System Administration & Security

COMP 175 | Fall 2021 | University of the Pacific | Jeff Shafer

Linux Fundamental Skills:
User Accounts
Sudo Command
Passwords & Cracking

Overview

Recap

- AWS
  - EC2 instances
  - Security Groups
  - VPC networks
  - Billing & alerts
- Linux Fundamentals
  - SSH
  - Directories & Navigation

This Week

- Lecture
  - User Accounts
  - Sudo command
  - Password principles
- **Lab 5 – Web Server (Part 2)**

User Accounts

User Accounts

Create the user “tiger”

```
$ sudo adduser tiger
```

Set (or reset) the password for the ”tiger” user

```
$ sudo passwd tiger
```

Create the group “tigerteam”

```
$ sudo addgroup tigerteam
```

Add user “tiger” to group “tigerteam”

```
$ sudo adduser tiger tigerteam
```

User Accounts

Remove the user “tiger” from the group “tigerteam”

```
$ sudo deluser tiger tigerteam
```

Delete the user “tiger”

```
$ sudo deluser tiger
```

Delete the group “tigerteam”

```
$ sudo delgroup tigerteam
```

User Accounts

List current password expiration, inactivation, etc… settings for user “tiger”

```
$ sudo chage -l tiger
```

Change password expiration, inactivation, etc… settings for user “tiger”

```
$ sudo chage tiger
```

User Accounts

```
$ cat /etc/passwd
```

```
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
tiger:x:1001:1001:Tiger,,,:/home/tiger:/bin/bash
```

- Contents of /etc/**passwd** – Fields separated by **:** character
  - Username or login name
  - Encrypted password – *Legacy, now in /etc/shadow*
  - User ID number
  - Group ID number
  - User description (name, phone, title, …)
  - User's home directory
  - User's login shell

User Accounts

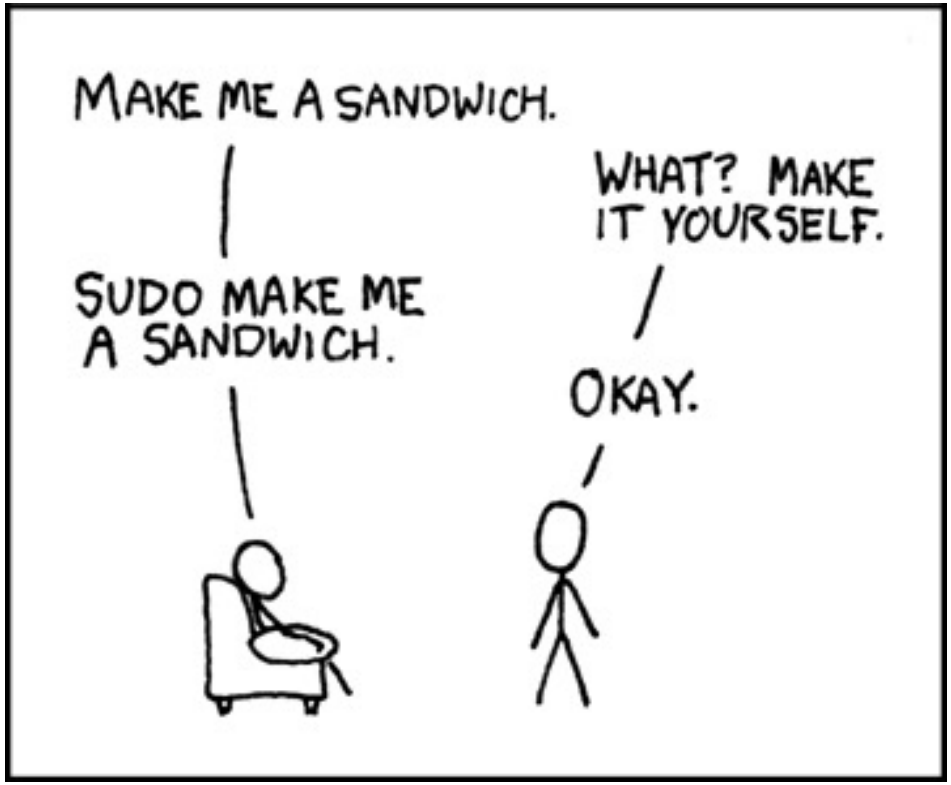
```
$ sudo cat /etc/shadow
```

```
ubuntu:!:18201:0:99999:7:::
grav:$6$zHTYSdnJ$XXXXXXXXXXXXX1:18201:0:99999:7:::
```

