -sT at204m02
<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/tcp</td>
<td>open ssh</td>
</tr>
<tr>
<td>80/tcp</td>
<td>open http</td>
</tr>
<tr>
<td>111/tcp</td>
<td>open rpcbind</td>
</tr>
<tr>
<td>443/tcp</td>
<td>open https</td>
</tr>
<tr>
<td>515/tcp</td>
<td>open printer</td>
</tr>
<tr>
<td>2049/tcp</td>
<td>open nfs</td>
</tr>
<tr>
<td>4049/tcp</td>
<td>open lockd</td>
</tr>
<tr>
<td>5432/tcp</td>
<td>open postgres</td>
</tr>
<tr>
<td>5901/tcp</td>
<td>open vnc-1</td>
</tr>
<tr>
<td>6000/tcp</td>
<td>open X11</td>
</tr>
<tr>
<td>32775/tcp</td>
<td>open sometimes-vpnc</td>
</tr>
</tbody>
</table>

Nmap run completed -- 1 IP address (1 host up) scanned in 43.846 seconds

Scanning Techniques

1. TCP connect() scan
2. TCP SYN scan
3. TCP FIN scan
4. TCP Xmas scan
5. TCP Null scan
6. TCP ACK scan
7. Fragmentation Scan
8. FTP bounce scan
9. Idle Scan
10. UDP scan
TCP connect() scan

- Use `connect()` system call on each port, following normal TCP connection protocol (3-way handshake).
- `connect()` will succeed if port is listening.
- Advantages: fast, requires no privileges
- Disadvantages: easily detectable and blockable.

TCP SYN Scan

Send SYN packet and wait for response
- SYN+ACK
  - Port is open
  - Send RST to tear down connection
- RST
  - Port is closed

Advantage: less likely to be logged or blocked
Disadvantage: requires root privilege

TCP FIN scan

- Send TCP FIN packet and wait for response
  - No response
    - Port is open
  - RST
    - Port is closed.

Advantages: more stealthy than SYN scan
Disadvantages: MS Windows doesn't follow standard (RFC 793) and responds with RST in both cases, requires root privilege.
**Xmas and Null Scans**

- Similar to FIN scan with different flag settings.
- Xmas Scan: Sets FIN, URG, and PUSH flags.
- Null Scan: Turns off all TCP flags.

**TCP ACK Scan**

- Does not identify open ports
- Used to determine firewall type
  - Packet filter (identifies responses by ACK bit)
  - Stateful
- Send TCP ACK packet to specified port
  - RST
    - Port is unfiltered (packet got through)
  - No response or ICMP unreachable
    - Port is filtered

**Fragmentation Scan**

- Modify TCP stealth scan (SYN, FIN, Xmas, NULL) to use tiny fragmented IP datagrams.
- Advantages: increases difficulty of scan detection and blocking.
- Disadvantages: does not work on all Oses, and may crash some firewalls/sniffers.
FTP Bounce Scan

FTP protocol supports proxy ftp
- Client requests server send file to another IP, port.
- If server can open connection, port is open.

Advantages:
- Hide identity of scanning host.
- Bypass firewalls by using ftp server behind firewall.

Disadvantages:
- Most ftp servers no longer support proxying.
- Printer ftp servers often do still support.

Idle Scan

Use intermediate idle host to do scan.
- Idle host must increment IP ID for each packet.
- Idle host must not receive traffic from anyone other than attacker.

Scan Process
1. Attacker connects to idle host to obtain initial IP ID X.
2. Send SYN packet to port Y of target with spoofed IP of idle host.
3. If port is open, target host will send SYN+ACK to idle host.
4. Idle host with send RST packet with IP ID X+1 to target.
5. Attacker connects with SYN to idle host to obtain updated IP ID.
6. Idle host sends back SYN+ACK to attacker:
   - Note that this action will increment IP ID by 1.
   - If IP ID is X+2, then port Y on target is open.

Advantages: hides attacker IP address from target.

