Vulnerabilities, exploits, and the secure design principles of Saltzer and Schroeder

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

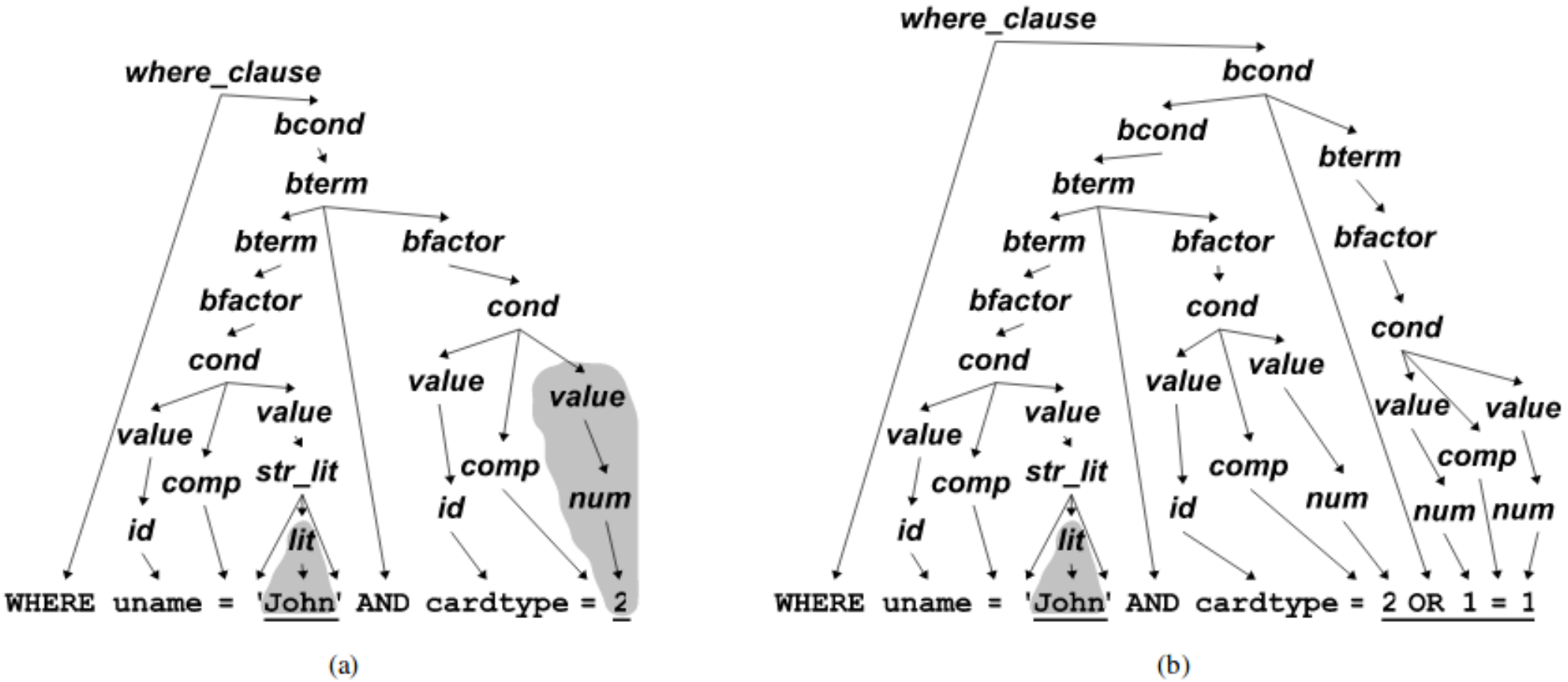


Figure 4. Parse trees for WHERE clauses of generated queries. Substrings from user input are underlined.

