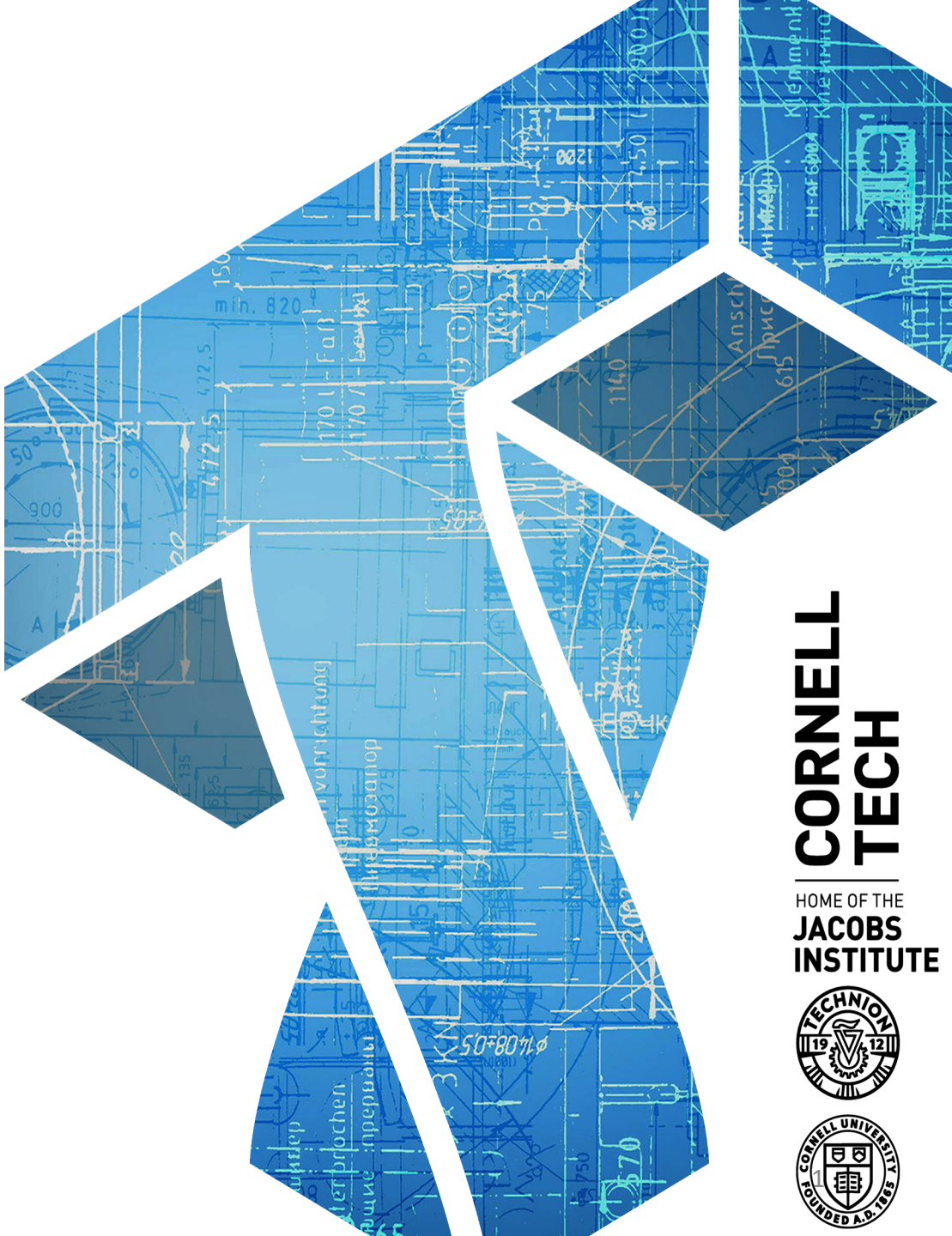
CS 5435: Authentication and passwords

Instructor: Tom Ristenpart

https://github.com/tomrist/cs5435-fall2019



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Airbnb as an example service

Hosts

Database
(e.g., SQL)

Guests

airbnb

What threats does Airbnb need to worry about?