- Contents of /etc/**shadow** – Fields separated by **:** character
  - Username or login name
  - Encrypted password (! or * represent blank password – not allowed to login)
  - Date of last password change
  - Minimum required days between password changes
  - Maximum allowed days between password changes
  - Number of days in advance to display password expiration message
  - Number of days after password expiration to disable the account
  - Account expiration date
  - Reserved field

Password Hashing Formats

- Password is encoded as: **`$id$salt$hashed`**
- The quality of Linux password hashing algorithms (in `crypt()` / glibc library) has improved over time
- **`$id`** allows for different hashing algorithms
  - `$1$` is MD5 *(old, insecure)*
  - `$2$` is Blowfish
  - `$3$` is Eksblowfish
  - `$4$` is NT hashing
  - `$5$` is SHA-256
  - `$6$` is SHA-512 *(new, secure)*



https://xkcd.com/149/

Sudo

Sudo

- The `sudo` command allows you run a command *as-if* you are logged in as the root (*admin*) account
- Common use cases
  - Installing or updating applications via package manager
  - Starting, stopping, configuring system services

https://www.howtoforge.com/tutorial/sudo-beginners-guide/

Sudo Examples

Run `apt update` as the `root` user:

```
$ sudo apt update
```

Run `nano test.txt` as the `root` user:
(nano is a text editor, and this will create or open the file test.txt)
Note 1: If the file already exists, it will retain its current owner & permissions
Note 2: If the file does *not* exist, it will be owned by the **root** user!
Whether this is desirable depends on your goals!

```
$ sudo nano test.txt
```

Sudo Configuration

Configuration file: `/etc/sudoers`:
(Edit with the special command `sudo visudo`)

```
$ sudo visudo
```

Why use visudo instead of another text editor?

visudo edits the *sudoers* file in a safe fashion. visudo locks the *sudoers* file against multiple simultaneous edits, provides basic sanity checks, and checks for parse errors before installing the edited file. If the *sudoers* file is currently being edited you will receive a message to try again later.

Sudo Configuration

Configuration file: `/etc/sudoers`:
(Edit with the special command `sudo visudo`)

```
$ sudo visudo
```

/etc/sudoers

```
Defaults env_reset
Defaults mail_badpass
Defaults secure_path="/usr/local/sbin:
/usr/local/bin:/usr/sbin:/usr/bin:/sbin
:/bin:/snap/bin"

root ALL=(ALL:ALL) ALL
%admin ALL=(ALL) ALL
%sudo ALL=(ALL:ALL) ALL

#includedir /etc/sudoers.d
```

/etc/sudoers.d/90-cloud-init-users

```
ubuntu ALL=(ALL) NOPASSWD:ALL
```

Format of `sudoers` file?

1. Username that rule will apply to (root)
2. Hosts that rule will apply to (ALL)
3. Groups that this user can run command as (ALL)
4. Commands that this user can run (ALL) *(and also don't prompt for password)*

Related Examples

Switch **U**ser to “tiger” (i.e. log on as them):

```
$ su tiger
# Prompted for “tiger” password
```

Switch **U**ser to root account (i.e. log on as root):

```
$ su
# Prompted for “root” password
```

As root, **S**witch **U**ser to root account (i.e. log on as root):

```
$ sudo su
# Prompted for **YOUR** password
```

```
$ sudo -i # Equivalent command
# Prompted for **YOUR** password
# Will be able to run interactively as root user
```

Sudo Pitfall (Example)

Get a directory listing and redirect output to a file

```
$ ls > /home/user/myfile.txt
```

Get a directory listing and redirect output to a file in a directory owned by root