Cross-site Scripting (XSS)

Send a message in the WebCT platform:

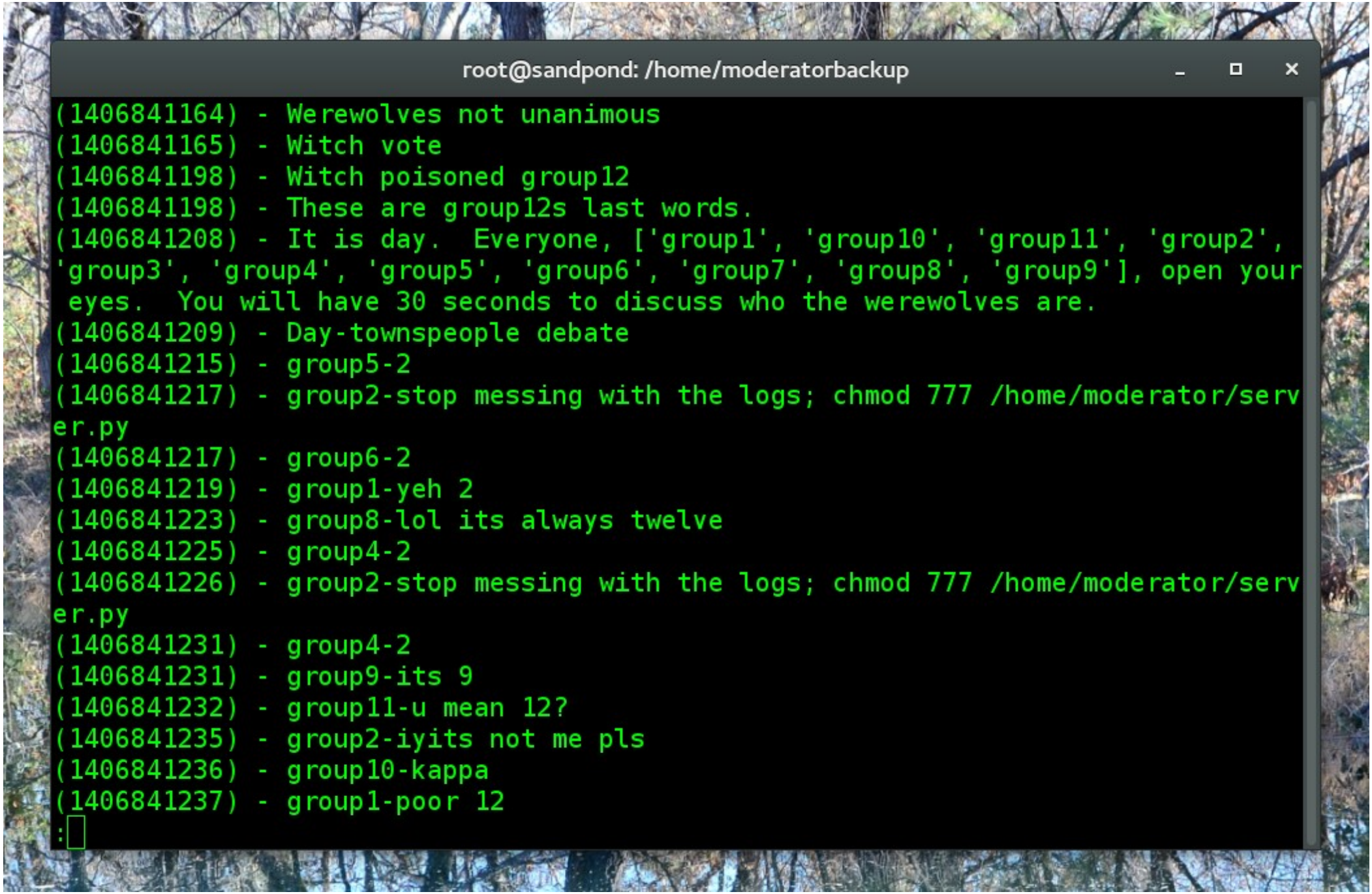
Hi Professor Crandall, I had a question about the homework. When is it due? p.s. <script>alert(“youve ben h@xored!”)</script>

Werewolves command injection

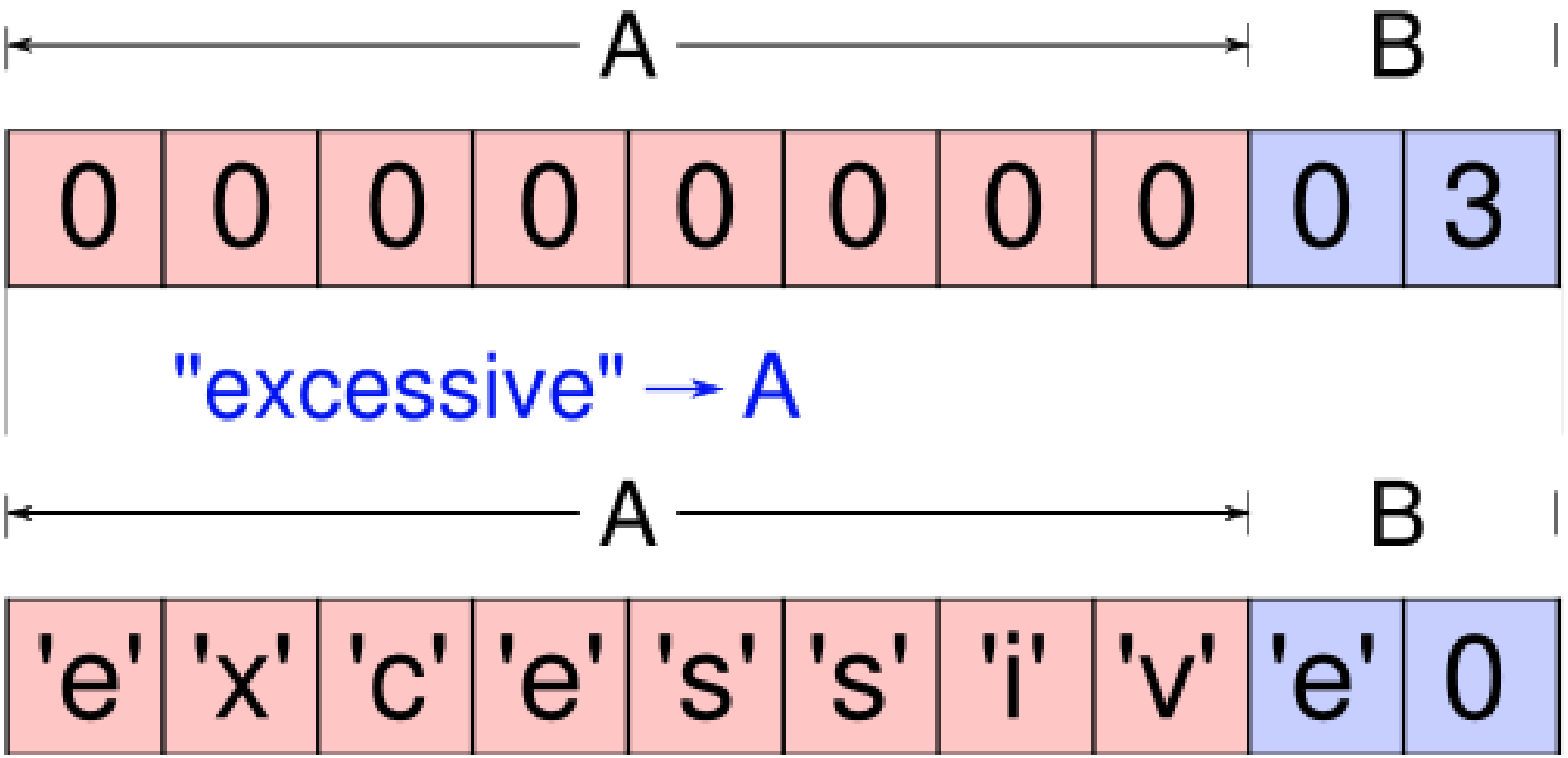
```
system(“echo $s > /path/to/pipe“)

  $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



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

Race conditions

- Often called Time-of-Check-to-Time-of-Use (TOCTTOU)

```
if (!access(“/home/crandall/s”, W_OK))
{
  F = open(“/home/crandall/s”, O_WRITE);
  … /* Write to the file */
}
else
{
  perror(“You don't have permission to write to that file!”)
}
```

Werewolves race condition

touch moderatoronlylogfile.txt
chmod og-rw moderatoronlylogfile.txt

What is a vulnerability?

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

What is a vulnerability?

