Quick note

- “Where does locale command get its data”
  - If you run the command locale, how does it know what to output?
- Check piazza
  - Also see piazza question 10
- This week (today), office hours 5pm-7pm (instead of 4pm-6pm)
- Assignment 1 due today
  - If you’re not done, remember the lateness policy
- Assignment 2 may be “easy”, other assignments much more difficult!
  - Time to adapt to online format, sort out classes, etc

Feedback / Office Hours

- Tameez Latib
  - tameezlatib@gmail.com, please add “CS35L” to the subject line
  - Office Hours: Monday **5pm-7pm** (or by appointment)
  - Feedback: https://forms.gle/6kcJ2aJtzAzFMhHQ7 (anonymous google form)

Encryption

- Warning: this is a very brief, high level overview
  - Some of the topics are covered in-depth in other courses
- In general:
  - Alice wants to send a private message to Bob
  - Alice and Bob agree on a method + key/password/etc
  - We call the unencrypted message “plaintext”
  - We call the encrypted message “ciphertext”
  - Alice sends over ciphertext
    - Anyone listening (Eve) will not be able to decrypt without the key
  - The goal is that ONLY Alice and Bob know the message contents
  - Eve can know the method, but does not know password
- Idea is simple, in practice it can be complicated...

The basics, first attempt:

- Okay, so let's try something simple: symmetric encryption
  - Alice and Bob have the same key
- Alice and Bob agree to use a caesar cipher.
  - Letter -> letter + key.
    - key = 5,
      - a -> a + 5 = f.
      - z -> z + 5 = e.
  - Only Alice and Bob know the key
- Is this safe?

Brute force

- Eve receives the encrypted message
- Eve tries all 26 keys
- Only one message will “make sense”

Second attempt:

- Okay, a bit more complicated
- We look up one of the best symmetric encryption algorithms
- Alice and Bob choose to use AES
  - Which has no apparent encryption flaws
- Is this safe?

Secure, but problematic

- If Eve does not know the key, there is no way she can decrypt
- Problem:
  - How do Alice and Bob agree on such a password?
  - Alice and Bob must communicate to send the password safely
    - Maybe they can encrypt it

Symmetric methods, summary

- Generally Fast
- Easy to understand (encrypt and decrypt use same key)
- Distribution of key is hard
- Current standard: AES
  - Algorithm is known
  - Nobody knows how to break it
  - Secure!
- Encrypt(plaintext, key) = ciphertext
- Decrypt(ciphertext, key) = plaintext

Third attempt:

- Let’s have two keys
  - Every person has public key and private key (asymmetric)
- To send a message to Bob, Alice encrypts using Bob’s public key
  - ONLY Bob has his private key, which can decrypt messages sent to Bob
- Now:
  - Distribution of keys no longer a problem
- Encrypt(plaintext, Bob public key) = ciphertext for Bob
- Decrypt(ciphertext for Bob, Bob private key) = plaintext
- Everyone (including Eve) knows public keys
- Is this safe?

Yes, and no

- Eve has no knowledge of private keys, so she cannot decrypt
- How does Alice know what Bob's public key is?
  - What if Eve sent hers
- These attacks are called "man in the middle"
  - Attacker intercepts messages
  - Attacker pretends to be someone else

Asymmetric

- Generally slower
- Harder to understand (two keys)
- No need to distribute key
- Example: RSA
- There are other attacks
  - Example: How do you know who you're talking to?

Which to use?

- You can actually use both
- Use Asymmetric encryption to send over the symmetric key
  - Alice creates a symmetric key, encrypts with Bob’s public key, and sends it
  - Bob decrypts with Bob’s private key, and now has the symmetric
- Now Alice and Bob both have the key, and can use fast symmetric encryption

SSH

- First server must authenticate itself
  - Check ~/.ssh/known_hosts
- Next, user must be authenticated by password OR
- SSH keys:
  - Server looks up your public key
  - Encrypts(message) with your public key and sends it
  - If you can decrypt and return message, the server knows it’s you
- Uses asymmetric encryption to share a secret key
- Authentication uses signatures
  - Might talk about on Wednesday

Lab: part 1

- For all parts:
  - Pretend you are client and server
    - Example: let lnxsrv07 be client lnxsrv09 be server
  - Write down which you used as client/server in log. Be consistent!
- Goal: Use ssh-agent. Login without password
- How? Use ssh keys
- Steps for you:
  - Try to create a public / private key
  - Find out where to store this information
    - Who creates? Who needs what info? (Client/server)
  - From a completely new session (log out and log back in):
    - Use ssh agent, add private key
    - Ssh without password

Part 2

- Goal: use X forwarding, make xeyes work
- On mac/linux: Ssh vs ssh -X vs ssh -Y ?
  - Xquartz
  - May need to edit ssh_config
    - X11Forwarding yes
- If on windows
  - Xming
  - putty -> connection -> ssh -> x11: enable X11 forwarding
- Try: xeyes
- Basically X forwarding takes care of graphical data

Part 3

- Goal: Multi-hop ssh
- http://sshmenu.sourceforge.net/articles/transparent-mulithop.html
  - Follow this tutorial
- That's it
-
- If you need any reference material (for any part), check the links on https://web.cs.ucla.edu/classes/fall20/cs35L/assign/assign2.html

Questions?

- About assignment 1/ 2
- About security / cryptography?
  - For the final, you don’t need to know a lot*
    - This isn’t a security class
- Grading for assignment 2:
  - Make sure everything is written down in log
  - For the homework questions:
    - Check the hint slide
    - No ‘right’ answer as long as you explain your choices (well)