UDP Scans

Send 0-byte UDP packet to each UDP port
- UDP packet returned:
  - Port is open
  - ICMP port unreachable
- Nothing:
  - Port listed as open|filtered
  - Could be that packet was lost.
  - Could be that server only returns UDP on valid input.

Disadvantages:
- ICMP error rate throttled to a few packets/second (RFC 1812), making UDP scans of all 65535 ports very slow.
- MS Windows doesn't implement rate limiting.
Version Scanning

- Port scanning reveals which ports are open
- Guess services on well-known ports.
- How can we do better?
  - Find what server: vendor and version
  - telnet/ncat to port and check for banner
- Version scanning

Banner Checking

$ nc www.nku.edu 80
GET / HTTP/1.1
HTTP/1.1 400 Bad Request
Date: Sun, 07 Oct 2007 19:27:08 GMT
Server: Apache/1.3.34 (Unix) mod_perl/1.29 PHP/4.3.1 mod_ssl/2.8.25 OpenSSL/0.9.7a
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
  <head>
    <title>400 Bad Request</title>
  </head>
  <body>
    <h1>Bad Request</h1>
    Your browser sent a request that this server could not understand.
    client sent HTTP/1.1 request without hostname (see RFC2616 section 14.23): /
  </body>
</html>

Version Scanning

1. If port is TCP, open connection.
2. Wait for service to identify self with banner.
3. If no identification or port is UDP,
   1. Send probe string based on well-known service.
   2. Check response against db of known results.
4. If no match, test all probe strings in list.
### nmap version scan

```
> nmap -sV at204m02
(The 1645 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE   VERSION
22/tcp       open  ssh       OpenSSH 3.7.1p2 (protocol 1.99)
80/tcp       open  http      Apache httpd 2.0.48 (mod_python/3.1.3 ...
DAV/2)
111/tcp     open  rpcbind   2-4 (rpc #100000)
443/tcp     open  ssl/http  Apache httpd 2.0.48 (mod_python/3.1.3 ...
DAV/2)
515/tcp     open  printer?
2049/tcp   open  nfs       2-3 (rpc #100003)
4045/tcp   open  nlockmgr 1-4 (rpc #100021)
5432/tcp   open  postgres?
5901/tcp   open  vnc       VNC (protocol 3-3)
6000/tcp   open  X11? 32775/tcp open  status   1 (rpc #100024)
```

### More nmap Tools

**Set source port**
- Bypass firewall by using allowed source port.
- Use port 80 for TCP, port 53 for UDP scans.

**Decoys**
- Send additional scans from list of decoys.
- Spoof IP addresses of decoy hosts.
- Defender has to investigate decoys + attacker.

### Defences

**Prevention**
- Disable unnecessary services.
- Block ports at firewall.
- Use a stateful firewall instead of packet filter.

**Detection**
- Network Intrusion Detection Systems.
- Port scans often have distinct signatures.
- IPS can react to scan by blocking IP address.
OS Fingerprinting

Identify OS by specific features of its TCP/IP network stack implementation.

- Explore TCP/IP differences between OSes.
- Build database of OS TCP/IP fingerprints.
- Send set of specially tailored packets to host
- Match results to identical fingerprint in db to identify operating system type and version.

nmap OS fingerprint examples

> nmap -O at204m02
... Device type: general purpose
Running: Sun Solaris 8
OS details: Sun Solaris 8
Uptime 10.035 days (since Sat Mar 27 08:59:38 2004)

> nmap -O 10.17.0.1
... Device type: router
Running: Bay Networks embedded
OS details: Bay Networks BLN-2 Network Router or ASN Processor revision 9

OS Fingerprinting Techniques

FIN probe
- RFC 793 requires no response
- MS Windows, BSDI, Cisco IOS send RST

Bogus flag probe
- Bit 7 of TCP flags unused
- Linux <2.0.35 keeps flag set in response

TCP ISN sampling
- Different algorithms for TCP ISNs

IP Identification
- Different algorithms for incrementing IPID
Passive Fingerprinting
- Identify OSes of hosts on network by sniffing packets sent by each host.
- Use similar characteristics as active technique:
  - TTL
  - MSS
  - Initial Window Size
  - Don't Fragment bit
- Tools: p0f

Fingerprinting Defences
Detection
- NIDS
Blocking
- Firewalling
- Some probes can't be blocked.
Deception
- IP personality changes Linux TCP/IP stack signature to that of another OS in nmap db.

