## Authentication (Recap)

<table>
<thead>
<tr>
<th></th>
<th>Hash (SHA1 SHA256)</th>
<th>MAC Message Authentication Code</th>
<th>Digital signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrity</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Authentication</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Non-repudiation</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Crypto system</td>
<td>None</td>
<td>Symmetric (AES)</td>
<td>Asymmetric (RSA, ECC)</td>
</tr>
</tbody>
</table>
How to authenticate *humans*?
Beyond CIA...

Who are you?

Prove it!

Identification

Authentication

Authorization

Here’s your stuff...

AAA Passwords
Identification

• Humans are generally indistinguishable in front of a computer
• A subject should provide a identifier (e.g. email has to be unique)
• The system will verify if you have the proof to claim an identity
  – This process is called Authentication
Authentication

• **Authentication** is the act of confirming the truth of an attribute of a datum or entity

• There are three authentication factors:
  – **Knowledge**: Something you know
  – **Ownership**: Something you have
  – **Inherence**: Something you are
Knowledge

• Something the user knows (e.g., a password, or PIN, challenge/response (the user must answer a question), pattern)

Strengths
• Easy to transport
• Can be changed

Weaknesses
• Can be forgotten
• Easy to duplicate
• Verifier often learns the secret
Ownership

• Something the user has (e.g., phone number, ID card, security token, etc.)

Strengths

• Easily transferable
• More difficult to clone

Weaknesses

• Can be lost or stolen
• Can be forged
  – e.g. a key can be made from photos
Inherence

• Something the user is or does (e.g., fingerprint, retinal pattern, DNA sequence, voice, etc.).

Strengths

• Non-transferable
• Usually identifies individual

Weaknesses

• Forgeable (ie, fingerprint from picture or from a glass)
• Can be lost (ie, loss or degradation)
• Can’t be changed
Dynamic Authentication with Devices

Time-Varying Codes
(One Time Password)

- Physical/Tamper proof
- Precise clock
- Hash chain inside

Challenge response

- U2F (Universal Second Factor) authentication protocol
- Challenge/response protocol
More authentication factors?

• Location factor
  – Where you are (ie. Gps, Mobile Cell, etc.)

• Ability factor
  – What you can do (ie. Keystroke Dynamics, mouse tracking, etc.)

• ...

• Usually they are classified in the inherence factor, It is an open problem
  – NIST SP 800-63-1
If you need more security?

• Could you use more authentication factor to verify the identity of a user?

— **Multi Factor Authentication** is born

• To increase the level of security, many systems will require a user to provide different types of authentication factor

  2-factor authentication

  • ATM card + PIN
  • Credit card + signature
  • Passport + fingerprint
  • ...
Clicker Question 1

Which multi-factor schema is more secure?

A. ATM Card + Pin + Fingerprint
B. Passport + fingerprint + face
C. Same level of security
D. It is not possible to establish
Clicker Question 1 - Answer

• **Answer: A**
  – It is the only three-factor authentication schema
  B. fingerprint and face
  – they belong both to the ownership authentication factor
  – C, D are not true
Multi-step Authentication

• User submits two or more authentication tokens

• ATM bank card (Two Factor)
  – Physical card (something you have)
  – PIN (something you know)

• Password + code sent to the phone (Two Step) Brown Authentication
  – Enter password (something you know)
  – Enter code (something you know)
Security Questions (aka Personal Questions)

• Questions about the user like city of birth, high school, first car, favorite color, etc.

• Used as supplementary authentication factor
  – When user logs in from new device
  – For password reset

• Answer selection strategies
  – Truthful answers
  – Untruthful but plausible answers
  – Randomly generated answers
Password Authentication
What Do These Passwords Have in Common?

- 123456
- password
- 123456789
- 12345678
- 12345
- 111111
- 1234567q
- sunshine
- qwerty
- iloveyou
- princess
- admin
- welcome
- 666666
- abc123
- football
- 123123
- monkey
- 654321
- !@$%^&*
- charlie
- aa123456
- donald
- Password1
- qwerty123

*Top 25 passwords used in 2018 according to SplashData*
Password Authentication

Client

Username, Password

Authentication Server

Success / Failure
Attacks on Passwords

Information States (MC CUMBER CUBE): Storage, Transmission, Processing
Why send the password?