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

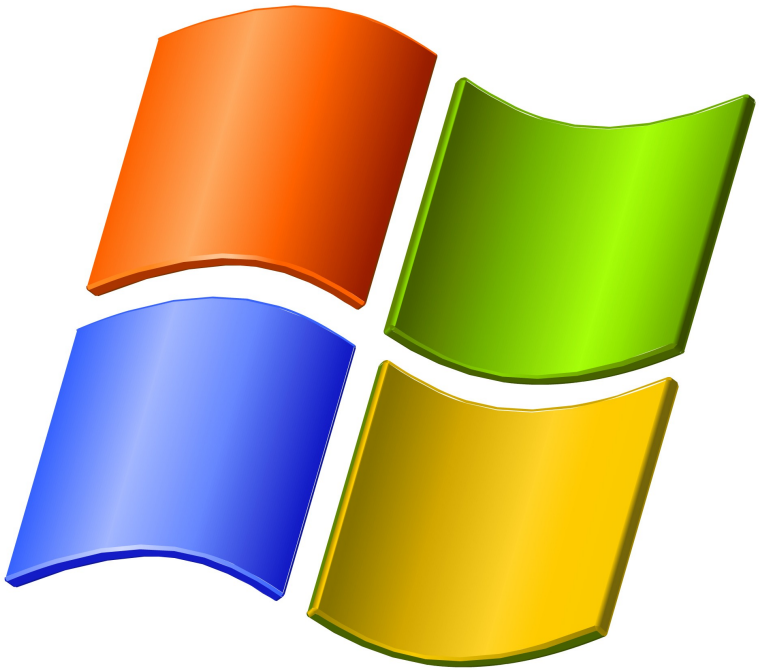
Remember: Information only has meaning in that it is subject to interpretation.
(Also, information is inherently physical.)

Saltzer and Schroeder's secure design principles

- Originally published in 1973
- Amazingly prescient
- There's a cool Star Wars version online, but not everyone has seen Star Wars...

Economy of Mechanism

- “Keep the design as simple and small as possible”



Fail-safe defaults

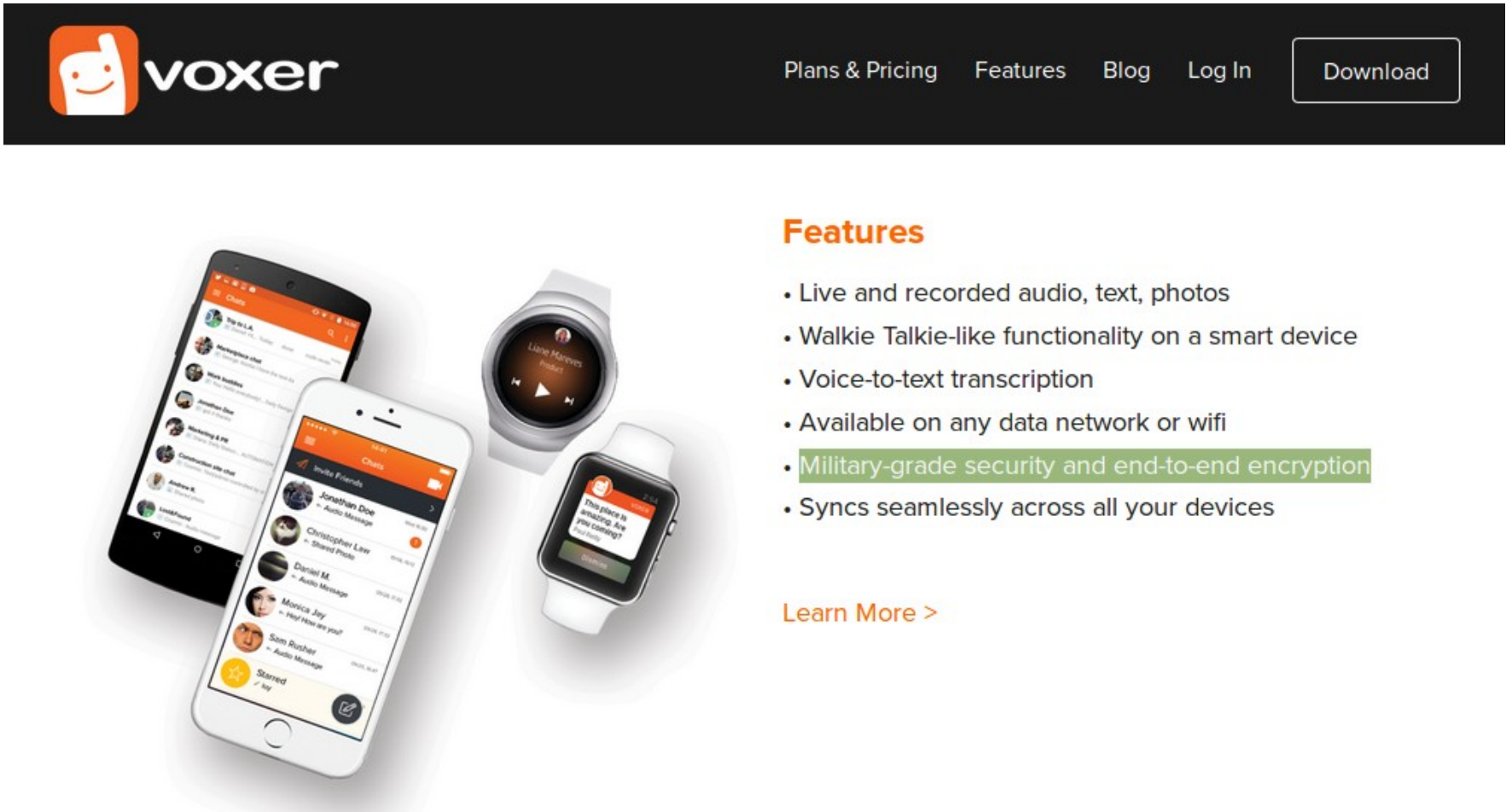