Vulnerability Scanning
Scan for vulnerabilities in systems
- Configuration errors
- Well-known system vulnerabilities
Scanning Tools
- Nessus
- Attack Tool Kit
- GFI LANguard Network Security Scanner
- ISS Internet Scanner
Vulnerability Scanner Architecture

User Interface ➔ Scanning Engine ➔ Scan Results ➔ Report Generation

Vulnerability Database

Nessus Report

Scanning Tools Summary

<table>
<thead>
<tr>
<th>Information</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP addresses of hosts</td>
<td>ping, nmap -sIP</td>
</tr>
<tr>
<td>Network topology</td>
<td>traceroute</td>
</tr>
<tr>
<td>Open ports</td>
<td>nmap port scans</td>
</tr>
<tr>
<td>Service versions</td>
<td>nmap -sV</td>
</tr>
<tr>
<td>OS</td>
<td>nmap -O, xprobe</td>
</tr>
<tr>
<td>Vulnerabilities</td>
<td>Nessus, ATK</td>
</tr>
</tbody>
</table>
What is a Firewall?

- A software or hardware component that restricts network communication between two computers or networks
- In buildings, a firewall is a fireproof wall that restricts the spread of a fire
- Network firewall prevents threats from spreading from one network to another

Internet Firewalls

Many organizations/individuals deploy a firewall to restrict access to their network from Internet
Types of Firewalls

Packet Filters
- Access control based on layer 2+3 (IP + TCP/UDP) headers, such as source and destination address and port.

Circuit-level Gateways
- TCP (layer 3) gateway
- Relay computer copies byte stream from client to server and vice versa.

Application Gateways
- Application protocol (layer 4) gateway

Distributed Firewall
- Central administration of host-based firewalls

Packet Filtering

Forward or drop packets based on TCP/IP header information, most often:
- IP source and destination addresses
- Protocol (ICMP, TCP, or UDP)
- TCP/UDP source and destination ports
- TCP Flags, especially SYN and ACK
- ICMP message type

Dual-homed hosts also make decisions based on:
- Network interface the packet arrived on
- Network interface the packet will depart on

Where to Packet Filter?

Gateway Router
- Filtering at interface between networks allows control via a choke point
- Can filter spoofed IP addresses

Host
- Filter packets on each individual computer
- How to manage thousands of packet filters?
Ingress/Egress Filtering

- Block spoofed IP addresses
- Ingress Filtering
  - Drop packets arriving on external interface whose source IP addresses claims to be from internal network
- Egress Filtering
  - Drop packets arriving on internal interface whose source IP address is not from internal network

Creating a Packet Filter

1. Create a security policy for a service.
   ex: allow only outgoing telnet service
2. Specify security policy in terms of which types of packets are allowed/forbidden
3. Write packet filter in terms of vendor's filtering language

Example: outgoing telnet

TCP-based service
Outbound packets
- Destination port is 23
- Source port is random port >1023
- Outgoing connection established by first packet with no ACK flag set
- Following packets will have ACK flag set
Incoming packets
- Source port is 23, as server runs on port 23
- Destination port is high port used for outbound packets
- All incoming packets will have ACK flag set
Example: outgoing telnet

- First rule allows outgoing telnet packets
- Second rule allows response packets back in
- Third rule denies all else, following Principle of Fail-Safe Defaults

<table>
<thead>
<tr>
<th>Dir</th>
<th>Src</th>
<th>Dest</th>
<th>Proto</th>
<th>S.Port</th>
<th>D.Port</th>
<th>ACK?</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out</td>
<td>Int</td>
<td>Any</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>23</td>
<td>Either</td>
<td>Accept</td>
</tr>
<tr>
<td>In</td>
<td>Any</td>
<td>Int</td>
<td>TCP</td>
<td>23</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Accept</td>
</tr>
<tr>
<td>Either</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Either</td>
<td></td>
<td>Deny</td>
</tr>
</tbody>
</table>

