The Internet

- A bunch of computer networks voluntarily interconnected
- Capitalized because there’s really only one
- No centralized network-level management
  - But technical collaboration, DNS, etc.

Layered model (OSI)

1. Physical (10BASE-T)
2. Data-link (PPP)
3. Network (IP)
4. Transport (TCP)
5. Session (SSL?)
6. Presentation (MIME?)
7. Application (HTTP)

Layered model: TCP/IP

- Application
- Transport
- Network
- Link

Packet wrapping

- application data
- segments
  - TCP data
  - TCP data
  - TCP data
- packets
  - IP
  - TCP
  - data
- frames
  - E1H
  - IP
  - TCP
  - data
  - E1T
**IP(v4) addressing**
- Interfaces (hosts or routers) identified by 32-bit addresses
  - Written as four decimal bytes, e.g. 192.168.10.2
  - First $k$ bits identify network, $32 - k$ host within network
  - Can’t (anymore) tell $k$ from the bits
  - We’ll run out any year now

**IP and ICMP**
- Internet Protocol (IP) forwards individual packets
- Packets have source and destination addresses, other options
- Automatic fragmentation (usually avoided)
- ICMP (I Control Message P) adds errors, ping packets, etc.

**UDP**
- User Datagram Protocol: thin wrapper around IP
- Adds source and destination port numbers (each 16-bit)
- Still connectionless, unreliable
- OK for some small messages

**TCP**
- Transmission Control Protocol: provides reliable bidirectional stream abstraction
- Packets have sequence numbers, acknowledged in order
- Missed packets resent later

**Flow and congestion control**
- Flow control: match speed to slowest link
  - “Window” limits number of packets sent but not ACKed
- Congestion control: avoid traffic jams
  - Lost packets signal congestion
  - Additive increase, multiplicative decrease of rate

**Routing**
- Where do I send this packet next?
  - Table from address ranges to next hops
- Core Internet routers need big tables
- Maintained by complex, insecure, cooperative protocols
  - Internet-level algorithm: BGP (Border Gateway Protocol)
Below IP: ARP

- Address Resolution Protocol maps IP addresses to lower-level address
  - E.g., 48-bit Ethernet MAC address
- Based on local-network broadcast packets
- Complex Ethernets also need their own routing (but called switches)

DNS

- Domain Name System: map more memorable and stable string names to IP addresses
- Hierarchically administered namespace
  - Like Unix paths, but backwards
- .edu server delegates to .umn.edu server, etc.

DNS caching and reverse DNS

- To be practical, DNS requires caching
  - Of positive and negative results
- But, cache lifetime limited for freshness
- Also, reverse IP to name mapping
  - Based on special top-level domain, IP address written backwards

Classic application: remote login

- Killer app of early Internet: access supercomputers at another university
- Telnet: works cross-OS
  - Send character stream, run regular login program
- rlogin: BSD Unix
  - Can authenticate based on trusting computer connection comes from
    (Also rsh, rcp)

Outline

Brief introduction to networking
Announcements intermission
Some classic network attacks
Second half of course
More Unix access control

What about the midterm?

- Will be graded and handed back in class Monday
- Solutions might be posted a bit earlier
Hands-on Assignment 1

Intended possibilities remaining in week 5:
- Format string vulnerability (exploited twice)
- `get_line` buffer overflow (not exploited with new buffer size)
- Negative time since modification?
- In class Monday: discussion of attacks

Project meetings schedule

Next week, same time of week as first meeting
- Unless we arrange otherwise
- Invitation/reminder emails out soon

Outline

Brief introduction to networking
Announcements intermission
Some classic network attacks
Second half of course
More Unix access control

Packet sniffing

- Watch other people’s traffic as it goes by on network
- Easiest on:
  - Old-style broadcast (thin, "hub") Ethernet
  - Wireless
- Or if you own the router

Forging packet sources

- Source IP address not involved in routing, often not checked
- Change it to something else!
- Might already be enough to fool a naive UDP protocol

