User Authentication

Paweł Szałachowski

Authentication vs Authorization

- **Authentication**

  - is the act of confirming the truth of an attribute of a single piece of data claimed true by an entity

    - In the identity context, it is the process of confirming the claimed identity

- Authorization

  - is the function of specifying access rights/privileges to resources related to information security and computer security in general and to access control in particular

Authentication

- When?

    - Logging into local or remote PC, network, or website

- How?

    - Basing on user's knowledge: passwords

    - Basing on user's possession: smart card or tokens

    - Basing on user: biometric

- Why it is so difficult to realize secure user authentication?

Passwords

- Passwords

  - passphrase, passcode, PIN, …

  - Easy do deploy, customize, use, replace, ..

- "Passwords will be obsolete next year."

  - You hear that every year…

- Authentication via username&password pair

- A lot of recent research

Password Lifetime

- Bootstrapping (information shared initially with server)

  - username, password

  - Optional: e-mail, recovery question(s), 2FA, …

- Authentication

  - Server checks if sent (username, password) pair matches

- Recovery

  - Update password after forgetting it

  - Critical point (usually, the weakest link)

    - security = min(password_security, recovery_security)

Password Storage

- Servers store passwords in cleartext

  - username: password (/etc/passwd in old Unix systems and many new systems!)

  - Server checks whether username and password match

- Adversary model

  - Able to compromise storage (e.g., malicious admin)

    - Can just read **all** passwords

- How to mitigate such an adversary?
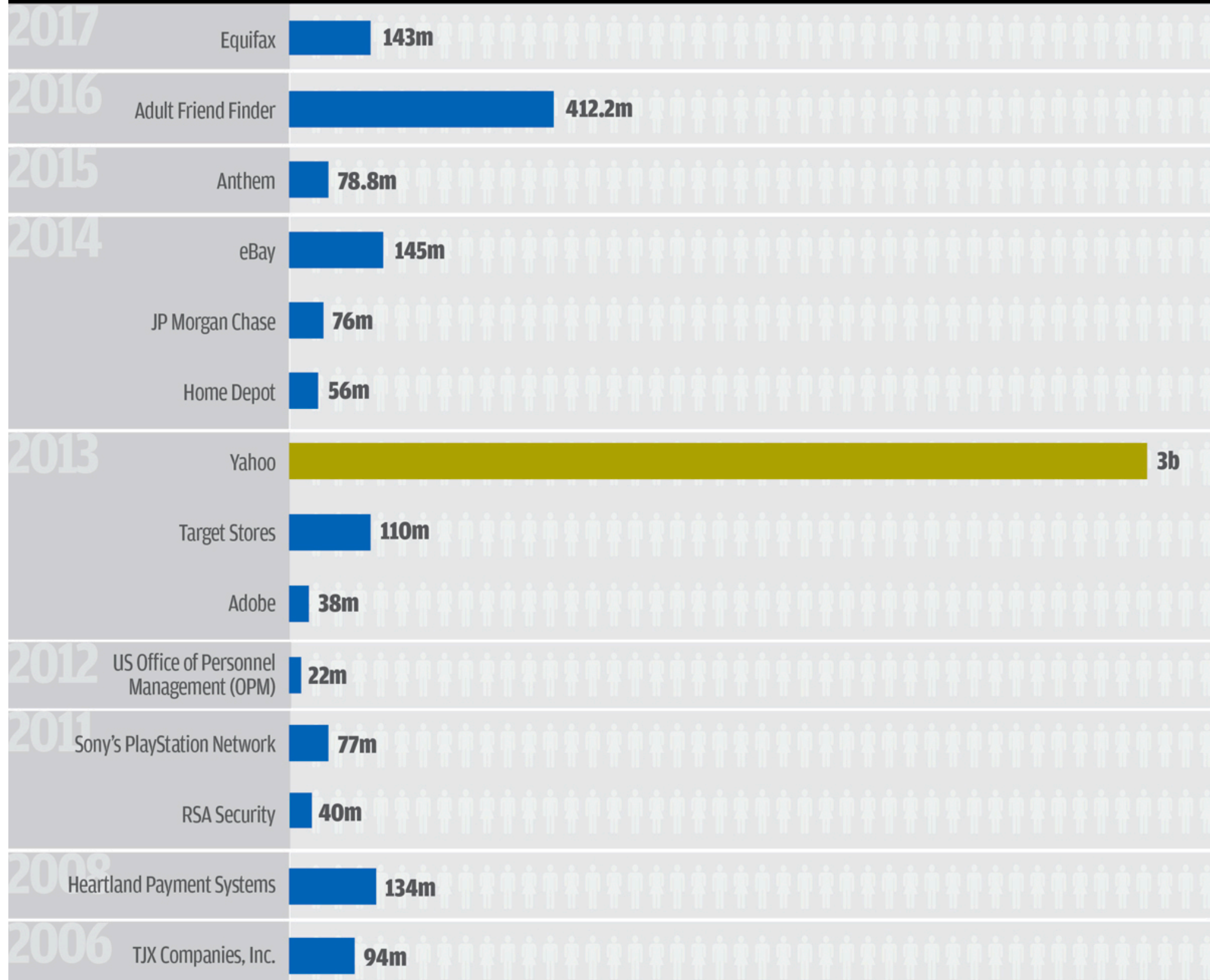
Biggest DATA BREACHES of the 21st century