Example: outgoing telnet

Red Hat Linux /etc/sysconfig/iptables
- A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 23 -j ACCEPT
- A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -m tcp --d tcp --sport 23 -j ACCEPT
- A RH-Firewall-1-INPUT -j REJECT

Limitations/Problems

- Must know details of TCP/UDP port usage of protocol to create filters
- Applications only identified by port number
  - What if external host is running a different TCP protocol on port 23?
- Order of rules important
  - Difficulties when adding a new service filter to an existing ruleset
Example: SMTP

Policy: Allow incoming and outgoing SMTP, deny all other services

<table>
<thead>
<tr>
<th>Dir</th>
<th>Src</th>
<th>Dest</th>
<th>Proto</th>
<th>S.Port</th>
<th>D.Port</th>
<th>ACK?</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>In</td>
<td>Ext</td>
<td>Int</td>
<td>TCP</td>
<td>Any</td>
<td>25</td>
<td>Either</td>
<td>Accept</td>
</tr>
<tr>
<td>Out</td>
<td>Int</td>
<td>Ext</td>
<td>TCP</td>
<td>Any</td>
<td>&gt;1023</td>
<td>Either</td>
<td>Accept</td>
</tr>
<tr>
<td>Out</td>
<td>Int</td>
<td>Ext</td>
<td>TCP</td>
<td>Any</td>
<td>25</td>
<td>Either</td>
<td>Accept</td>
</tr>
<tr>
<td>In</td>
<td>Ext</td>
<td>Int</td>
<td>TCP</td>
<td>Any</td>
<td>&gt;1023</td>
<td>Either</td>
<td>Accept</td>
</tr>
<tr>
<td>Either</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Either</td>
<td>Deny</td>
<td></td>
</tr>
</tbody>
</table>

Solution: Revise rules to consider source port and ACK flag

<table>
<thead>
<tr>
<th>Dir</th>
<th>Src</th>
<th>Dest</th>
<th>Proto</th>
<th>S.Port</th>
<th>D.Port</th>
<th>ACK?</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>In</td>
<td>Ext</td>
<td>Int</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>25</td>
<td>Either</td>
<td>Accept</td>
</tr>
<tr>
<td>Out</td>
<td>Int</td>
<td>Ext</td>
<td>TCP</td>
<td>25</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Accept</td>
</tr>
<tr>
<td>Out</td>
<td>Int</td>
<td>Ext</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>25</td>
<td>Either</td>
<td>Accept</td>
</tr>
<tr>
<td>In</td>
<td>Ext</td>
<td>Int</td>
<td>TCP</td>
<td>25</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Accept</td>
</tr>
<tr>
<td>Either</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Either</td>
<td>Deny</td>
<td></td>
</tr>
</tbody>
</table>

Example: SMTP

- Rules 1-2 allow outgoing SMTP
- Rules 3-4 allow incoming SMTP
- Rule 5 denies all other protocols
- Problem:
  - What about external user attacking an internal X server on port 23?
  - Rules 2 + 4 allows all connections where both ends use ports >1023

Solution: Revise rules to consider source port and ACK flag
Stateful Packet Filters

- Saves packet data to keep state, in order to reconstruct connection at IP level
  - Even though UDP has no ACK flag, can construct connection by remembering outgoing packet for UDP 53 (DNS) and know that a response should come from that port to the source port of original packet
- Can examine packets at application layer
  - Examine FTP packet stream for PASV/PORT commands to find return port for ftp data stream

Packet Filtering Summary

Advantages:
- One packet filter can protect an entire network
- Efficient (requires little CPU)
- Supported by most routers

Disadvantages:
- Difficult to configure correctly
  - Must consider rule set in its entirety
- Difficult to test completely
- Performance penalty for complex rulesets
- Stateful packet filtering much more expensive
- Enforces ACLs at layer 2 + 3, without knowing any application details

