ECE 435 – Network Engineering
Lecture 7

Vince Weaver

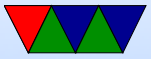
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

5 February 2025

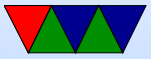
Announcements

- HW#1 almost done grading
- HW#2 due Friday, don't forget



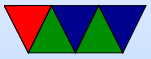
Remote Connections

- One of the first uses of networks was logging into remote computers
- In the old days computers were super expensive
- Often there'd be one big one in a central location and you could connect remotely with smaller, cheaper terminals
- wam / glue story



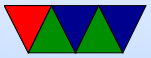
Historical – telnet

- log in to remote system
- (tcp port 23)
- everything (including passwords) sent in plain text
- client not much more complex than HW#1
- telnetd server providing connections, gives you a login via the Linux pty (pseudo-tty) interface



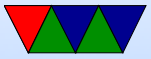
Historical – rsh/rlogin

- 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



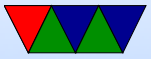
SSH secure shell (background)

- Encrypts a connection between machines
- 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 support for Version 1 essentially discontinued

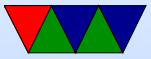


SSH (implementation)

- uses public-key cryptography
- transport layer: arranges initial key exchange, server authentication, key re-exchange
- user authentication layer
  - can have password
  - can set up keys to allow passwordless
- connection layer: set up channels
- lots of encryption types supported, old ones being obsoleted as found wanting

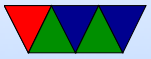


- Various ssh servers/clients
  - openssh
  - dropbear
  - mosh (fancy alternative)
  - ssh3 (under development)



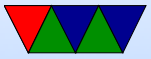
Diffie-Helman key Exchange

- (we'll talk about this later)



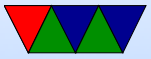
ssh downsides

- Any downsides?
- Takes a lot of compute power. Not a problem for most modern machines.
- Maybe more of an issue on small embedded systems
- Does make it hard to put your 8-bit machine on the internet without a helper device
- Older protocols/keys expire which can make it hard to connect to older machines/operating-systems



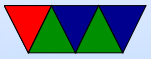
ssh security

- Brute forcing passwords is a major issue.
- Any publicly visible ssh server constantly being hit by username / password dictionary attacks
- Also issue if machine with your private key compromised (can be used to connect to any machine you had setup to allow passwordless login)



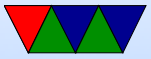
ssh attack mitigation

- Fail2ban
- Nonstandard port
- Port knocking
- Call asterisk for one-time pin?
- No-password (key only)
- Two-factor authentication (LCD keyfob)



Encryption Background

- Most crypto papers involve Alice and Bob (maybe Eve)
- **Plaintext** is transformed by some sort of function parameterized by a "key" into **Ciphertext**.
  This is then transmitted. The other side then decrypts
- 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

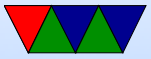


Encryption Types – Substitution

- Substitute each character for another with lookup table
- Decrypt by just doing the reverse
- Trivial Example: rot13 (Caesar Cipher)
  - A-N, B-O, C-P, etc.

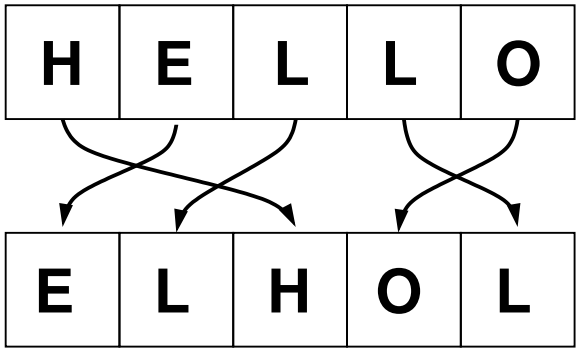
A	B	C	D	E	F	. . .	Z
N	O	P	Q	R	S	. . .	M

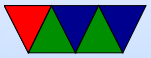
  - ECE encrypts to RPR
  - Weakness: English text easy to predict ('e' most common letter)
  - What about double-rot13?



Encryption Types – Transposition

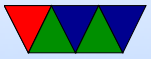
- transposition cipher, keep letters same, re-arrange order





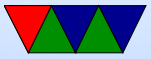
Encryption Types – One Time Pad

- Unbreakable
- Have random data, same size as that being encrypted
- xor random data with message
- Downsides:
  - must have enough bits
  - bits must be strongly random
  - cannot reuse data
  - transporting data to other end, destroying when done



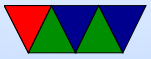
Secret Key Algorithm

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



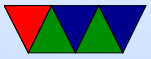
Symmetric Key Algorithms

- 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.



Symmetric Key Implementations – Historical (DES)

- 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.



Symmetric Key Implementations – Historical (3DES)

- 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)