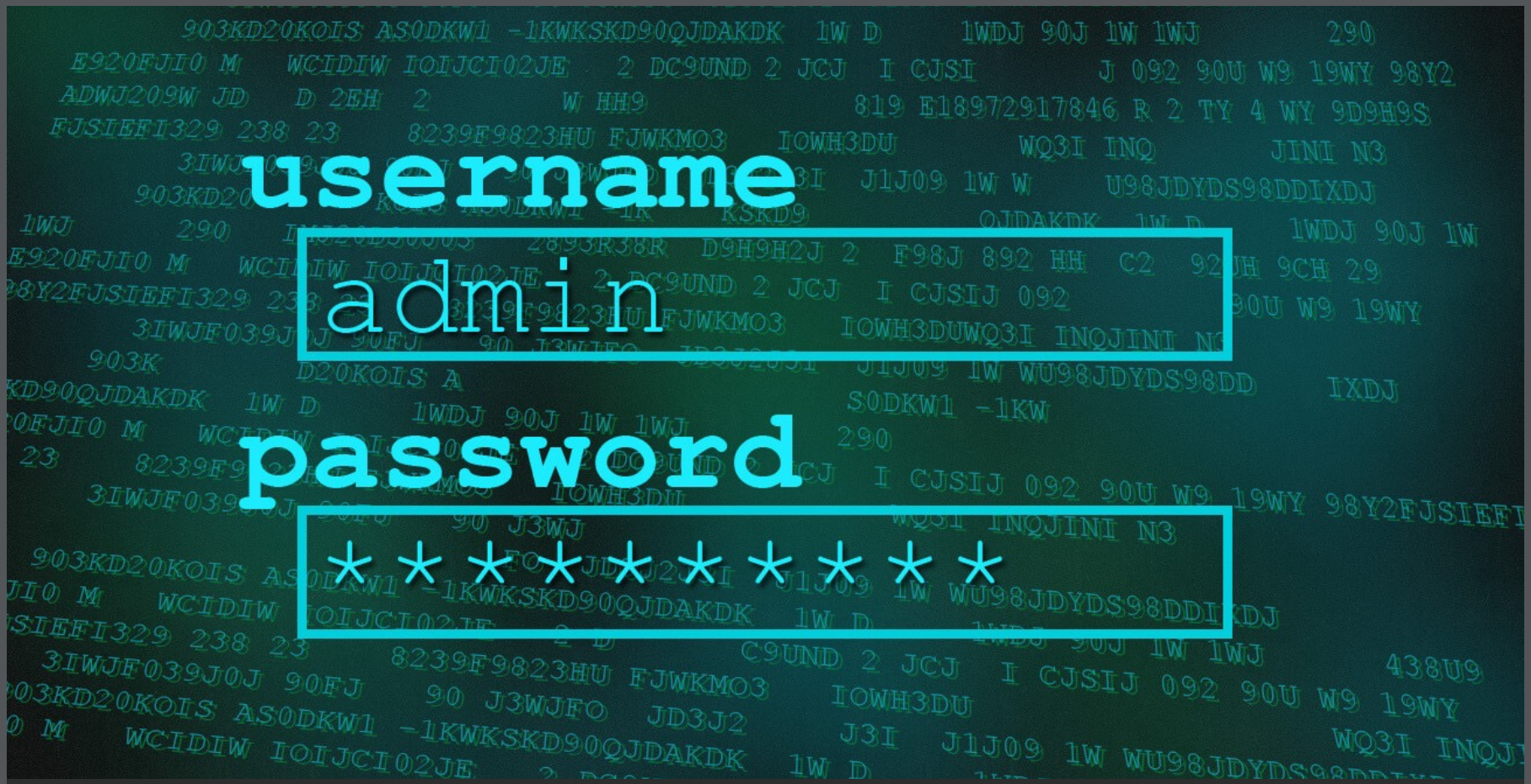
```
$ sudo ls > /root/myfile.txt     # This WON'T WORK
```

The "list" command is run as the root user…

But the output redirection (">") is done by the SHELL of the non-sudo user! No escalated permissions here…

Solution? Find another way…

```
sudo sh -c "ls > /root/myfile.txt"
```



Passwords

Password Creation

- **Who creates passwords?**
- **User**: typically guessable passwords
- **System**: can produce hard-to-guess passwords (e.g., random ASCII character strings)
  - But users can't remember them
- **Administrators:** Same as above

User Passwords

- Top-10 Most Common Passwords of 2016
  - 123456
  - 123456789
  - qwerty
  - 12345678
  - 111111
  - 1234567890
  - 1234567
  - Password
  - 123123
  - 987654321
- Users pick terrible passwords!
  - (duh)

https://blog.keepersecurity.com/2017/01/13/most-common-passwords-of-2016-research-study/

Password Strength

- Strength = Resistance to Brute Force
  - High entropy = high resistance
  - If 2^X guesses are required, entropy is X
- Example: Password of length *L* from alphabet of *N* characters
  - N^L possible passwords
  - $2^X = N^L \rightarrow$ **$X = L \log_2 N$**
- NIST recommendations (2006)
  - 14 bits minimum entropy, 30 bits better…

Password Strength

- Example: 8 character password, 26 character alphabet
  - Entropy = 8 $\log_2$ 26 = 37 bits
  - **So are we good?**
- Huge problem – *real* humans are not choosing uniformly random characters for their passwords
  - How about imposing some rules on passwords the users can select?

Help: List of Password Rules

Jeff

portal.cs.oag.state.tx.us/OAGStaticContent/portal/login/help/listPasswordRules....