• “Why not use a MAC instead?”
• Usually, establish a secure channel first with other cryptography
• Allows the server to decide how to hash passwords (which is important for later)
Password Complexity
Characters in Passwords

- Consider a standard US English keyboard
- Lower case characters: 26
- UPPER and lower case: 52
- Digits: 10
- Special characters: 32
- Standard keyboard characters: 94
- All 7-bit ASCII characters: 128
Size of Password Space

• 6-character password
• Only lower case letters, no numbers or symbols

| Char | * | * | * | * | * | * | * | * |
Size of Password Space

- 6-character password
- Only lower case letters, no numbers or symbols
- There are $26 \times 26 \ldots \times 26$ or $26^6 \approx 309M$ possible passwords
- Easy to try all of them

<table>
<thead>
<tr>
<th>Char</th>
<th>*</th>
<th>*</th>
<th>*</th>
<th>*</th>
<th>*</th>
<th>*</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choices</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
</tbody>
</table>
Other Password Schemes

• How many possible 6-character passwords?
  • Digits (10): $10^6$
  • UPPER and lower case (52): $52^6$
  • Special characters: &, %, $, @, “, |, ^, <, ... (32): $32^6$
  • Standard keyboard characters (94): $94^6$
  • All 7-bit ASCII characters (128): $128^6$
## Number of Possible Passwords

Assume a standard keyboard with 94 characters

<table>
<thead>
<tr>
<th>Password length</th>
<th>Number of passwords</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$94^5 = 7,339,040,224$</td>
</tr>
<tr>
<td>6</td>
<td>$94^6 = 689,869,781,056$</td>
</tr>
<tr>
<td>7</td>
<td>$94^7 = 64,847,759,419,264$</td>
</tr>
<tr>
<td>8</td>
<td>$94^8 = 6,095,689,385,410,816$</td>
</tr>
<tr>
<td>9</td>
<td>$94^9 = 572,994,802,228,616,704$</td>
</tr>
</tbody>
</table>
Brown University Password Policy

• Cannot contain your first name, last name, or username
• Cannot match your last three passwords
• Must be at least 10 characters in length
• Must contain at least one lowercase character
• Must contain at least one number
• Must contain at least one special character
• Must contain at least one uppercase character

Source: https://it.brown.edu/information-security/guard-your-privacy/strong-passwords
Strong Passwords

• Long passwords preferred
• Use all available characters
  – UPPER/lower case characters
  – Digits
  – Special characters: &, %, $, £, “, |, ^, §, …

• Which of the following passwords are strong?
  – Seattle1
  – M1ke03
  – P@$w0rd
  – TD2k5s@}ecV87^R:@DKlksj298RLO<j;-*h
Clicker Question 2

Which password policy is more secure?

<table>
<thead>
<tr>
<th>Policy A</th>
<th>Policy B</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 characters total:</td>
<td>8 of any keyboard characters</td>
</tr>
<tr>
<td>• 1 lowercase</td>
<td></td>
</tr>
<tr>
<td>• 1 uppercase</td>
<td></td>
</tr>
<tr>
<td>• 1 digit</td>
<td></td>
</tr>
<tr>
<td>• 1 symbol</td>
<td></td>
</tr>
<tr>
<td>• 4 of any keyboard characters</td>
<td></td>
</tr>
</tbody>
</table>

A. Policy A  
B. Policy B  
C. Both are equally secure  
D. It is not possible to evaluate the security
Clicker Question 2 - Answer

• **Answer: A, B, or C**
  – Depends on the assumptions you make
    • Policy B has bigger password space, though users can pick bad passwords (i.e. `password, 12345678`)
    • What if both policies prevented users from picking common passwords?
Password Complexity In Practice
Password Strength Meter

- *********
  - Password strength: Too short
- *********
  - Password strength: Weak
- *********
  - Password strength: Fair
- *********
  - Password strength: Good
- *********
  - Password strength: Strong

Password must have:
- At least 6 characters
- At least one number
- At least one uppercase letter
- At least one special character

2/8/24 AAA Passwords
History of LUDS

- **Lower- and Uppercase letters, Digits and Symbols**
- Became standard ~ 1985
- US Defense Department - *Password Management Guideline* (Green Book)
  - Guessing space
- NIST - *Password Usage of the Federal Information Processing Standards*
  - Take English words (dictionary) into consideration
# LUDS in Practice

passwordmeter.com

<table>
<thead>
<tr>
<th>Test Your Password</th>
<th>Minimum Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Password:</strong></td>
<td>Minimum 8 characters in length</td>
</tr>
<tr>
<td><strong>Hide:</strong></td>
<td>Contains 3/4 of the following items:</td>
</tr>
<tr>
<td><strong>Score:</strong></td>
<td>- Uppercase Letters</td>
</tr>
<tr>
<td><strong>Complexity:</strong></td>
<td>- Lowercase Letters</td>
</tr>
<tr>
<td></td>
<td>- Numbers</td>
</tr>
<tr>
<td></td>
<td>- Symbols</td>
</tr>
</tbody>
</table>

