Authentication, vulnerabilities, exploits, and the secure design principles of Saltzer and Schroeder
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
SHA-512

username

/petc/passwd

/petc/shadow

Salt

password

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.
Plagiarized from https://i.imgur.com/28gf87cbfe2.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

```bash
$ sudo grep "hal" /etc/passwd
hal:x:1003:1003:Hal,,,:/home/hal:/bin/bash
$ sudo grep "hal" /etc/shadow
hal:$6$4asLz5vU$5FDnfwLtlXQf/EEsxB3f3YbjM3fzTtw9EwKY8vsnUE4e8uKlvoy0ST99nqwh5QrHwt3SvGsciQk2D980Q9.:17259:0:99999:7::

$ ls -l /etc/passwd
-rw-r--r-- 1 root root 2021 Apr 2 22:49 /etc/passwd

$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1532 Apr 2 22:49 /etc/shadow
$$
```
Phishing

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

Hi [redacted]

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

[Button: 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

```
$ sudo grep "hal" /etc/passwd
hal:x:1003:1003:Hal,,,:/home/hal:/bin/bash
$ sudo grep "hal" /etc/shadow
hal:$6$4asLz5vU$l5FDnfwLtlXQf/EESsxI3f3YbjM3fzTtw9EwKy8vsnEU4e8uKIvoy0ST99nquwH5QrHwt3SvGsciQk2D980Q9O:17259:0:99999:7:
$ ls -l /etc/passwd
-rw-r--r-- 1 root root 2021 Apr 2 22:49 /etc/passwd
$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1532 Apr 2 22:49 /etc/shadow
```

-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
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 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
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)
- Puring 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'
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

```bash
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
Buffer overflows

"excessive" → A

'e' 'x' 'c' 'e' 's' 's' 'i' 'v' 'e' 0
Format string vulnerabilities

```c
scanf("%s", string)
printf(string)
%500x%500x%12x\xbf\xff\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)

```c
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?

Information only has meaning in that it is subject to interpretation.
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”
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
Resources

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