TCP spoofing

- Forging source address only lets you talk, not listen
- Old attack: wait until connection established, then DoS one participant and send packets in their place
- Frustrated by making TCP initial sequence numbers unpredictable
  - But see Oakland’12, WOOT’12 for fancier attacks, keyword “off-path”
ARP spoofing
- Impersonate other hosts on local network level
- Typical ARP implementations stateless, don't mind changes
- Now you get victim’s traffic, can read, modify, resend

rlogin and reverse DNS
- rlogin uses reverse DNS to see if originating host is on whitelist
- How can you attack this mechanism with an honest source IP address?
- Remember, ownership of reverse-DNS is by IP address

rlogin and reverse DNS
- rlogin uses reverse DNS to see if originating host is on whitelist
- How can you attack this mechanism with an honest source IP address?
- Remember, ownership of reverse-DNS is by IP address

Outline
- Brief introduction to networking
- Announcements intermission
- Some classic network attacks
- Second half of course
- More Unix access control

Cryptographic primitives
- Core mathematical tools
- Symmetric: block cipher, hash function, MAC
- Public-key: encryption, signature
- Some insights on how they work, but concentrating on how to use them correctly

Cryptographic protocols
- Sequence of messages and crypto privileges for, e.g., key exchange
- A lot can go wrong here, too
- Also other ways security can fail even with a good crypto primitive
Crypto in Internet protocols
- How can we use crypto to secure network protocols
  - E.g., rsh → ssh
- Challenges of getting the right public keys
- Fitting into existing usage ecosystems

Web security: server side
- Web software is privileged and processes untrusted data: what could go wrong?
  - Shell script injection (Ex. 1)
  - SQL injection
  - Cross-site scripting (XSS) and related problems

Web security: client side
- JavaScript security environment even more tricky, complex
- More kinds of cross-site scripting
- Possibilities for sandboxing

Security middleboxes
- Firewall: block traffic according to security policy
- NAT box: different original purpose, now de-facto firewall
- IDS (Intrusion Detection System): recognize possible attacks

Malware and network DoS
- Attacks made possible by the network
  - Viruses, trojans, bot nets
    - Detection?
    - Mitigation?
  - Distributed denial of service (DDoS)

Adding back privacy
- Every Internet packet has source and destination addresses on it
- So how can network traffic be private or anonymous?
  - Key technique: overlay a new network
  - Examples: onion routing (Tor), anonymous remailing
Usability of security

- Prevent people from being the weakest link
- Usability of authentication
- “Secure” web sites, phishing
- Making decisions about mobile apps

Electronic voting

- Challenging: hard versions of many hard problems:
  - Trust in software
  - Usability
  - Simultaneously public and private
- Some deployed systems arguably worse than paper
- Can do better with crypto and systems approaches

Electronic money (Bitcoin)

- Current payment systems have strong centralized trust
  - US Federal Reserve and mint
  - Banks, PayPal
- Could they be replaced by a peer-to-peer distributed system?
- Maybe

Outline

- Brief introduction to networking
- Announcements intermission
- Some classic network attacks
- Second half of course
- More Unix access control

“POSIX” ACLs

- Based on a withdrawn standardization
- More flexible permissions, still fairly Unix-like
- Multiple user and group entries
  - Decision still based on one entry
- Default ACLs: generalize group inheritance
- Command line: getfacl, setfacl

ACL legacy interactions

- Hard problem: don’t break security of legacy code
  - Suggests: “fail closed”
- Contrary pressure: don’t want to break functionality
  - Suggests: “fail open”
- POSIX ACL design: old group permission bits are a mask on all novel permissions
**“POSIX” “capabilities”**
- Divide root privilege into smaller (~35) pieces
- Note: not real capabilities
- First runtime only, then added to FS similar to setuid
- Motivating example: ping
- Also allows permanent disabling

**Privilege escalation dangers**
- Many pieces of the root privilege are enough to regain the whole thing
  - Access to files as UID 0
  - CAP_DAC_OVERRIDE
  - CAP_FOWNER
  - CAP_SYS_MODULE
  - CAP_MKNOD
  - CAP_PTRACE
  - CAP_SYS_ADMIN (mount)

**Legacy interaction dangers**
- Former bug: take away capability to drop privileges
- Use of temporary files by no-longer setuid programs
- For more details: “Exploiting capabilities”, Emeric Nasi

**Next time**
- Symmetric crypto primitives