## Test Results

<table>
<thead>
<tr>
<th>Additions</th>
<th>Type</th>
<th>Rate</th>
<th>Count</th>
<th>Bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Characters</td>
<td>Flat</td>
<td>+(n*4)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Uppercase Letters</td>
<td>Cond/Incr</td>
<td>+((len-n)*2)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lowercase Letters</td>
<td>Cond/Incr</td>
<td>+((len-n)*2)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Numbers</td>
<td>Cond</td>
<td>+(n*4)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Symbols</td>
<td>Flat</td>
<td>+(n*6)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Middle Numbers or Symbols</td>
<td>Flat</td>
<td>+(n*2)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Requirements</td>
<td>Flat</td>
<td>+(n*2)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

## Deductions

<table>
<thead>
<tr>
<th>Deductions</th>
<th>Type</th>
<th>Rate</th>
<th>Count</th>
<th>Bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Letters Only</td>
<td>Flat</td>
<td>-n</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Numbers Only</td>
<td>Flat</td>
<td>-n</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Repeat Characters (Case Insensitive)</td>
<td>Comp</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Problems with LUDS

- Characters frequency is not random
- Frequently used words (password, name)
- Special Dates (Birthday of a relative)
- Keyboard Patterns
- Wrong password strength estimation
  - P@$w0rd1
zxcvbn: realistic password strength estimation

- A model developed by Dropbox
- Easy to adopt
- A rigorous estimation
  - Estimating guess attack directly
- Non-probabilistic
  - Assume attackers know the patterns that make up a password
Through 20 years of effort, we’ve successfully trained everyone to use passwords that are hard for humans to remember, but easy for computers to guess.
What about AI?

- Lakera is an AI security company that has developed an online game
- In this game, you have to try to get the young wizard Gandalf to reveal the password using natural language questions
- Each level requires increasingly complex techniques to deceive the implemented protection mechanisms

gandalf.lakera.ai
Practicing

• You can practice in class with something similar to your real password:
  – passwordmeter.com (LUDS)
  – zxcvbn alternatives to bit.ly/2dp7BD3
    (no more public on dropbox from 9/1/2017):
    • goo.gl/DUSwys (Cygnius)
    • goo.gl/epu13n (Takecontrolbooks)
BREAK!
Storing Passwords
How Should the Server Store Passwords?

• Our goal is to defend from attacks that *exfiltrate* the password database stored by the server
  – Most common password-related attack on server
• We don’t consider other password attacks on the server
  – Eavesdropping passwords submitted by users
  – Modifying the password authentication code
Security mindset: Trust models

• Trust describes a situation in which the security of a system depends on the decisions made by other systems outside of its control.

• Security of system A depends on decisions made by system B
  — “A trusts B”

• Trust ≠ trustworthiness
Attempt #1 - Plaintext

Client

Server

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>$p_1$</td>
</tr>
<tr>
<td>$u_2$</td>
<td>$p_2$</td>
</tr>
<tr>
<td>$u_3$</td>
<td>$p_3$</td>
</tr>
</tbody>
</table>

2/8/24  AAA Passwords
Attempt #1 - Plaintext

• Advantages
  – Easier to manage
  – Less computational needs

• What could go wrong?
  – If database is stolen, so are passwords!
  – Admins have access to passwords.

Attempt #2 - Encryption

- Store encrypted passwords
- Decrypt and compare on login

Server

<table>
<thead>
<tr>
<th>$u_1$</th>
<th>$c_1 = E_k(p_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_2$</td>
<td>$c_2 = E_k(p_2)$</td>
</tr>
<tr>
<td>$u_3$</td>
<td>$c_3 = E_k(p_3)$</td>
</tr>
</tbody>
</table>

2/8/24 AAA Passwords
Attempt #2 - Encryption

• Advantages
  – If encrypted passwords are stolen, they can’t be decrypted
  – Only administrators with key can decrypt

• What could go wrong?
  – If the encrypted passwords are stolen, what is to keep the key from also being stolen?
  – Anyone with the key (admins) can view passwords

• Ex. Adobe (2013)
Attempt #3 - Hashing

Example:

Password -> Hash Function

1337p4Ss -> SHA2

Hash

a487cb0eeb4a484a269c703bce7f8c46b53d4860267a24900ae7ceb577315eb1

AAA Passwords
Attempt #3 - Hashing

Recall cryptographic hashing:

- Variable length input, fixed length “random” output
- One-way
  - Given hash $x$, hard to find $p$ such that $H(p) = x$
- Weak collision resistance
  - Given input $p$, hard to find $q$ such that $H(p) = H(q)$
- Strong collision resistance
  - Hard to find distinct $p, q$ such that $H(p) = H(q)$
Attempt #3 - Hashing

- Hash the password, store the hash
- Hash the user-supplied password and compare

\[
\begin{align*}
  u_1 & \quad d_1 = H(p_1) \\
  u_2 & \quad d_2 = H(p_2) \\
  u_3 & \quad d_3 = H(p_3)
\end{align*}
\]
Attempt #3 - Hashing