Accounts Compromised — by the millions — by the billions

Year	Company	Accounts
2017	Equifax	143m
2016	Adult Friend Finder	412.2m
2015	Anthem	78.8m
2014	eBay	145m
2014	JP Morgan Chase	76m
2014	Home Depot	56m
2013	Yahoo	3b
2013	Target Stores	110m
2013	Adobe	38m
2012	US Office of Personnel Management (OPM)	22m
2011	Sony's PlayStation Network	77m
2011	RSA Security	40m
2008	Heartland Payment Systems	134m
2006	TJX Companies, Inc.	94m

SOURCE: CSO

Password Storage

- Use one-way function (e.g., hash) to store passwords

  - username : H(password)

  - Server hashes sent password and checks whether the username and the hash matches

- Adversary stealing password database cannot read passwords (only their hashes are visible)

Dictionary Attack

- Dictionary Attack

    1. Steal database of usernames and password hashes

    2. Prepare list of common password (e.g., English dictionary)

    3. For each word compute H(word)

    4. Compare H(word) with stolen hashes

- Effective dictionaries are huge

    - Servers require "hard" passwords (length, digits, special characters, …)

   - That introduces significant overhead

Rainbow Tables

- Space-time tradeoff

- Observation: passwords sets are limited

  - e.g., 6 characters from: a-z, A-Z, 0-9, ~!@#$%^&…

  - bytes like "\x00", whitespaces, etc… are not used

- How to launch the dictionary attack w/o precomputing such a dictionary?

- Idea: from one word generate hash, from which generate the next word, …

  - Hash function (H) generates a pseudorandom bytestream

  - *Reduction function* (R) reduces bytestream to a word from the password set.

$$\text{aaaaaa} \xrightarrow{H} \text{281DAF40} \xrightarrow{R} \text{sgfnyd} \xrightarrow{H} \text{920ECF10} \xrightarrow{R} \text{kiebgt}$$

Rainbow Tables

- Rainbow Table Creation

  - Generate a random set of initial passwords

  - For each password compute n-long password chain (applying hash and reduction functions alternately)

  - Store only the first and the last passwords of each chain

    - These values are called startpoint and endpoint, respectively

- Password cracking (with known hash h)