Circuit Gateways

Proxy host relays TCP/UDP connections
- Client makes connection to proxy
- Proxy forwards connection to server

Proxy provides:
- Access Control
  - Proxies specified source + dest ports / IP addresses
- Logging
- Anonymity
Circuit Gateways

Advantages:
- User-level authentication possible
- Efficient logging, as proxy deals with circuit connections instead of individual packets

Disadvantages:
- Clients have to be recompiled or reconfigured to use proxy service
- Some services can’t be proxied
- Cannot protect you from all protocol weaknesses

Application Gateways

Proxy for a specific application
- HTTP is most common
- SMTP is effectively proxied by default

Advantages
- Filtering based on specifics of application protocol

Disadvantages
- Applications are very complex (SMTP header, data, attachments)

Distributed Firewalls

- Each individual host has a firewall
- Policy set by a central management server

Advantages:
- Can protect machines when no choke point available, including mobile laptops
- No single point of failure

Disadvantages:
- Can’t prevent IP spoofing
**Screened Subnet Architecture**

Isolates internal network from external networks by means of a perimeter network, often called a DMZ.

---

**Screened Subnet**

Bastion hosts isolated from internal network

- Compromise of a bastion host doesn't directly compromise internal network
- Bastion hosts also can't sniff internal traffic, since they're on a different subnet

No single point of failure

- Attacker must compromise both exterior and interior routers to gain access to internal net

Advantages: greater security

Disadvantages: higher cost and complexity

---

**Screened Subnet**

External Access

- Filtered: via interior + exterior routers
- Proxied: use a bastion host as a proxy server

Bastion Hosts

- Proxy server
- External web/ftp servers
- External DNS server
- E-mail gateway
Screened Subnet

Exterior Router
- Simple filtering rules
- Ingress/egress filtering
- DOS prevention
- Simple ACLs
- May be controlled by ISP

Interior Router
- Complex filtering rules
- Must protect internal network from bastion hosts as well as external network

Recommendation: use different hardware/software for interior and exterior routers

Tunneling

Tunneling: Encapsulation of one network protocol in another protocol
- Carrier Protocol: protocol used by network through which the information is travelling
- Encapsulating Protocol: protocol (GRE, IPsec, L2TP) that is wrapped around original data
- Passenger Protocol: protocol that carries original data

ssh Tunneling

SSH can tunnel TCP connections
- Carrier Protocol: IP
- Encapsulating Protocol: ssh
- Passenger Protocol: TCP on a specific port

POP-3 forwarding
```plaintext
ssh -L 110:pop3host:110 -1 user pop3host
```
- Uses ssh to login to pop3host as user
- Creates tunnel from port 110 (leftmost port #) on localhost to port 110 (rightmost port #) of pop3host
- User configures mail client to use localhost as POP3 server, then proceeds as normal
Virtual Private Network (VPN)
- Two or more computers or networks connected by a private tunnel through a public network (typically the Internet)
- Requirements:
  - Confidentiality: encryption
  - Integrity: MACs, sequencing, timestamps
- Firewall Interactions
  - Tunnels can bypass firewall
  - Firewall is convenient place to add VPN features

Firewall Limitations
Cannot protect from internal attacks
- May be able to limit access with internal firewalls to a segment of your network
Cannot protect you from user error
- Users will still run trojan horses that make it past your AV scanner
Firewall mechanism may not precisely enforce your security policy

Key Points
1. TCP/IP insecure layered architecture
   1. IP addresses can be spoofed
   2. TCP sessions can be hijacked
2. Denial of service attacks
   1. Technical attacks exploit an implementation flaw
   2. Brute force attacks saturate network bandwidth
3. Port scanning allows attackers to find targets
   1. Stealth scans can avoid firewalls or NIDS
4. Firewalls block some classes of attacks
   1. Can block packets by IP, port, or TCP flags
   2. Packet filters vs. stateful firewalls
References