Network Threats/Attacks
Internet Structure

- TCP/IP for packet routing and connections
- Border Gateway Protocol (BGP) for route discovery
- Domain Name System (DNS) for IP address discovery
OSI Protocol Stack

application
presentation
session
transport
network
data link
physical

email, Web, NFS
RPC
TCP
IP
Ethernet
Data Formats

application layer

transport layer

network layer

data link layer

message

segment

packet

frame

IPv6 only

(IPv4 may fragment)
TCP (Transmission Control Protocol)

◆ Sender: break data into segments
  • Sequence number is attached to every packet
◆ Receiver: reassemble segments
  • Acknowledge receipt; lost packets are re-sent
◆ Connection state maintained by both sides
IP (Internet Protocol)

◆ Connectionless
  - Unreliable, “best-effort” protocol

◆ Uses addresses (and prefixes thereof) used for routing
  - Longest-prefix match
  - Typically several hops in route

<table>
<thead>
<tr>
<th>IP Packet</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>128.83.130.239</td>
</tr>
<tr>
<td>Dest</td>
<td>171.64.66.201</td>
</tr>
<tr>
<td>Seq #</td>
<td>33040</td>
</tr>
</tbody>
</table>
ICMP (Control Message Protocol)

◆ Provides feedback about network operation
  • Out-of-band (control) messages carried in IP packets
  • Error reporting, congestion control, reachability, etc.

◆ Example messages:
  • Destination unreachable
  • Time exceeded
  • Parameter problem
  • Redirect to better gateway
  • Reachability test (echo / echo reply)
  • Timestamp request / reply
Security Issues in TCP/IP

- Network packets pass by and thru untrusted hosts
  - Eavesdropping (packet sniffing)
- IP addresses are public
  - Smurf attacks
- TCP connection requires state
  - SYN flooding
- TCP state easy to guess
  - TCP spoofing and connection hijacking
Packet Sniffing

- Many old applications send data unencrypted
  - Plain ftp, telnet send passwords in the clear (as opposed to sftp and ssh)
- Network Interface Card (NIC), such as ethernet adaptor, in “promiscuous mode” can read all data on its broadcast segment

Solution: encryption (e.g., IPsec), improved routing
“Smurf” Attack

1 ICMP Echo Req
Src: victim’s address
Dest: broadcast address

Looks like a legitimate “Are you alive?” ping request from the victim

Stream of ping replies overwhelms victim

Every host on the network generates a ping reply (ICMP Echo Reply) to victim

Solution: reject external packets to broadcast addresses
“Ping of Death”

- When old Windows machine receives an ICMP packet with payload over 64K, machine crashes and/or reboots

  - Programming error in older versions of Windows
  - Packets of this length are illegal, so programmers of old Windows code did not account for them

Solution: patch OS, filter out ICMP packets
“Teardrop” and “Bonk”

◆ TCP fragments contain Offset field
◆ Attacker sets Offset field to:
  • overlapping values
    – Bad/old implementation of TCP/IP stack crashes when attempting to re-assemble the fragments
  • ... or to very large values
    – Target system crashes

Solution: use up-to-date TCP/IP implementation
“LAND”

- Single-packet denial of service (DoS) attack
- IP packet with \(<\text{source-address, port}>\) equal to \(<\text{destination-address, port}>\), SYN flag set
- Triggers loopback in Windows XP SP2 implementation of TCP/IP stack
  - Locks up CPU

Solution: ingress filtering???
TCP Handshake Reminder

- **C**
  - SYN
  - SYN, ACK
  - ACK

- **S**
  - Listening...
  - Spawn thread, store data (connection state, etc.)
  - Wait
  - Connected
SYN Flooding Attack

Listening...

Spawn a new thread, store connection data

... and more

... and more

... and more

... and more
SYN Flooding Explained

◆ Attacker sends many connection requests (SYNs) with spoofed source addresses

◆ Victim allocates resources for each request
  • New thread, connection state maintained until timeout
  • Fixed bound on half-open connections

◆ Once resources exhausted, requests from legitimate clients are denied

◆ This is a classic DoS attack example
  • Common pattern: it costs nothing to TCP client to send a connection request, but TCP server must spawn a thread for each request - asymmetry!
  • Other examples of this behavior?
Preventing Denial of Service

- DoS is caused by asymmetric state allocation
  - If server opens new state for each connection attempt, attacker can initiate many connections from bogus or forged IP addresses
- Cookies allow server to remain stateless until client produces:
  - Server state (IP addresses and ports) stored in a cookie and originally sent to client
- When client responds, cookie is verified
SYN Cookies

Compatible with standard TCP; simply a “weird” sequence number scheme

$F(AES$ or a truncated hash$)$

$F$(source addr, source port, dest addr, dest port, coarse time, server secret)

Sequence # = cookie

$S\rightarrow C$: SYN$_S$, ACK$_C$

$C\rightarrow S$: ACK$_S$(cookie)

Does not store state

- Cookie must be fresh, and unforgeable
- Client should not be able to invert a cookie (why?)