  - Keep computing (up to n times): R(h), H(R(h)), R(H(R(h))), H(R(H(R(h)))) …

  - With each reduction check whether its outputs is an endpoint in the table

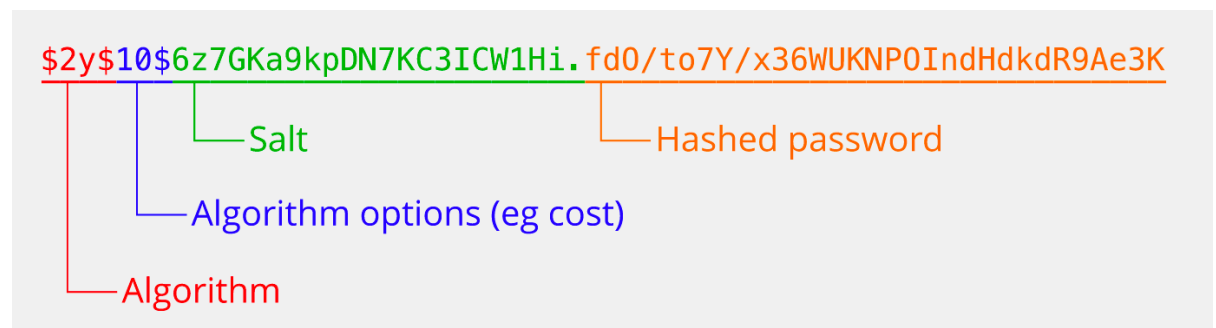   - If so, start recreating the chain to find h

Example: Rainbow Tables
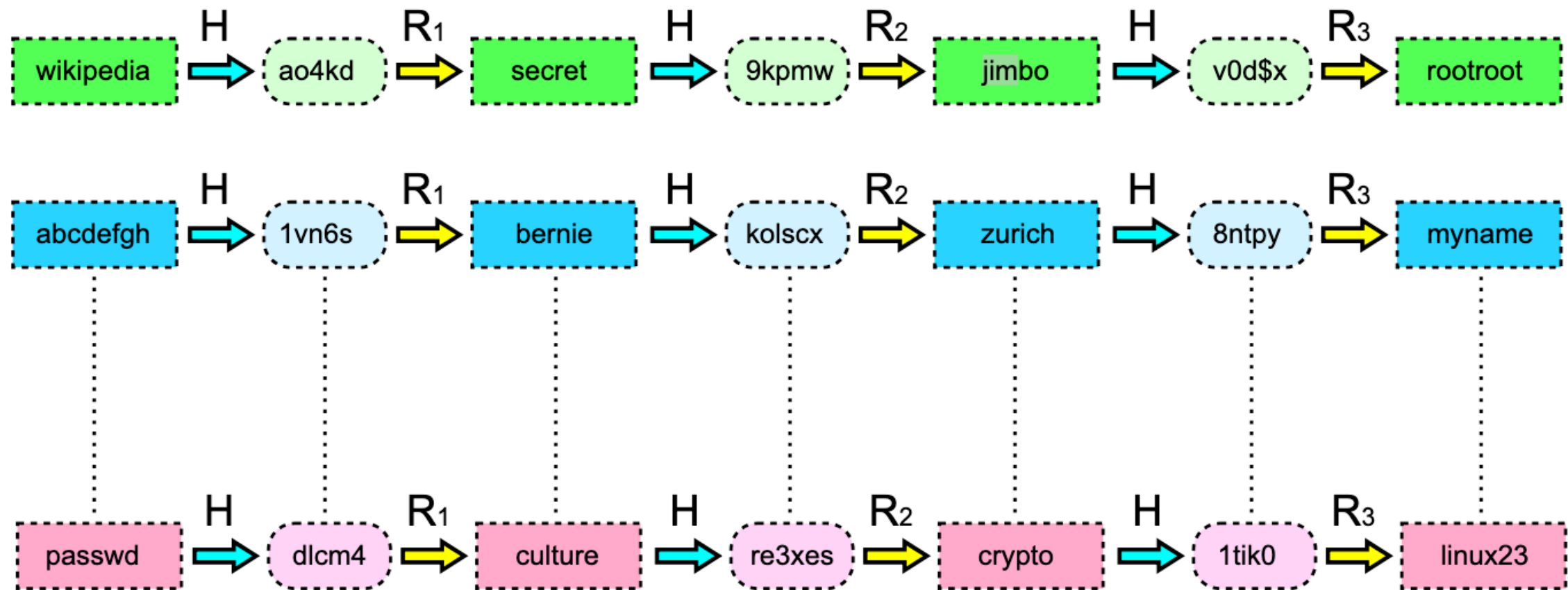
- What is the rainbow table generated?