Rules

1. The password must be **exactly** 8 characters long.
2. It must contain **at least** one letter, one number, and one of the following special characters.
   a. The **only** special characters allowed are: @ # $
   b. A special chaacter must **not** be located in the first or last position.
3. Two of the same characters sitting next to each other are considered to be a "set." No "sets" are allowed. **Example: rr, tt**
4. Avoid using names, such as your name, user ID, or the name of your company or employer.
5. Other words that cannot be used are Texas, child, and the months of the year.
6. A new password cannot be too similar to the previous password.
   a. **Example:** previous password - abc#1234; unacceptable new password - acb$1243
   b. Characters in the first, second, and third positions cannot be identical. (abc*****)
   c. Characters in the second, third, and fourth positions cannot be identical. (*bc#****)
   d. Characters in the sixth, seventh, and eighth positions cannot be identical. (*****234)
7. A password can be changed voluntarily (no Help Desk assistance needed) once in a 15-day period. If needed, the Help Desk can reset the password at any time.
8. The previous 8 passwords cannot be reused.

One way to create a password is creative spelling and substitution. Examples:

1. phuny#2s
2. fish#1ng
3. t0pph@ts
4. run$4you
5. ba#3ries

Top of page

Password Recipes

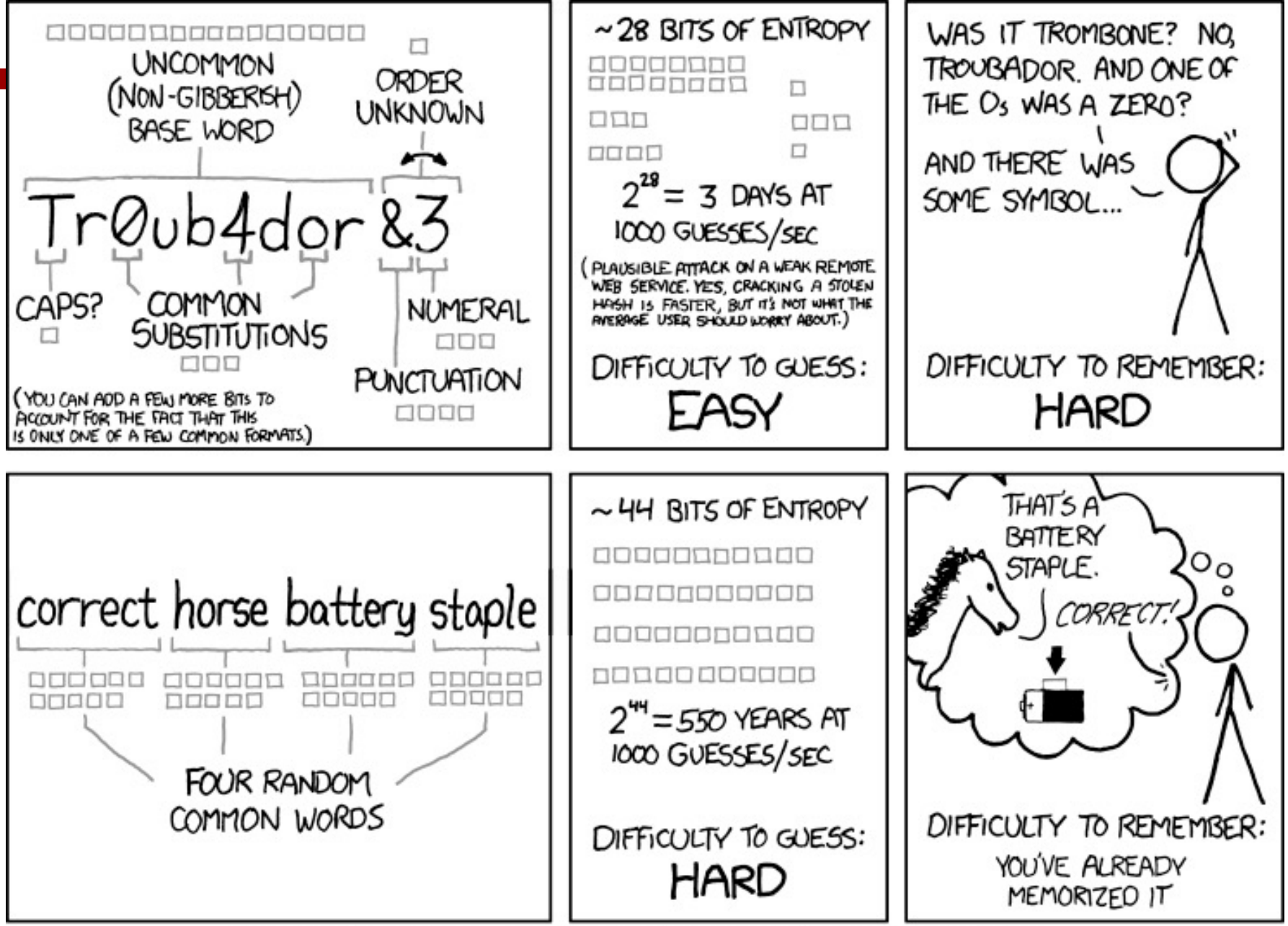
Attorney General of Texas, Child Support Division
http://portal.cs.oag.state.tx.us/OAGStaticContent/portal/login/help/listPasswordRules.htm

