Network security

- First need to review basic networking
  - Network Architecture
  - IP
  - UDP
  - TCP
  - DNS
- And vulnerabilities in their architecture
TCP/IP Protocol Stack

TCP Header Format

- **Ports** plus IP addresses identify a connection

  ![TCP Header Format Diagram]

Connection Setup:
Agree on initial Sequence #'s

- Three-way handshake
Basic TCP/IP Security Issues

- No Authentication/Authorization
  - Anyone can send to any port on any host; port scanning, Denial of Service (DoS), worms
- No Attribution
  - Nothing enforces correctness of IP address; IP spoofing
- Network packets not private
  - Intermediate networks not necessarily trusted; packet sniffing
- TCP/IP state can be easy to guess
  - TCP connection spoofing, blind port scanning

1. Packet Sniffing

- Promiscuous NIC reads all packets
  - Read all unencrypted data (e.g., "wireshark")
  - ftp, telnet (and POP, IMAP) send passwords in clear!

Alice  Network  Bob

Prevention: Encryption

2. TCP Connection Spoofing

- Why random initial sequence numbers?  $(SN_C, SN_S)$
- Suppose init. sequence numbers are predictable
  - Attacker can create TCP session on behalf of forged source IP
    - Breaks IP-based authentication (e.g. SPF, /etc/hosts)

Attacker  
\[
\text{TCP SYN} \\
\text{srcIP=victim} \\
\text{ACK} \\
\text{srcIP=victim} \text{ AN=predicted SN}_S \\
\text{command} \\
\]

Server  
\[
\text{SYN/ACK} \\
\text{dstIP=victim} \text{ SN=server SN} \\
\text{victim} \\
\]

\[
\text{server thinks command is from victim IP addr} \\
\]
**Example DoS vulnerability**

- Suppose attacker can guess seq. number for an existing connection:
  - Attacker can send Reset packet to close connection. Results in DoS.
  - Naively, success prob. is $1/2^{32}$ (32-bit seq. #’s).
  - Most systems allow for a large window of acceptable seq. #’s
    - Much higher success probability.
  - Attack is most effective against long lived connections (expensive to set up again; BGP)

---

**Random initial TCP SNs**

- Unpredictable SNs prevent basic packet injection
  - … but attacker can inject packets after eavesdropping to obtain current SN
- Most TCP stacks now generate random SNs
  - Random generator should be unpredictable
  - GPR’06: Linux RNG for generating SNs is predictable
    - Attacker repeatedly connects to server
    - Obtains sequence of SNs
    - Can predict next SN
    - Attacker can now do TCP spoofing
      (create TCP session with forged source IP)

---

**Blind port scanning**

- Similar issue: identification field
  - Hosts typically increment by one after each packet to ensure id field is unique (recall: for fragmentation)
  - If you receive a pkt from host A at time $t_1$ with id =10, and another packet at time $t_2$ with id=12, you can infer… that host A sent another packet somewhere
**Blind port scanning**

1. SYN
2. RST id=x
3. SYN srcIP=S, dstPort=X
4. RST if port closed or SYMACK if open
5. RST if SYMACK, else nothing
6. SYN
7. Response from S... What is value of id?
   \(x+1\) or \(x+2\)?

**Protocol Vulnerabilities**

- Network protocols are frequently designed assuming all actors are benign
- What if one participant in a communication doesn't obey the protocol?
- Simple example: routing
  - Nothing prevents UCSD from claiming to "own" 128.95/16 (University of Washington)
  - This happens, both on purpose and by accident
- Other examples
  - TCP congestion control
  - DNS poisoning

**How TCP congestion control works**

- Sender maintains "congestion window"
  - Limit on amount of outstanding data
  - Grows when data is successfully delivered
  - Shrinks when data is lost
- Receiver sends ACKs in response to data
  - ACKs tell sender that data has been received
  - Indicate the next data item expected
- Works if everyone plays fair
  - Sender could ignore protocol and send faster
  - What about receiver? (e.g., Web browser)
Sources of vulnerability