- Find password (preimage) corresponding to "re3xes"

Salted Passwords

- Dictionary-like attacks are effective as single hash value can be compared against many pre-computed values

  - Imagine one huge dictionary pre-computed for given hash function

- Idea: randomize password hashing by random value (salt)

  - username : salt, H(salt, password)

  - Salt is random and unique per username

- The same passwords will have different hashes

- Pre-computed dictionaries are useless

  - Dictionary has to be pre-computed per salt (what is inefficient)

Password Cracking

- Boils down to fast hashing

    - GPUs, ASICS, … (Bitcoin helped a lot:-)

- PBKDF2 (Password-Based Key Derivation Function 2)

    - Repeat hashing to get the final hash

        - Sliding computational cost (can be adjusted)

- bcrypt

    - Used by OpenBSD and some Linux distributions

- scrypt

    - Memory-expensive hashing

```
$2y$10$6z7GKa9kpDN7KC3ICW1Hi.fdO/to7Y/x36WUKNPOIndHdkdR9Ae3K
```
Salt — Hashed password
Algorithm options (eg cost)
Algorithm

14

Password Breach Detection

- How to detect password breaches?

  - Admins, operators, (successful) hackers, or hosting companies have access to filesystems

    - Some attacks are very difficult to detect

      - Password files have to be writeable/readable by superusers and some processes

  - With a stolen password file an adversary can start offline attacks (e.g., the dictionary attack)

- Observation: if an adversary is able to crack a number of passwords, these are then used to log in (for identity or information theft, malicious activities, …)

  - What if some of these passwords are decoy passwords?

HoneyWords

- Analogy with HoneyPot systems

- Each user has its username, *real* password, and a list of *decoy* passwords

  - Server stores passwords salted and hashed

  - Real and decoy passwords are indistinguishable

- HoneyChecker is a secured and isolated module that knows real passwords (server has connection with its HoneyChecker)

1. A user sends to a server its password (to authenticate)

2. Server checks whether the password is listed and if so asks the HoneyChecker

3. If the password is correct the HoneyChecker returns YES, otherwise NO

  - An alarm can be raised and the user's account can be blocked.

