Authentication

Andy Podgurski
EECS Dept.
Case Western Reserve University

Definition & Purpose

- *Authentication is the process used by a system to confirm that a subject's claimed identity is its actual identity*
- It is needed to enforce security policies based on identity
  - e.g., access control policies

Qualities Used in Authentication

Authentication mechanisms use any of three qualities to confirm a user's identity:

- Something the user *knows*
  - e.g., password
- Something the user *is*
  - e.g., fingerprint
- Something the user *has*
  - e.g., ID card

Passwords

- User enters name or assigned user ID
- Protection system requests password
- If supplied password matches one on file for the user, they are authenticated

Attacking Passwords

- People often choose poor passwords, e.g.,
  - Same as user ID
  - Derived from person's name
  - Word in dictionary
  - Alphabetic characters only
- Attackers exploit this to guess passwords

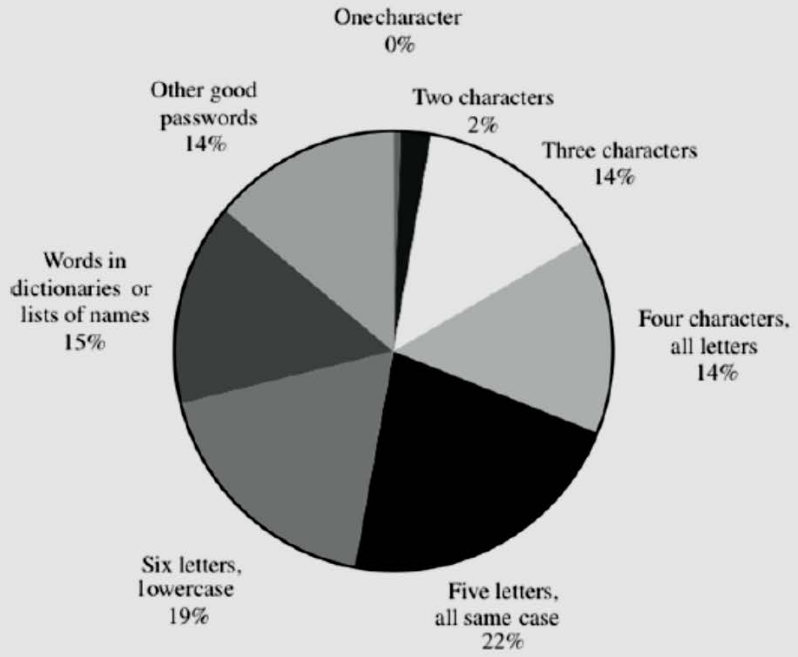


FIGURE 2-1 Distribution of Password Types

RockYou Password Leak

The top 20 passwords of 32 million

Rank	password	total
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542

Rank	password	total
11	Nicole	17168
12	Daniel	16409
13	babygirl	16094
14	monkey	15294
15	Jessica	15162
16	Lovely	14950
17	michael	14898
18	Ashley	14329
19	654321	13984
20	Qwerty	13856

Imperva (2010). Consumer Password Worst Practices

Guidelines for Passwords

- Use as many kinds of characters as possible
- Choose long passwords
- Avoid names or words
- Use variants for multiple passwords
- Change the password regularly
- Don’t write it down (if practical)
- Don’t disclose it to others

Password Hashing

1. The user creates an account.
2. Their password is hashed using a *cryptographic hash function* and stored in a database.
3. When the user attempts to login, the hash of the password they entered is checked against the stored hash of their real password.
4. If the hashes match, the user is granted access. If not, the user is told they entered invalid login credentials.
5. Steps 3 and 4 are repeated each time someone tries to login to their account.

Cracking Hashes

Assume file of hashed passwords is stolen.

- *Dictionary attack*
  - Uses a file containing words, phrases, common passwords, etc.
  - Each word in the file is hashed, and its hash is compared to the password hash
- *Brute-force attack*
  - Tries every possible combination of characters up to a given length
- *Lookup tables*
  - Pre-compute hashes of passwords in PW dictionary
  - Store them, and their corresponding password, in a lookup table

Adding Salt

- Randomizes the hash by appending a *random string*, called a *salt*, to the password before hashing
- This makes the same password hash into a completely different string each time
- To check if a password is correct, the salt is needed
- It is usually stored in the user account database along with the hash, or as part of the hash string itself
- It's *not feasible* for attackers to store tables of precomputed hash values for common passwords

Example: Hashing with Salt

```
hash("hello") =
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824

hash("hello" + "QxLUF1bgIAdeQX") =
9e209040c863f84a31e719795b2577523954739fe5ed3b58a75cff2127075ed1

hash("hello" + "bv5PehSMfV11Cd") =
d1d3ec2e6f20fd420d50e2642992841d8338a314b8ea157c9e18477aaef226ab

hash("hello" + "YYLmfY6IehjZMQ") =
a49670c3c18b9e079b9cfaf51634f563dc8ae3070db2c4a8544305df1b60f007
```

Adding Salt cont.