Recompute cookie, compare with with the one received, only establish connection if they match

More info: http://cr.yp.to/syncookies.html
Anti-Spoofing Cookies: Basic Pattern

◆ Client sends request (message #1) to server
◆ Typical protocol:
  • Server sets up connection, responds with message #2
  • Client may complete session or not (potential DoS)
◆ Cookie version:
  • Server responds with hashed connection data instead of message #2
    – Does not spawn any threads, does not allocate resources!
  • Client confirms by returning cookie (with other fields)
    – If source IP address is bogus, attacker can’t confirm
    – WHY?
Passive Defense: Random Deletion

- If SYN queue is full, delete random entry
  - Legitimate connections have a chance to complete
  - Fake addresses will be eventually deleted. WHY?
- Easy to implement

SYN_C

<table>
<thead>
<tr>
<th>half-open connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>121.17.182.45</td>
</tr>
<tr>
<td>231.202.1.16</td>
</tr>
<tr>
<td>121.100.20.14</td>
</tr>
<tr>
<td>5.17.95.155</td>
</tr>
</tbody>
</table>
TCP Connection Spoofing

◆ Each TCP connection has associated state
  • Sequence number, port number
◆ TCP state is easy to guess
  • Port numbers standard, seq numbers predictable
◆ Can inject packets into existing connections
  • If attacker knows initial sequence number and amount of traffic, can guess current number
  • Guessing a 32-bit seq number is not practical, BUT...
  • Most systems accept a large window of sequence numbers (to handle massive packet losses)
  • Send a flood of packets with likely sequence numbers
“Blind” IP Spoofing Attack

Can’t receive packets sent to Bob, but maybe can penetrate Alice’s computer if Alice uses IP address-based “authentication”

- rlogin and other (insecure) remote access tools use address-based authentication
DoS by Connection Reset

- If attacker can guess the current sequence number for an existing connection, can send Reset packet to close it
- Especially effective against long-lived connections
  - For example, BGP route updates
    - Adjacent BGP routers keep long-lived TCP connections
User Datagram Protocol (UDP)

UDP – alternative to TCP, connectionless protocol
- Simply sends datagram to application process at the specified port of the IP address
- Source port number provides return address
- Applications: media streaming, broadcast

No acknowledgements, no flow control, no message continuation

So…. Denial of Service by UDP data flood
Countermeasures

◆ Above transport layer: Kerberos
  • Provides authentication, protects against application-layer spoofing
  • Does not protect against connection hijacking

◆ Above network layer: SSL/TLS and SSH
  • Protects against connection hijacking and injected data
  • Does not protect against DoS by spoofed packets

◆ Network (IP) layer: IPsec
  • Protects against hijacking, injection, DoS using connection resets, IP address spoofing
  • But muddled/poor key management...

◆ Below network layer?
IP Routing

Routing of IP packets is based on IP addresses
- 32-bit host identifiers (128-bit in IPv6)

Routers use a forwarding table (FIB)
- Entry = [destination, nxt hop, interface, metric ]
- Table look-up for each IP packet to decide how to route it

Routers learn routes to hosts and networks via routing protocols
- Host identified by its IP address, network – by IP prefix

BGP (Border Gateway Protocol) is the core Internet protocol for establishing inter-AS routes
Distance-Vector Routing

- Each node keeps vector with distances to all nodes
- Periodically sends distance vector to all neighbors
- Neighbors reciprocate; node updates its vector based on received information
  - **Bellman-Ford algorithm**: for each destination, router picks the neighbor advertising the cheapest route, adds his entry into its own routing table and re-advertises
  - Used in RIP (routing information protocol)

**Split-horizon update**
- Do not advertise a route on an interface from which you learned the route in the first place!
Good News Travels Fast

G1 advertises route to network A with distance 1
G2-G5 quickly learn the good news and install the routes to A via G1 in their local routing tables
Bad News Travels Slowly