• Registration
  – Hash password, store hash

• Login
  – Hash user-supplied password, compare with stored hash

• What advantages does this scheme have?
  – If database is stolen, hashes need to be cracked
    • Correct
  – Cracking must be done brute-force for every password
    • Is this accurate?
Clicker Question 3

**Which hash will you try to decode first?**

| A. fa80328eaf40ecbf22943747d8fe63e3 |
| B. b57bc9ee094db754ca74e034875deb1d |
| C. 35e89a9469522038161a3c173815d8e7 |
| D. d6ed2c6957eb5d5228be0942cf93ea72 |

A. fa80328eaf40ecbf22943747d8fe63e3
B. b57bc9ee094db754ca74e034875deb1d
C. 35e89a9469522038161a3c173815d8e7
D. d6ed2c6957eb5d5228be0942cf93ea72
Clicker Question 3 - Answer

C. fa80328eaf40ecbf22943747d8fe63e3

Identical hash from different users come from the same password and probably it is a common word
• What could go wrong?
  • Identical passwords produce identical hashes
  • Once you’ve cracked a given hash, you can trivially crack it every time you see the same hash again
  • Humans pick bad passwords
    • Frequency analysis
    • Precompute massive tables for popular hash functions
    • Common passwords are very common!
    • Even a small table cracks most passwords
    • Updated version
Clicker Question 4

Mallory steals a database of encrypted passwords (but cannot steal the key). Could she recover the plaintext passwords?

A. Yes, all of them
B. Yes, a fraction of them
C. No, since the database is encrypted
D. No, since it is computationally infeasible
Clicker Question 4 - Answer

• **Answer: B**
  – Identical passwords produce identical ciphertexts
  – If you know one password, you know all passwords same ciphertext
  – Humans pick bad passwords and hints
    • Frequency analysis (0.5% of users use password)
    • Password hints (e.g., numbers 123456)
  – Unique passwords with good hints are safe
Attempt #4 - Salting

- **Password**: 
- **Salt (Random)**: 
- **SHA2**: 
  - 1337p4Ss 
  - fA4dY5 
- **Hash Function**: 
  - Hash (SHA2): 
    - 35e89a9469522038161a3c173815d8e7d6ed2c6957eb5d5228be0942cf93ea72

2/8/24
AAA Passwords 58
Attempt #4 - Salting

Client → Server

\[
\begin{align*}
  u_1 & \quad d_1 = H(p_1, s) \\
  u_2 & \quad d_2 = H(p_2, s) \\
  u_3 & \quad d_3 = H(p_3, s)
\end{align*}
\]
Attempt #4 - Salting

- Store hash of salted password
- Hash the password and salt, then compare
- Advantages
  - In order to precompute, need password and salt
  - Since salts are random, guessing salt is useless
  - Even if salt is known, computation must be redone for every site
Clicker Question 5

Using hashing and salting to store passwords, the server successfully defends over frequency analysis attacks.

A. True.
B. False.
C. Not enough information.
Clicker Question 5 - Answer

Using hashing and salting to store passwords, the server successfully defends over frequency analysis attacks.

A. True.
B. True.
C. Not enough information.
Attempt #4 - Salting

● What could go wrong?
  ○ Identical passwords and identical salts produce identical hashes
  ○ Humans pick bad passwords --> frequency analysis
  ○ If you crack one password, you crack all identical ones
  ○ For big sites, precomputation is worth it
Attempt #5 – Per-User Salting

- Hashing same password with different salt will produce different hashes

1337p4Ss

4themm

SHA2

fa80328eaf40ecbf22943747d8fe63e3b57bc9ee094db754ca74e034875deb1d

1337p4Ss

rsthnks

SHA2

2674d6e9c0c1f5ea3235cafccac433a30a9de88ddfa0c2e044b53daa63c8afdd
Attempt #5 - Per-User Salting

- Generate a salt, hash the password, store salt and hash
- Hash the given password with the user’s salt and compare

\[
\begin{align*}
\text{Server} \\
\begin{array}{|c|c|}
\hline
u_1 & s_1 \quad d_1 = H(p_1, s_1) \\
\hline
u_2 & s_2 \quad d_2 = H(p_2, s_2) \\
\hline
u_3 & s_3 \quad d_3 = H(p_3, s_3) \\
\hline
\end{array}
\end{align*}
\]
Attempt #5 - Per-User Salting

- Generate a salt, hash the password, store the hash
- Hash the given password with the user’s salt and compare
- Advantages
  - Since every user has different salt, identical passwords will not have identical hashes
  - No frequency analysis
  - No using known passwords to crack other passwords
  - No precomputation, hence much harder to crack
Summary

• Human Authentication
• AAA Identification, Authentication, Authorization, Accounting
• Password
  – Authentication
  – Complexity
  – Storage (salting)