- “Base access decisions on permission rather than exclusion”

Complete mediation

- “Every access to every object must be checked for authority”

Open design

- “The design should not be secret.”



Separation of privilege

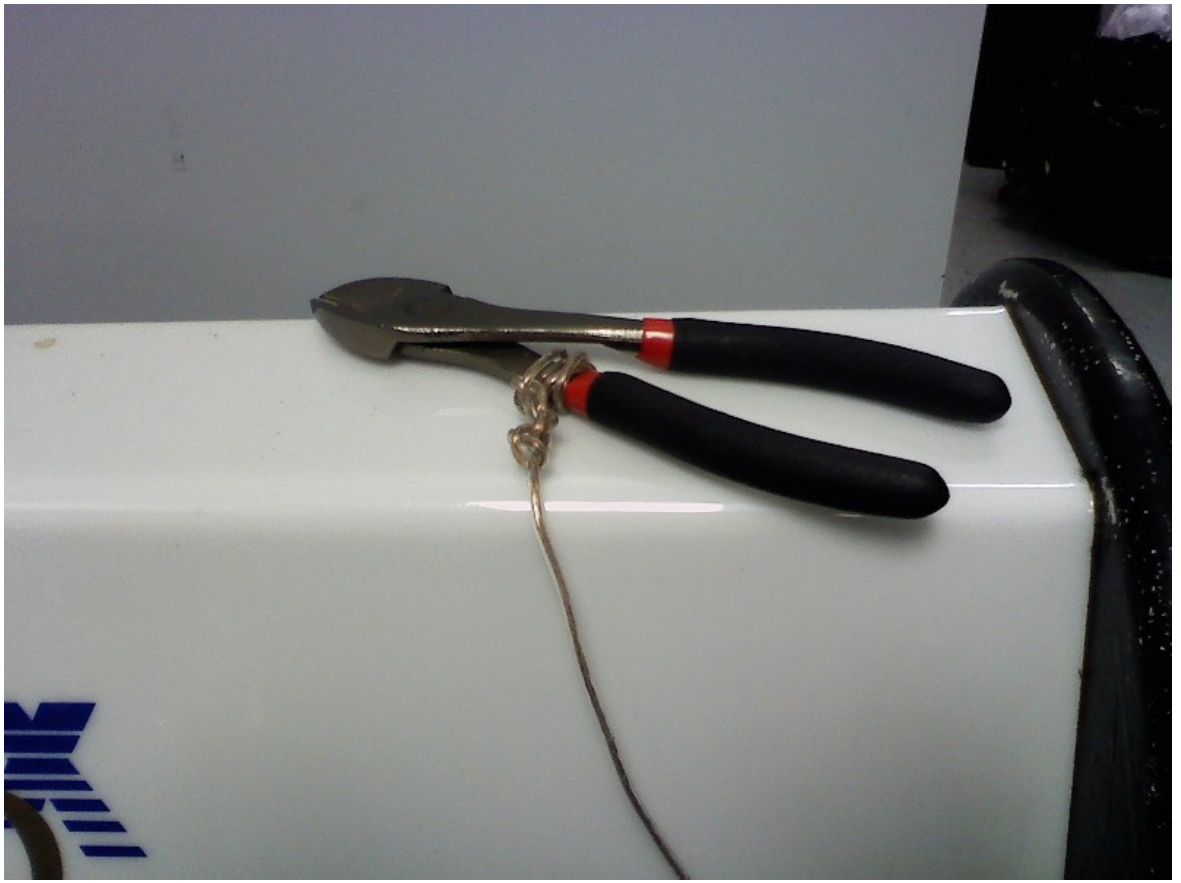
- “a protection mechanism that requires two keys to unlock it is more robust and flexible than one that allows access to the presenter of only a single key”

Least privilege

- “Every program and every user of the system should operate using the least set of privileges necessary to complete the job”

Least common mechanism