- G1’s link to A goes down
- G2 is advertising a pretty good route to G1 (cost=2)
- G1’s packets to A are forever looping between G2 and G1
- G1 is now advertising a route to A with cost=3, so G2 updates its own route to A via G1 to have cost=4, and so on
  - G1 and G2 are slowly counting to infinity
  - Split-horizon updates only prevent two-node loops
Overview of BGP

• BGP is a path-vector INTER-AS protocol
• Just like distance-vector, but routing updates, for each entry *also* contain an AS-level path to destination
  • List of traversed AS-s and a set of network prefixes belonging to the first AS on the list
• Each BGP router receives UPDATE messages from neighbors, selects one “best” path for each prefix, and advertises to its neighbors
  • Can be shortest path, but doesn’t have to be
  • AS doesn’t have to use the path it advertises!
AS 2 provides **transit** service for AS 7

- Traffic to and from AS 7 travels through AS 2
Some BGP Statistics

- BGP routing tables contain about 125,000 address prefixes mapping to about 17-18,000 paths
- Approx. 10,000 BGP routers
- Approx. 2,000 organizations are AS-es
- Approx. 6,000 organizations own prefixes
- Average route length (AS hops) is about 3.7
- 50% of routes have length less than 4 AS-s
- 95% of routes have length less than 5 AS-s
BGP Misconfiguration

- Domain advertises good routes to addresses it does not know how to reach
  - Result: packets go into a network “black hole”
- April 25, 1997: “The day the Internet died”
  - AS7007 (Florida Internet Exchange) de-aggregated the BGP route table and re-advertised all prefixes as if it originated paths to them
  - In effect, AS7007 was advertising that it has the best route to every host on the Internet
  - Huge network instability as incorrect routing data propagated and routers crashed under traffic
BGP Security

◆ BGP update messages contain no authentication or integrity protection
  • However, today BGP updates are sent over secure tunnels

◆ Attacker may falsify advertised routes
  • Modify IP prefixes associated with the route
    – Can blackhole traffic to certain IP prefixes
  • Change AS path
    – Either attract traffic to attacker’s AS, or divert traffic away
    – Interesting economic incentive: an ISP wants to dump its traffic on other ISPs without routing their traffic in exchange
  • Re-advertise/propagate AS path without permission
    – For example, multi-homed customer may end up advertising transit capability between two large ISPs
AS36561 (YouTube) advertises 208.65.152.0/22
YouTube (February 24, 2008)

- Pakistan government wants to block YouTube
  - AS17557 (Pakistan Telecom) advertises 208.65.153.0/24
  - All YouTube traffic worldwide directed to AS17557

Result: two-hour YouTube outage
Other BGP Incidents

✦ May 2003: Spammers hijack unused block of IP addresses belonging to Northrop Grumman
  • Entire Northrop Grumman ends up on spam blacklist
  • Took two months to reclaim ownership of IP addresses

✦ May 2004: Malaysian ISP hijacks prefix of Yahoo California data center

✦ Dec 2004: Turkish ISP advertises routes to the entire Internet, including Amazon, CNN, Yahoo
DNS: Domain Name Service

DNS maps symbolic names to numeric IP addresses
(for example, www.uci.edu ↔ 128.195.188.233)
DNS Root Name Servers

- Root name servers for top-level domains
- Authoritative name servers for subdomains
- Local name resolvers contact authoritative servers when they do not know a name

Feb 6, 2007: DoS attack on root DNS servers
DNS Caching

◆ DNS responses are cached
  • Quick response for repeated translations
  • Other queries may reuse some parts of lookup
    – NS records for domains

◆ DNS negative queries are cached
  • Don’t have to repeat past mistakes, e.g., typos

◆ Cached data periodically times out
  • Lifetime (TTL) of data controlled by owner of data
  • TTL passed with every record
Cached Lookup Example
DNS “Authentication”

Request contains random 16-bit transaction id \(\rightarrow\) TXID

Response accepted if TXID is the same
Stays in cache for a long time (TTL)
DNS Spoofing

**6.6.6.6**

Trick client into looking up host1.foo.com (how?)

Guess TXID, host1.foo.com is at 6.6.6.6
Another guess, host1.foo.com is at 6.6.6.6
Another guess, host1.foo.com is at 6.6.6.6

Client

Local resolver

ns.foo.com DNS server

Several opportunities to win the race
If attacker loses, has to wait until TTL expires
... but can try again with host2.foo.com, host3.foo.com, etc.
... but what’s the point of hijacking host3.foo.com?
Exploiting Recursive Resolving

6.6.6.6

Trick client into looking up host1.foo.com

Guessed TXID, very long TTL
I don’t know where host1.foo.com is
Ask the authoritative server at ns2.foo.com
It lives at 6.6.6.6

If attacker wins, all future DNS requests will go to 6.6.6.6
The cache is now poisoned... for a very long time!
No need to win future races!