Passwords
Selection and Policies

Top 25 most common passwords by year according to SplashData

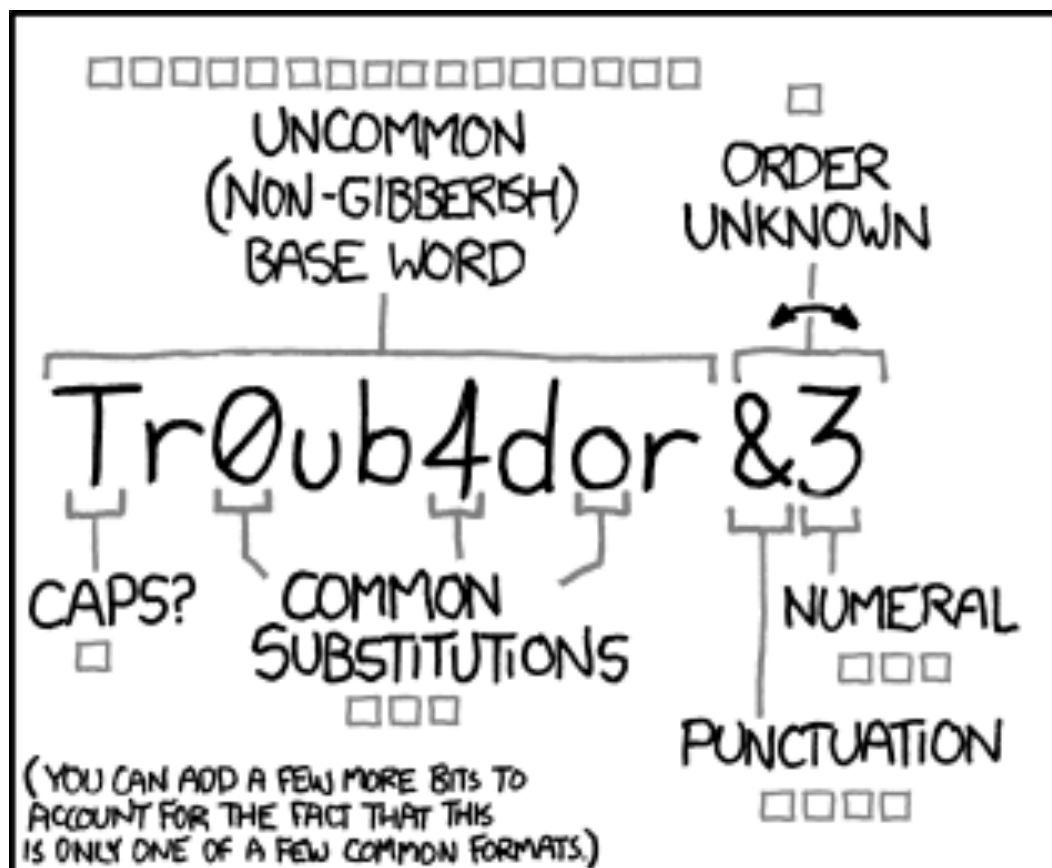| Rank	2011[4]	2012[5]	2013[6]	2014[7]	2015[8]	2016[3]	2017[9]
1	password	password	123456	123456	123456	123456	123456
2	123456	123456	password	password	password	password	password
3	12345678	12345678	12345678	12345	12345678	12345	12345678
4	qwerty	abc123	qwerty	12345678	qwerty	12345678	qwerty
5	abc123	qwerty	abc123	qwerty	12345	football	12345
6	monkey	monkey	123456789	123456789	123456789	qwerty	123456789
7	1234567	letmein	111111	1234	football	1234567890	letmein
8	letmein	dragon	1234567	baseball	1234	1234567	1234567
9	trustno1	111111	iloveyou	dragon	1234567	princess	football
10	dragon	baseball	adobe123[a]	football	baseball	1234	iloveyou
11	baseball	iloveyou	123123	1234567	welcome	login	admin
12	111111	trustno1	admin	monkey	1234567890	welcome	welcome
13	iloveyou	1234567	1234567890	letmein	abc123	solo	monkey
14	master	sunshine	letmein	abc123	111111	abc123	login
15	sunshine	master	photoshop[a]	111111	1qaz2wsx	admin	abc123
16	ashley	123123	1234	mustang	dragon	121212	starwars
17	bailey	welcome	monkey	access	master	flower	123123
18	passw0rd	shadow	shadow	shadow	monkey	passw0rd	dragon
19	shadow	ashley	sunshine	master	letmein	dragon	passw0rd
20	123123	football	12345	michael	login	sunshine	master
21	654321	jesus	password1	superman	princess	master	hello
22	superman	michael	princess	696969	qwertyuiop	hottie	freedom
23	qazwsx	ninja	azerty	123123	solo	loveme	whatever
24	michael	mustang	trustno1	batman	passw0rd	zaq1zaq1	qazwsx
25	Football	password1	000000	trustno1	starwars	password1	trustno1

Weak Password

- Default passwords: password, default, admin, guest, …

- Dictionary words: chameleon, RedSox, sandbags, bunnyhop!, IntenseCrabtree, …

- Words with numbers appended: password1, deer2000, john1234, …

- Words with simple obfuscation: p@ssw0rd, l33th4x0r, g0ldf1sh, …

- Doubled words: crabcrab, stopstop, treetree, passpass, …

- Common sequences from a keyboard row: qwerty, 12345, asdfgh, fred, etc.

