ECE 435 – Network Engineering
Lecture 5

Vince Weaver

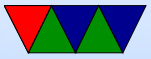
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

1 February 2022

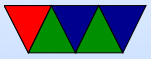
Announcements

- HW#2 posted, due Thursday.
  This is possibly the most difficult assignment.

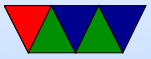


HW#1 Review – Sockets Code

- What if close early control-C
- Why port 31337?
- With write syscall, need to set the size to send back.
  If you always send size of buffer, it sends lots of useless zeros.
  If you always send 25 (cut-and-paste error) you don't send all data.
  Use `strlen()` to get size?
- Don't ignore compiler warnings.

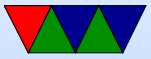


What if `toupper()` not found?
manpage. Need to include `ctype.h`



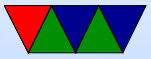
HW#1 Review – Specifications

- When you type "bye" it would exit both sides.
  (bye by itself? cr/lf? byet?
- Postel's Law: strict what send, generous receive?
- Example of browser accepting herf instead of href? why could this be bad?



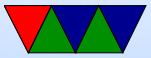
HW#1 Review – Something Cool

- Something cool... don't interfere with default behavior. You can optionally read command line args, but don't change default. Also if requires command line args, in error message say what they would be
- Can you just document it in the README? Sadly people don't always read documentation?



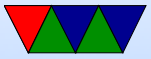
HW#1 Review – The Rest

- Comment your code!
- OSI reference model – was hoping for names not number
  - Bits and voltages – physical layer (1?)
    Not hardware layer
  - Routing packets – network layer (3?)



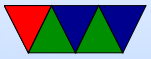
Homework #2 Notes

- If connecting on same machine, can use localhost
  if over network, must use IP address. Can find this various ways (`ip addr` on Linux)
- If browser confused, be sure you aren't sending extra zeros. strlen() is your friend
- Debugging? `strace`, `tcpdump`
- If getting segfaults, try using `gdb`



HW#2 Hints – Reading Request

- First be sure you are getting the incoming header. Print it or use strace to verify.
- Some web-browsers might send really big requests, be sure getting it all
  - Use big enough buffer? 4096 bytes? How big?
  - How would a "real" system do this?
  - `malloc()`, `realloc()` if not big enough?
    Might be overkill for this homework



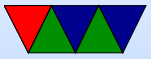
HW#2 Hints – Parsing the Request

- Know how to search for a string and point to location after it?
  - Find a string and point to beginning of it.

```
char *pointer;
pointer=strstr(haystack,needle);
```

  - Look for `"GET "`

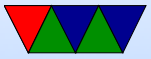
    Actually points to beginning of GET. How to skip ahead?
  - pointer+=4 is one way. (pointer math, ugh)
  - How to get to first space?



- `strtok(pointer," ");`
  Will split the string into chunks, put 0 at end.
- Also can do this manually;

```
pointer2=pointer;
while(*pointer) {
   if (pointer==' ') {
      *pointer=0;
      break;
   }
   pointer++;
}
printf("%s\n",pointer2);
```

- Be sure to strip off initial /, and if it's just / return index.html

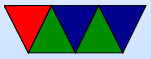


HW#2 Hints – Generating Response

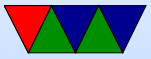
- Print to stdout to verify what sending, also can use lynx / wget.
- Know how to construct a string on the fly? `strcat()`, `sprintf()`
  `strcpy()` first bit in.
  `strcat()` additional strings.
  If you want formatting you can do things like

```
sprintf(temp_string,"File size=%d\r\n",filesize);
strcat(out_string,temp_string);
```

  Create big enough buffer.



- How to find size of a file?
  Can read it in, and count. Or can use the stat (`man stat.2`) need .2 (or `man -a`) as there's a command line tool called stat that comes ip first.

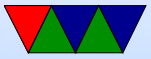


HW#2 Hints – Sending File Contents

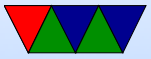
- How to read/write file. There are a large number of ways to do this. `open()/read()/write()/close` `fopen()/fread/fwrite/fclose` (careful! Buffered! And maybe need fdopen() to print to file descriptor).

```
fd=open(filename,O_RDONLY);
if (fd<0) fprintf(stderr,"Error opening %s\n",filename);
while(1) {
   result=read(fd,buffer,256);
   if (result<=0) break;
   write(network_fd,buffer,result);
}
```

  Be sure to close afterward.

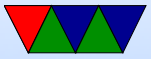


Remote Connections



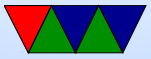
telnet/rlogin/rsh/ssh

- 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



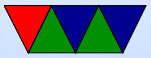
SSH secure shell (background)

- 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 there's a push to drop Version 1
- uses public-key cryptography

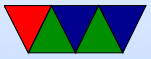


SSH (implementation)

- 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?



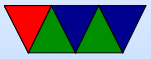
- This is a public/private key thing
- Based on discrete logarithms?
- Wikipedia has a weird colored paint analogy



ssh security

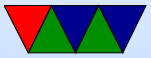
Brute forcing passwords is a major issue.

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



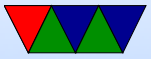
Alternatives to SSH?

- mosh



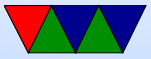
Encryption

- 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



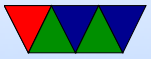
Encryption Types

- 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.



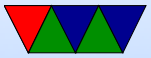
Secret Key Algorithm

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



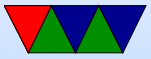
Symmetric Key Algorithms

- 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.



Symmetric Key Implementations

- 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



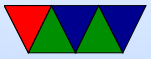
NIST had a contest to find new standard

Rijndael won.

NSA allows for classified data

Intel chips have AES instructions

Galois Field Theory (Gal-wah? interesting mathematician)



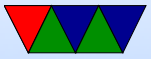
AES Encryption

1. Key Expansion

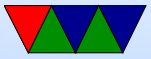
2. AddRound on initial key (add/xor)

3. 9/11/13 rounds (depending on key size)

   (a) non-linear substitution (w lookup table)
   (b) transposition/row shift
   (c) mix columns (matrix multiply)
   (d) addround (xor again)

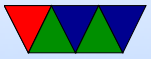


4. Final round: a,b,d again



AES Attacks

- In theory take billions of years to brute force
- "Attack" means finding some way to decode key faster than brute force
- Have been some but none really effective yet
- Side Channel Attacks are possible though



AES Performance

- Pentium Pro 200MHz: 11 MBits/s
- Modern Intel/AMD with AES in hardware, multiple GB/s