Password Recipes

- Will password rules help entropy?
  - Users are annoyed and choose weaker passwords ☹
  - Users pick easy to guess passwords that minimally comply with recipe ☹
- **Warning!** The attackers know <u>all of your clever password tricks</u>, and program their brute force attempts to try these permutations!
  - Only an *idiot* attacker would brute force `aaaaaaaaaaaaaaaa` to `zzzzzzzzzzzzzzzz`

Password Creation

- What if the system adds some randomness at the beginning or end of the user password? (and user must remember it all)
  - Users choose weaker base passwords ☹
- Password wallets / Password managers ☺
  - Pro: Have truly random + unique passwords ☺ ☺
  - Con: Have to trust password manager ☹
- Passphrases instead of passwords?



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

"XKCD Method"

- Good analysis of XKCD method math for `Tr0ub4dor&3`
  - https://blog.agilebits.com/2011/08/10/better-master-passwords-the-geek-edition/
- Passphrase assumption:
  - Get a dictionary of 2^{11} easy to spell English words
  - Pick 4 of them at **RANDOM**
  - Hence, 2^{44} combinations to brute force (44 bits of entropy)
  - Few days on a GPU via Hashcat? (for non-KDF hashes)
- Is it as good as a truly random 30 character password? No. That would be 30 $\log_2(26)$ = 141 bits of entropy.
  - But it's much much better than the password your mom usually picks

Kerckhoff's Principle

- Simplified version by Claude Shannon
  - "The enemy knows the system"
- Assume adversary knows everything about your password generation scheme (no secret methods!)
- Only safety is via high entropy and many (many!) brute-force combinations

Password Cracking

Password Cracking

- Why do we care about this in a class about *system administration?*
  - Do you want to *audit* the quality of your user's passwords?
  - Do you want to understand how attackers might be working against you?

Obtaining Passwords : Methods

Online Attack

- Generate password *guess* and send it to target to verify
- Pros
  - Will work if you have *no other choice*
- Cons
  - Slow (network latency + target throttling)
  - Can lock out legitimate users due to repeated failures
  - Can set off security alarms

Offline Attack

- Generate password *guess*, hash it, and compare to hashed password you previously obtained via exploit
- Pros
  - Dramatically faster!
    - No network latency
    - No target throttling
    - Parallelizable
  - No risk of account lockouts
  - Less detectable

Obtaining Passwords : Cracking

- Brute force password cracking (either online or offline) requires **wordlist** + set of permutations on the wordlist
  - Engine just tries every possible word + permutation and checks result
- The larger the wordlist, the longer it will take to test
  - Speed also affected by available parallelism (GPUs?) and complexity of the password hashing algorithm (more on cryptography later!)
- Vary size based on specific scenario
  - Shorter wordlists for online attacks?
  - Longer wordlists for offline attacks?

Obtaining Passwords : Cracking

- Kali Linux *(security-focused Linux distribution)* has a number of small and medium wordlists available
  - `/usr/share/metasploit-framework/data/wordlists/`
  - `/usr/share/wordlists/`
- Larger wordlists can be obtained online
  - https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm (15GB uncompressed)

Password Cracking Utilities

Online Attack

- **THC Hydra**
- Free & Cross-Platform
- Supports **large variety** of online applications to target
  - HTTP, SSH, FTP, SMB, SMTP, RDP, VNC

https://github.com/vanhauser-thc/thc-hydra

Offline Attack

- **John the Ripper**
- Free & Cross-Platform
- Supports **huge variety** of password hashes
  - Linux, Mac OS, Windows, database servers, WiFi PSKs, encrypted private keys, disk images, compressed archive files, …

https://www.openwall.com/john/

Wrap-Up

- Questions?
- Concerns?

- **This Week**
  - **Lab 4** – Web Server (Part 1)
  - **Lab 5** – Web Server (Part 2)