- Numeric sequences based on well known numbers such as 911 (9-1-1, 9/11), 314159... (pi)

- Identifiers: jsmith123, 1/1/1970, 555–1234, one's username, …

- Anything personally related to an individual: license plate number, Social Security number,…

Good (high-entropy) Passwords

- vJI6xft.CNqG295Fvu9B3w/5cY

  - It is a pretty good password… except you will forget it

- Sentence to password

  - I really like the "51.502: Systems Security" course!

  - Irlt"51.502:SS"c!

- Few random words

  - CorrectHorseBatteryStaple

- Few random words with tweaks (separators, upper cases, …)

  - --31_throw_SIGNAL_march_74--
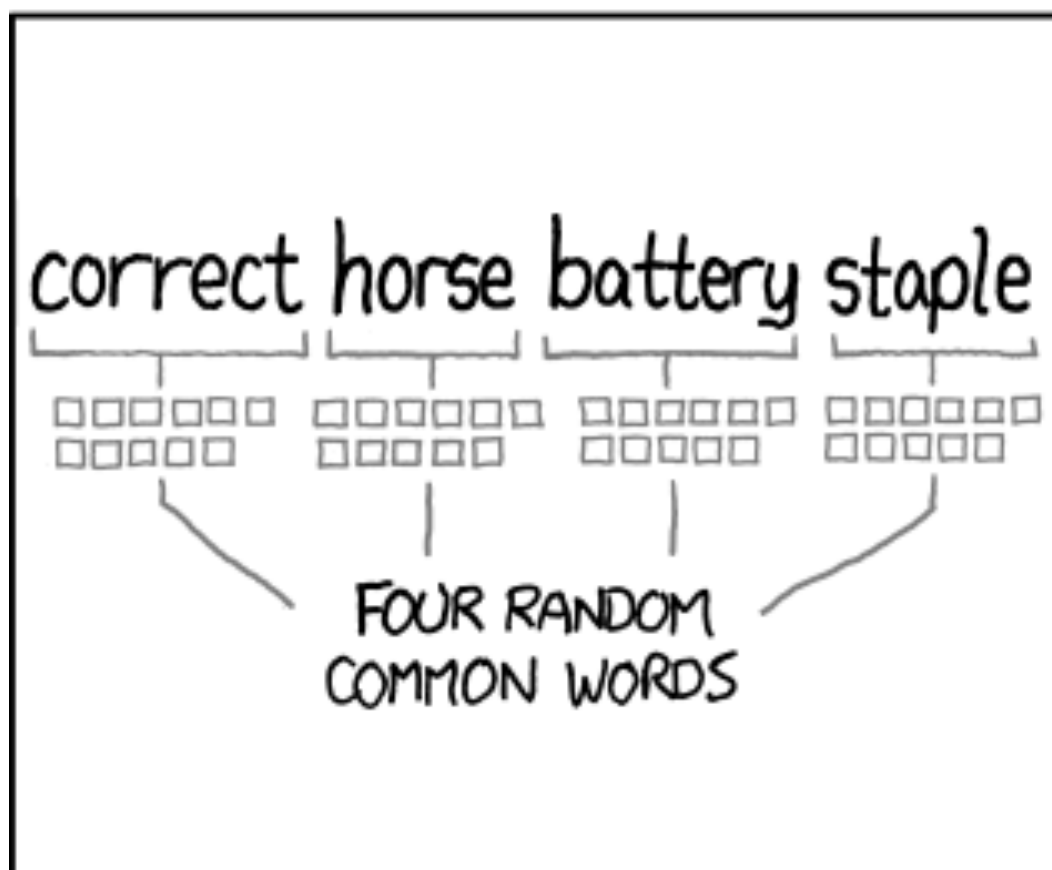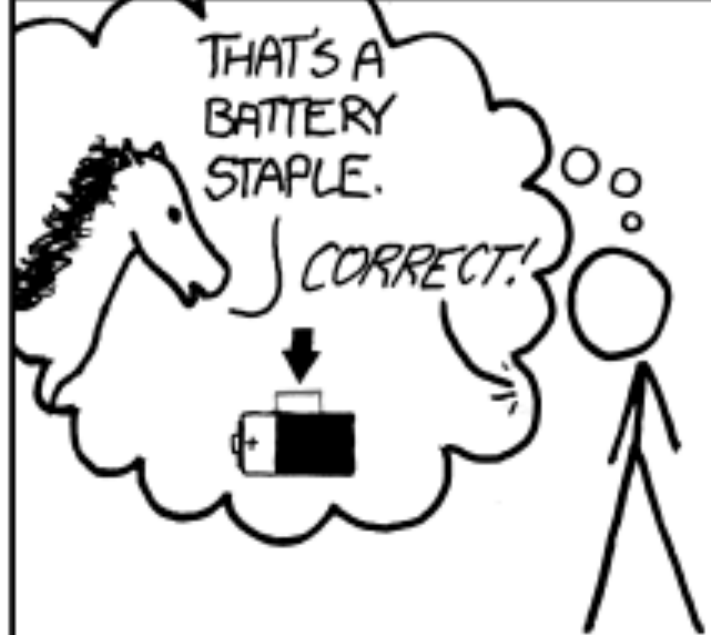
Through 20 years of effort, we've successfully trained everyone to use passwords that are hard for humans to remember, but easy for computers to guess.

