Computer Networks
ICS 651

- Ethernet Network Design
- application-level protocol issues
- HTTP
- HTTPS (SSL/TLS)

Ethernet-based home setup

- one host, H, connected to the internet (the “router”)
- internally, 100Mb/s or 1Gb/s switch-based Ethernet
- externally cable modem, ADSL, or other technology
- H does Network Address Translation (L4 translation) so:
  - IP packets going out are rewritten to have H as source address, and often a different source port
  - incoming packets are rewritten to have the correct destination host and port number
  - for ICMP Echo, can rewrite identifier
  - for ICMP Error messages, need to look up port number in original header
- H can also:
  - perform firewall functions
  - allocate addresses via DHCP
  - be the default router for all internal computers

Network Address Translation and Firewall

- for TCP we know when the connection starts and ends, so can de/allocate state in the NAT box
  - and we generally disallow inbound connections
  - except when explicitly configured to accept them
- for UDP/ICMP we don't know when the connection is done, so we must cache: allocate when the first packet between (pair of IPs and ports) is sent from inside to outside, deallocate after a timeout
  - drop inbound UDP/ICMP that do not match a translation
- it generally doesn’t matter what local port we use for outgoing connections
- to run a server, must “poke a hole” in the firewall, i.e. tell the firewall to accept inbound TCP connections to a given port

Our Network So Far

- DNS
- IP, ICMP, and routing
- TCP and UDP
- Ethernet and WiFi (and SLIP/serial ports)
- can carry data end-to-end, reliably or quickly
- can build robust, efficient, fast, inexpensive local area networks
- what is missing: why are we carrying the data?

Tasks for the application

- user interaction
- accomplishing specific tasks
- encoding real-world data (and voice/video)
- "creating" and "consuming" data

Application Layer Functions

- client-server interactions
- sessions: logging in, user state
- security
- ultimate end-to-end evaluation of reliability (e.g. "reload" button in browser) and performance
- applications nest, e.g. social networking sites are nested within the web

Some Application Layer Protocols

- HTTP
- HTTPS/SSL/TLS
- ssh
- SMTP/POP/IMAP
- NTP
- FTP
- telnet
- etc.

HTTP

- HyperText Transfer Protocol
- request and reply headers, both encoded in ASCII (HTTP/2 and HTTP/3 use binary)
- headers are variable length, with variable fields
- first line is required, some fields are required
- header ends at first empty line
- ancestry: FTP
- HTTP/1.1 allows multiple requests/replies to be sent on a single TCP connection
- in-class exercise: explain why this is an improvement over HTTP/1.0 (one request/reply per TCP connection)

HTTP/1.1 example

- an HTTP request might look like this:

```
GET /~esb/ HTTP/1.1
Host: www2.hawaii.edu
Accept: */*
Connection: close
```

- a corresponding HTTP reply might look like this:

```
HTTP/1.1 200 OK
Date: Tue, 27 Nov 2018 20:59:07 GMT
Server: Apache/2.2.15 (Red Hat)
Last-Modified: Mon, 06 Aug 2018 23:29:37 GMT
ETag: "20a3d58c-225e-572cca69b1e19"
Accept-Ranges: bytes
Content-Length: 8798
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
...
```

HTTPS

- secure version of HTTP
- two types of protection:
  - authentication provides evidence that communication is occurring with the intended party (as identified by the URL)
  - encryption hides the contents of the communication (including passwords and credit card numbers) from eavesdroppers
- HTTPS is very similar to HTTP, except: SSL or TLS is used as an end-to-end secure tunnel (VPN) to connect the browser to the server (see RFC 2818)
- Transport Layer Security, or TLS (RFC 5246) provides authentication and secrecy using public-key cryptography to agree on a shared secret key
- this shared secret key is then used to encrypt the data

HTTPS authentication

- any public-key cryptosystem depends, for security, on knowing the other party's public key
- otherwise, man-in-the-middle attacks can easily succeed
- in https, the authenticity of a server's public key (correspondence between a public key and an IP+port number combination) is guaranteed by having the public key signed by a **certificate authority** (CA)
- most web browsers are pre-configured with the public keys of multiple certificate authorities, assumed trustworthy
- e.g. in firefox, Preferences -> Privacy & Security ->View Certificates
- the communication is safe as long as the certificate authority can be trusted (not always the case)
- servers usually authenticate clients in other ways, e.g. with a password or a credit card number or by access to an email address

Active Web

- the static web has been, and continues to be, was immensely successful as a repository of static information (of varying reliability)
- however, the user interaction model of the static web is limited to users clicking links
- for many purposes this is not adequate, e.g. logging in
- so the web evolved to provide support for server-side and client-side code execution that could provide different models of interactivity and customization
- this code generally increases the vulnerability to attack of both the client and the server
- client-side code might make further requests from the server, either using HTTP or HTTPS, or a custom protocol
- cookies allow the server to store state on the client, for purposes of later authentication or identity matching

A Better World-Wide Web?

- in-class discussion: is it possible to redesign the Internet to avoid at least some of the antisocial behavior in today's Internet?
- without breaking what has made the Internet as successful as it is?
- first step: list antisocial behaviors

Network Security

- Alice and Bob are trying to communicate securely, Charlie wishes to do all sorts of mischief
- for example, if Charlie is the man in the middle, he can read all the messages between Alice and Bob, maybe remove some or all of them, and perhaps add his own
- this might be accomplished by owning or subverting a router, or with other tricks including ARP spoofing or DNS spoofing
- in theory, encryption protects the contents, authentication guarantees that the sender is a machine with the correct key
- as long as the correct public key is known, Public-Key Encryption works well (but slowly)

Network Security in Practice

- in practice, applications have vulnerabilities:
  - array overflow (e.g. heartbleed) or stack overflow
  - being tricked into executing code or SQL commands (shellshock)
  - being designed for a different (e.g., secure) environment
  - not being secure against random or malicious inputs (shellshock)
- once the attacker can execute some code on a machine, privilege escalation might lead to executing other code as the superuser (root user)
- firewalls can prevent access to all applications other than the ones explicitly selected by a knowledgeable user -- but cannot protect permitted applications

Network Security in Practice

- in practice, applications have vulnerabilities:
  - array overflow (e.g. heartbleed) or stack overflow
  - being tricked into executing code or SQL commands (shellshock)
  - being designed for a different (e.g., secure) environment
  - not being secure against random or malicious inputs (shellshock)
- once the attacker can execute some code on a machine, privilege escalation might lead to executing other code as the superuser (root user)
- firewalls can prevent access to all applications other than the ones explicitly selected by a knowledgeable user -- but cannot protect permitted applications