- ACKs mean things that they don’t prove
  - I was sent in response to a data packet
  - That data packet has been received
  - I have received all the data up to X-1
  - I have (still) not yet received data X
- Sender assumes things that aren’t necessarily true
  - At most one ACK generated per data packet
  - Every ACK acknowledges a full-sized packet

What’s supposed to happen

Sender | Receiver
--- | ---
Round Trip Time (RTT) | Data 1:1460
| ACK 1461
Data 1461:2920
| ACK 2921:4380
| ACK 4381

- Rule: grow window by one full-sized packet for each valid ACK received
- Congestion window doubles each round trip time

Example of breaking the rules

Sender | Receiver
--- | ---
Round Trip Time (RTT) | Data 1:1461
| ACK 486
| ACK 973
| ACK 1461
Data 1461:2920
Data 2921:4381
Data 4381:5841
Data 5841:7301

- Rule: grow window by one full-sized packet for each valid ACK received
- Send M ACKs for one pkt
- Growth factor proportional to M
Why my Web browser is faster than yours

Page fetch from CNN.com

Sequence Number (bytes)

Time (sec)

Modified Client

Normal Client

DNS

Domain Name System

- Hierarchical Name Space

- Hierarchical service
- 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

DNS Root Name Servers
DNS Lookup Example

DNS record types (partial list):
- **NS**: name server (points to other server)
- **A**: address record (contains IP address)
- **MX**: address in charge of handling email
- **TXT**: generic text (e.g. used to distribute site public keys (DKIM))

Caching

- DNS responses are cached
  - Quick response for repeated translations
  - Useful for finding servers as well as addresses
    - NS records for domains

- DNS negative queries are cached
  - Save time for nonexistent sites, e.g. misspelling

- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record

DNS Packet

- Query ID:
  - 16 bit random value
  - Links response to query

(from Steve Friedl)
Resolver to NS request

- SRC IP = 68.94.150.1
- DST IP = 68.94.33.30
- SRC port = 53
- DST port = 53

UDP packet:
- Record type = NS
- Record class = IN
- Record TTL = 10800
- Record RD Length = 33
- Record Name = a.com
- Record Class = IN
- Record Ttl = 10800
- Record Rdata = 192.168.1.2
- Record Rtype = NS

Response to resolver

- Response contains IP addr of next NS server (called "glue")
- Response ignored if unrecognized QueryID

Authoritative response to resolver

- Bailiwick checking: response is cached if it is within the same domain of query (i.e. a.com cannot set NS for b.com)
- Final answer
Basic DNS Vulnerabilities

- Users/hosts trust the host-address mapping provided by DNS:
  - Used as basis for many security policies:
    - Browser "same origin" policy, URL address bar, user trust

- Obvious problems
  - Interception of requests or compromise of DNS servers can result in incorrect or malicious responses
    - e.g.: hijack network route to spoof DNS
  - Solution – authenticated requests/responses
    - Provided by DNSsec … but no one uses DNSsec

DNS cache poisoning (a la Kaminsky’08)

- Victim machine visits attacker’s web site, downloads Javascript

If at first you don’t succeed …

- Victim machine visits attacker’s web site, downloads Javascript
Defenses

- Increase Query ID size. How? Some proposals
  - Randomize src port, additional 11 bits
    Now attack takes several hours
  - Ask every DNS query twice:
    - Attacker has to guess QueryID correctly twice (32 bits)
    - Not clear DNS system can handle load

Summary

- Core protocols not designed for security
  - Eavesdropping, Packet injection, Route stealing, DNS poisoning
  - Patched over time to prevent basic attacks
    (e.g. random TCP SN)
  - More secure variants exist (limited deployment)
    IP -> IPsec
    DNS -> DNSsec
    BGP -> SBGP
- Still need to be careful about protocol semantics