ECE 435 – Network Engineering
Lecture 4

Vince Weaver

`http://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

7 September 2017

Announcements

- HW#1 was due.
- HW#2 will be posted. Write a mini webserver.

HTTP/2

- 2015. RFC 7540
- Google push through, extension of their SPDY (speedy) Microsoft and Facebook giving feedback
- Why does google care about (relatively) small increases in web performance?
- Leaves a lot of high level things the same. Negotiate what level to use.
- Decrease latency of rendering web pages:
  - compress headers

- Server can push data the browser didn't request yet but it knows it will need (like images, etc)
- pipeline requests

  Send multiple requests without waiting for response

  good on high-latency links (FIFO on 1.1, new makes it asynchronous)
- multiplex multiple requests over one TCP connection
- head-of-line blocking problem?

  line of packets held up by processing of first

  FIFO first requests waits until done until next, can't run in parallel

- Page load time 10-50% faster
- While can use w/o encryption, most browsers say will only do with encryption
- Criticism: was rushed through. Is way complex. Does own flow control (has own TCP inside of TCP) Re-implements transport layer at application layer

What if Server Overloaded?

- Slashdot effect (modern: HackerNews?)
- caching/proxy – squid
- Content Delivery Network – akami
- Server farms

Security

- SSL – Secure Socket Layer
- Replaced by TLS (Transport Layer Security)
- Port 443 for https
- Public key encryption.

Setting Up a Web-server

- Apache

Web Seach

- Web-bots index the web. robots.txt file
- Altavista, Hotbot, Excite, Inktomi, etc.
- Curated search like Yahoo (people organize links rather than automatically search)
- Google (1996 some machine in Stanford, 1997-1998)
- MSN search 1999, rebranded Microsoft Bing 2009

telnet/rlogin/rsh/ssh

- telnet – login to remote system (tcp port 23) everything (including passwords) sent in plain text
- rsh/rlogin – remote shell, remote login. (tcp port 514) Didn't even need password, could configure to let you run commands on remote machine. Security based if you had same username on both machines, assumption was getting root on a UNIX machine and connected to Ethernet was expensive/difficult

SSH secure shell

- tcp port 22
- can login, run commands, tunnel tcp/ip, tunnel X11, file transfer (scp, sftp)
- Large number of RFCs
- Version 1: 1995, originally freeware but became private
- Version 2: 2005, openBSD based on last free version
- For security reasons there's a push to drop Version 1
- uses public-key cryptography
- transport layer: arranges initial key exchange, server

authentication, key re-exchange

- user authentication layer: can have password, or can set up keys to allow passwordless, DSA or RSA key pairs
- connection layer: set up channels
- lots of encryption types supported, old ones being obsoleted as found wanting
- Various ssh servers/clients. openssh. dropbear
- Diffie-Helman key exchange?
  - This is a public/private key thing
  - Based on discrete logarithms?
  - Wikipedia has a weird colored paint analogy

Encryption

- Most crypto papers involve Alice and Bob (maybe Eve)
- Plaintext is transformed by some sort of function parameterized by a "key" into cyphertext. This is then transmitted. The other side then decrypts it.
- What can be kept secret? Security by obscurity? Kerckhoff's principle: "All algorithms must be public; only the keys are secret."
- Combination lock analogy. Longer the key, the harder it is to brute-force

- easy: rot13
  Substitution cipher. Weakness: English text easy to predict ('e' most common letter)
  What about double-rot13?
- transposition cipher, keep letters same, re-arrange order
- hard: one-time-pad
  unbreakable. Downside, must keep it, must have enough bits, cannot reuse, transporting.

Secret Key Algorithm

- Key is secret
- How do you get it to the other person?
- How many keys do you need (ideally one per connection)

Symmetric Key Algorithms

- Use same key for encryption and decryption
- Block ciphers, take block of data and encrypt it to same size block (why in blocks?)
- P-box (permutation), S-box (substitution)
- shift/permute/xor
- *very* important that the key is picked randomly.
- DES – Data Encryption Standard
  From 1976. 64 bit key (56-bits used). NSA had say on key size. 19 stages based on Key. widely used until

broken. Competition to break various sizes.

- 3DES (running DES three times) [encrypt/decrypt/encrypt with only two keys? Why? 112 bits seen as enough, also if set keys to same then it's same as single-DES (back compat)]
- AES – Advanced Encryption Standard – replaces DES
NIST had a contest to find new standard
Rijndael won. Intel chips have AES instructions

Public Key Encryption

- Assymetric/Public Key
- Encryption key weakest link of symmetric encryption, as both sides have it and if anyone leaks it, all is lost
- Have a public key that anyone can use to encrypt a message. Can only be (easily) decrypted by a secret, private key
- Hard to solve math problems. Integer factorization, discrete logarithm, elliptic curves
- Often only used to encrypt small amounts of data,

i.e. used to encrypt a symmetric key used for longer transactions

- RSA – Rivest/Shamir/Adleman at MIT
  - Choose two large primes p and q (1024+ bits)
  - n=p*q, z=(p-1)*(q-1)
  - Choose number relatively prime to z: d (no common factors)
  - Find e such that e*d mod z=1
  - Divide plaintext into blocks $0 \leq P < n$, blocks of k bits where k largest $2^k < n$
  - To encrypt, compute $C = P^e \ mod \ n$

- To decrypt, compute $P = C^d \; mod \; n$
- public key is e,n. private key is d,n
- Hard to break as you need to factor n (hard)
- How do you find p and q? Random number, then apply various tests to determine if prime
- Example from Tanenbaum Figure 8-17:
  Pick two large primes: p=3, q=11
  n=p*q=33, z=(p-1)*(q-1)=20
  d=7 (no common factors with 20)
  $7 * e \; mod \; 20 = 1$ so e=3
  To encrypt say "13", $13^3 = 2197, mod 33 = 19$

To decrypt say "19", $19^7 = 893871739 mod 33 = 13$

- Other Types
  - Prime Number Factoring
  - Elliptic Curve Cryptography (ECC)
    Smaller keysize
- Common uses: public key encryption, public key used to encrypt message only holder of private key can decrypt digital signature: message signed with private key and anyone with access to public key can verify the original sender

Cryptographic Hash Functions

- Maps a document of arbitrary size to a fixed size
- Easy to calculate, hard to reverse. Only real feasible way to reverse is brute-force search
- Should not be able to find two different messages with same hash
- Small changes in document should lead to very different hashes
- Two items with same hash are a collision
  Are collisions useful? If you can map documents of

same filetype, or if somehow same document with lots of garbage on end

- Break file up into chunks, do a series of operations to "compress" it, often shift, xor, or, add, and, not
- md5 `md5sum`
  128-bit md5 hashes, create checksum, uniquely ID file
  Well, not really unique. It's been broken, can find (with great difficulty) collisions
- SHA-1
  Developed by NSA
  Used by git

- Uses: passwords (/etc/shadow), (mostly) uniquely iding a file (git), verifying file contents (download, error checking), bitcoin?
- Problem: how do you verify the public key belongs to the person who they say it is? (on website? what if someone intercepts and replaces, mitm style)