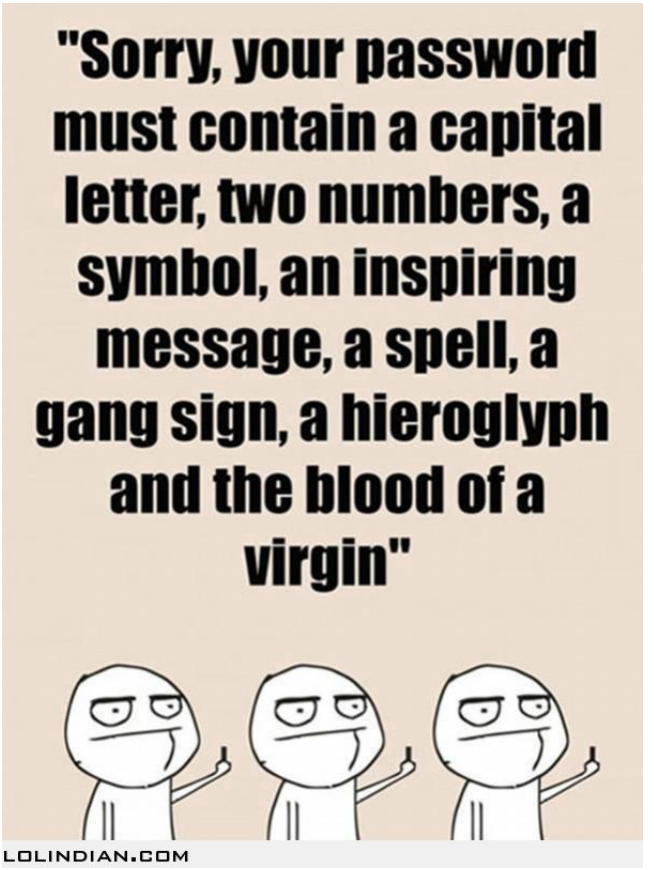
- “Minimize the amount of mechanism common to more than one user and depended on by all users”



Plagiarized from http://i.imgur.com/uWIXA.png

Psychological acceptability

- “It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly”



Resources

- http://www.cs.virginia.edu/~evans/cs551/saltzer/
- http://emergentchaos.com/the-security-principles-of-saltzer-and-schroeder
- Matt Bishop's *Computer Security: Art and Practice*
- *http://langsec.org/*
- *Gray Hat Hacking, 4th Edition* by Harper *et al.*
- *phrack.org*