Password Policies

- Format restriction

  - length, lower/upper case, special characters, digits

  - Forbidden words (from dictionaries, frequent passwords, …)

  - Forbidden known formats (calendar dates, phone numbers, …)

- Blacklist

  - "good" passwords could be blacklisted too (e.g., if revealed previously)

- Expiration

  - Users are forced to change passwords every X days

  - Cannot reuse old passwords

  - Benefits of refreshing passwords are questioned

Case Study

- staffselfhelp.sutd.edu.sg

  - 8 characters, >0 upper case, >0 digit, >0 special

  - Expire every 90 days

  - Cannot reuse any of 6 last passwords

One-Time Password (OTP)

- Lamport's OTP

  - For random s Alice computes H(s), H(H(s)), H(H(H(s))), … $H^{1000}(s)$

  - Alice bootstraps server with $H^{1000}(s)$

  - To authenticate 1st session Alice reveals $H^{999}(s)$. For the 2nd $H^{998}(s)$…

- Disadvantages?

OTPs

- Time-based OTPs

  - Alice bootstraps server with K

  - To authenticate at time T Alice sends H(K, T)

- Counter-based OTPs

  - Alice bootstraps server with K, C is set to 0

  - For new session, Alice authenticates sending H(K, C++)

- Which ones are *better*?

OTPs

- Challenge-Response (Nonce-based) OTPs

  - Alice bootstraps server with K

  - For every session, server sends Nonce

    - Alice authenticates sending H(K, Nonce)

Two-Factor Authentication

- Protection against stolen/cracked passwords

  - Introduce second factor to authentication

- Passwords + OTP

  - To authenticate basing on both: user's knowledge and possession

  - OTP implemented as physical tokens, Apps, …

- Usability

  - Requiring it for every interaction may be annoying

- Password + Device (App)

Alternatives

- Certificates

  - Usability issues (certificates are difficult to manage)

- Biometrics

  - E.g., your fingerprint

    - Your ID, unique, convenient to carry and use

    - Rather local than remote authentication (hardware required, etc …)

      - Replay attacks

- Passwordless Login Links

- Visual Passwords

Misc

- Password managers

  - Remember one strong password

    - Usually, used for en/decrypting other passwords

      - Encrypted passwords can be outsourced (e.g., to cloud)

    - Popular but centralized (one master password)

- Password transmission

  - Always use encryption (otherwise passwords are visible)

  - Zero-knowledge Password Proof (e.g., the SRP protocol)

Reading

- [And] chapter 2 (especially, 2.4)

- https://people.csail.mit.edu/alinush/6.858-fall-2014/papers/passwords-extended.pdf

- https://www.securityevaluators.com/casestudies/password-manager-hacking/

Questions ?