To Store a Password

1. Generate a long random salt using a *cryptographically secure pseudorandom number generator*.
2. Append the salt to the password and hash it with a standard cryptographic hash function such as SHA256.
3. Save both the salt and the hash in the user's database record.

To Validate a Password

1. Retrieve the user's salt and hash from the database.
2. Append the salt to the given password and hash it using the same hash function.
3. Compare the hash of the given password with the hash from the database. If they match, the password is correct. Otherwise, the password is incorrect.

Graphical Passwords

- Images are *easier to remember* than strings
- Visual memory has *large capacity* and *good retention*
- Humans remember the "gist" of images quickly
- *Heterogeneous* images needed

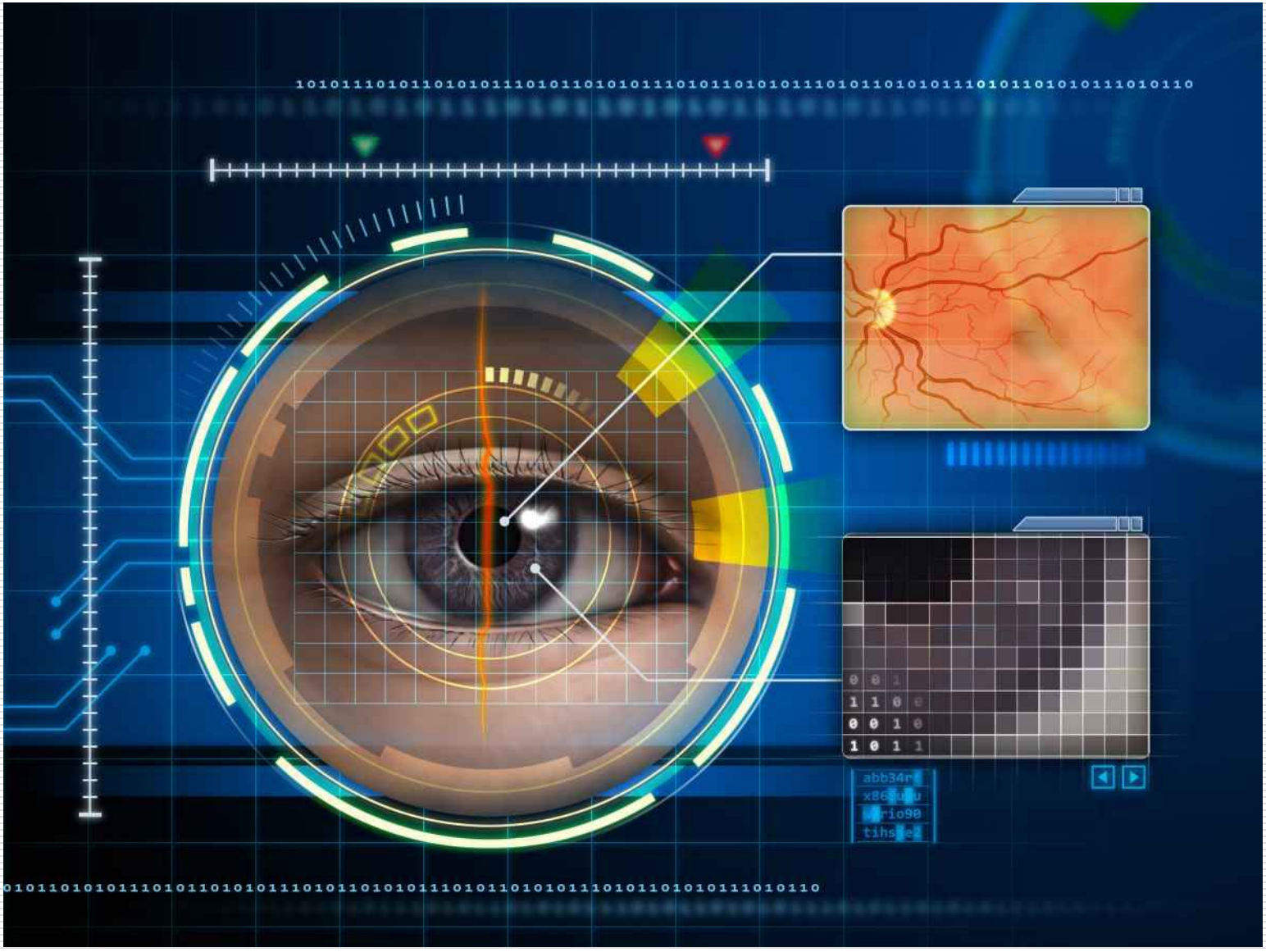
VIP (De Angeli et al., 2005)
"select the images from your password set"

Personal Security Questions

- Ideally, user should be *only* person to know answer
  - *What is the name of your first girlfriend/boyfriend?*
- Someone who knows you or can do research about you is much more likely to know or guess the answers

Biometrics

- Biological properties, based on physical characteristics of user, e.g.,
  - Fingerprint
  - Hand geometry
  - Retina and iris patterns
  - Handwriting, signature, hand motion
  - Voice
  - Blood vessels in finger or hand
  - Facial features



http://www.popsci.com/sites/popsci.com/files/styles/medium_1x_/public/images/2014/12/biometric-eyes.jpg?itok=WmuC7xw0