[Kaminsky]
Triggering DNS Lookup

◆ Any link, any image, any ad, anything can cause a DNS lookup
  • No Javascript required, though it helps
◆ Mail servers will look up what bad guy wants
  • Upon first greeting: HELLO
  • Upon first learning who they’re talking to: MAIL FROM
  • Upon spam check (oops!)
  • When trying to deliver a bounce
  • When trying to deliver a newsletter
Reverse DNS Spoofing

**Trusted access is often based on host names**
- E.g., permit all hosts in .rhosts to run remote shell

**Network requests such as rsh or rlogin arrive from numeric source addresses**
- System performs reverse DNS lookup to determine requester’s host name and checks if it’s in .rhosts

**If attacker can spoof the answer to reverse DNS query, he can fool target machine into thinking that request comes from an authorized host**
- No authentication for DNS responses and typically no double-checking (numeric → symbolic → numeric)
Pharming

◆ Many anti-phishing defenses rely on DNS
◆ Can bypass them by poisoning DNS cache and/or forging DNS responses
  • Browser: give me the address of www.paypal.com
  • Attacker: sure, it’s 6.6.6.6 (attacker-controlled site)
◆ Dynamic pharming
  • Provide bogus DNS mapping for a trusted server, trick user into downloading a malicious script
  • Force user to download content from the real server, temporarily provide correct DNS mapping
  • Malicious script and content have the same origin!
    – So, malicious script can access (sensitive) content
JavaScript/DNS Intranet attack (I)

- Consider a Web server intra.good.net
  - IP: 10.0.0.7, inaccessible outside good.net network
  - Hosts sensitive CGI applications
- Attacker at evil.org gets good.net user to browse www.evil.org (e.g., via a link in an email msg)
- Places JavaScript on www.evil.org that accesses sensitive application on intra.good.net
  - This doesn’t work since JavaScript is subject to the same origin policy -- user’s browser tries to prevent client-side scripts from different places
  - ... but suppose the attacker controls DNS
JavaScript/DNS Intranet attack (II)

- The “same origin” policy is now satisfied!
Other DNS Vulnerabilities

- DNS implementations can also have vulnerabilities
  - Reverse query buffer overrun in old releases of BIND
  - MS DNS for NT 4.0 crashes on chargen stream
- Denial of service
  - Oct 2002: ICMP flood took out 9 root servers for 1 hour
- Can use “zone transfer” requests to download DNS database and map out the network
  - “The Art of Intrusion”: NYTimes.com and Excite@Home
  - Solution: block port 53 on corporate name servers

See http://cr.yp.to/djbdns/notes.html
DNS Vulnerabilities: Summary

Zone file
- Dynamic updates
- Unauthorized updates

Master
- Impersonating master
- Corrupting data

Slaves

Resolver
- Cache pollution by data spoofing
- Cache impersonation

Stub resolver

Zone administrator
Domain Hijacking and Other Risks

◆ Spoofed ICANN registration and domain hijacking
  • Authentication of domain transfers based on email addr
  • Aug ’04: teenager hijacks eBay’s German site
  • Jan ’05: hijacking of panix.com (oldest ISP in NYC)
    – "The ownership of panix.com was moved to a company in Australia, the actual DNS records were moved to a company in the United Kingdom, and Panix.com's mail has been redirected to yet another company in Canada."
  • Many other domain theft attacks

◆ Misconfiguration and human error

ICANN: Internet Corporation for Assigned Names and Numbers
Solving the DNS Spoofing Problem

◆ Long TTL for legitimate responses
  • Does it really help?

◆ Randomize port in addition to TXID
  • 32 bits of randomness – makes it harder for attacker to guess TXID

◆ DNSSEC
  • Cryptographic authentication of host-address mappings
DNSSEC

◆ Goals: authentication and integrity of DNS requests and responses

◆ PK-DNSSEC (public key)
  - DNS server signs its data (can be done in advance)
  - How do other servers learn the public key?

◆ SK-DNSSEC (symmetric key)
  - Encryption and MAC: $E_k(m, MAC(m))$
  - Each message contains a nonce to avoid replay
  - Each DNS node shares a symmetric key with its parent
  - Zone root server has a public key (hybrid approach)

MORE INFO: [http://www.dnssec.net/presentations](http://www.dnssec.net/presentations)