Problems with Biometrics

- May be consider *intrusive*
- Relatively *costly*
- Reader may be *single point of failure*
- Based on *sampling* and *thresholds*
- *False readings* occur
- Some people are more likely to pass, some are more likely to fail
- Relatively *slow*
- *Forgeries* possible

Authentication Based on Tokens

- Token is a physical object in your possession
  - e.g., badge, ID card, dongle

Types of Tokens

- *Passive token*: contents never change
  - e.g., photo
- *Active token*: has some variability or interaction with environment
  - e.g., card with writeable magnetic strip
- *Static token*: value remains fixed
  - e.g. non-writeable magnetic strip cards
- *Dynamic tokens*: have computing power on token
  - e.g., SecureID token from RSA generates random number each minute

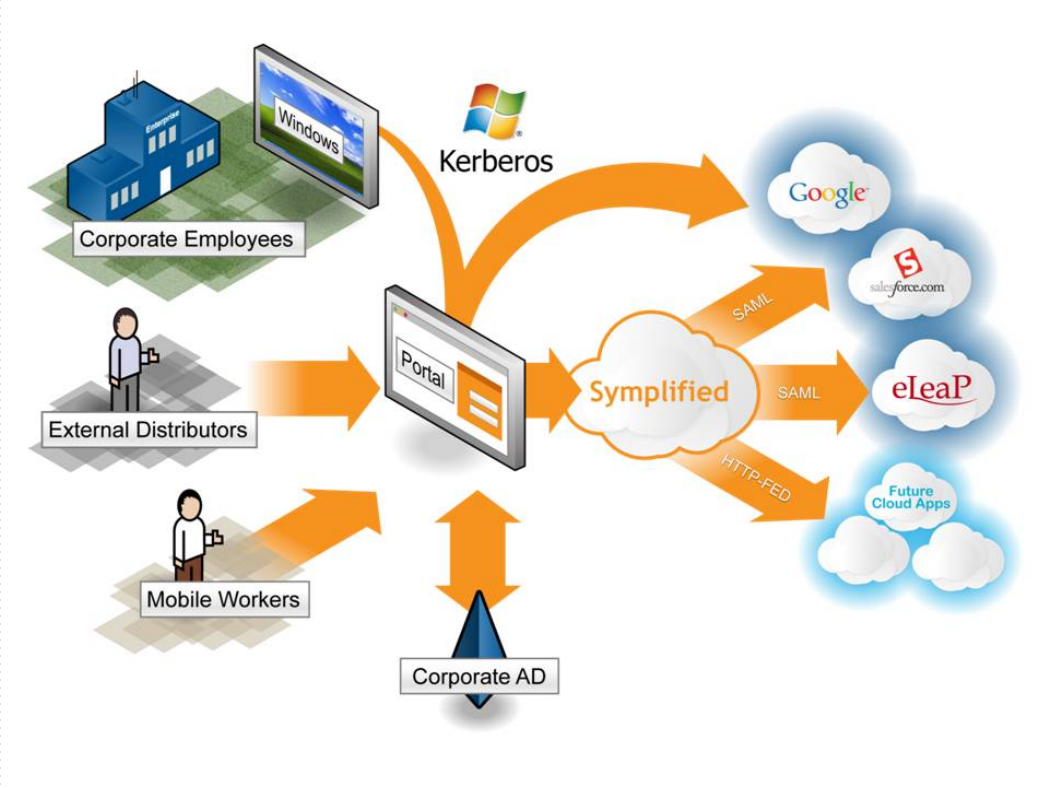
Skimming

- Use of device to *illicitly read* authentication data
  - e.g., at ATM or credit card reader
- Dynamic tokens help prevent skimming
  - Skimmed information is not sufficient

Federated Identity Management

- *FIM*: common policies, practices and protocols used to manage identity and trust across organizations
- *Single sign-on*: one authentication process and credentials used across multiple systems

Federated Identity Management cont.



wikibon.org/wiki/v/Identity_Management_Pays-Off

“Risk-Based” Authentication

- Uses *statistics* or *machine learning* for authentication
- Authenticator considers *multiple factors*
  - e.g., password, IP address, location, browser information, time of login, typing patterns
- *Progressive authentication* may require more information if confidence is low

Sources

- J. Bonneau et al., *Passwords and the evolution of imperfect authentication*, CACM 58, 7, July 2015
- Defuse Security, *Salted Password Hashing – Doing it Right*, crackstation.net/hashing-security.htm
- S. Marechal, *Advances in password cracking*, Journal of Computer Virology 4, 2008
- C.P. Pfleeger et al., *Security in Computing*, 5th ed, Prentice-Hall
- D. Shinder, *Understanding and selecting authentication methods*, www.techrepublic.com, 2001
- X. Suo et al, *Graphical passwords: a survey*, 21st Annual Computer Security Applications Conference (ACSAC 2005)