Malicious user threats

Account ***abuse*** issues:

- Users posting malicious comments, reviews
- Scammers setting up fraudulent listings
- …

We'll return to these issues in a later lecture

Account ***security*** issues:

- Accounts given proper ***permissions***
- Protecting accounts from compromise

Need:

1. Permissions model
2. User authentication mechanisms

Permissions models

Different kinds of users:

- Guests
- Hosts
- Admins (e.g., moderators)

Subjects

Different kinds of resources:

- Profile pages
- Credit card data
- …

Objects

Access control matrix

Due originally to Lampson in 1971

Objects

Subjects

	resource 1	**resource 2**	**…**	**resource n**
user 1	read, write	read, write, own		read
user 2				
…				
user m	append	read, execute		read,write, own

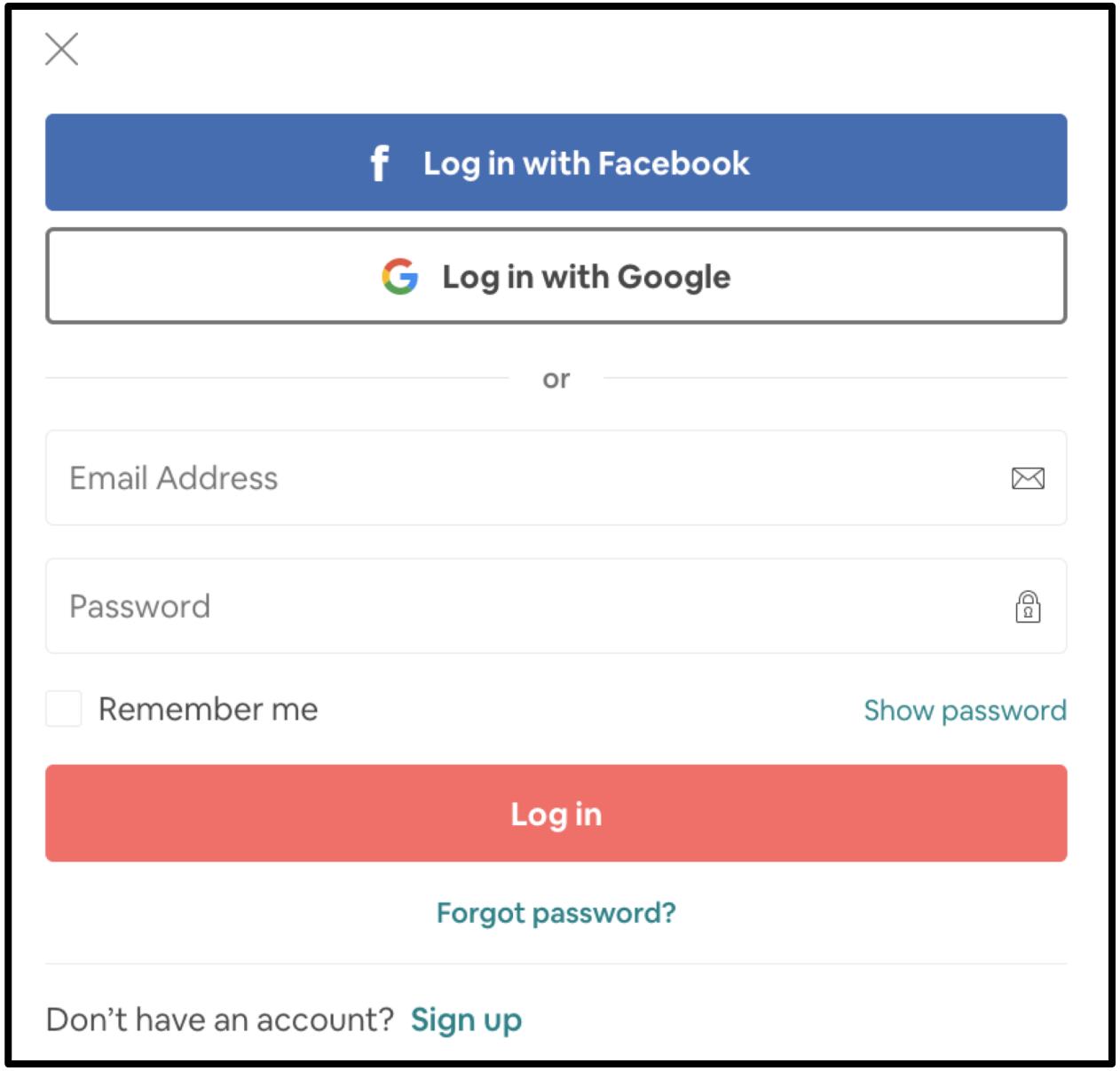
User i has permissions for resource j as indicated in cell [i,j]

Access control matrix often implicit. Sometimes: ***access control list*** (ACL)

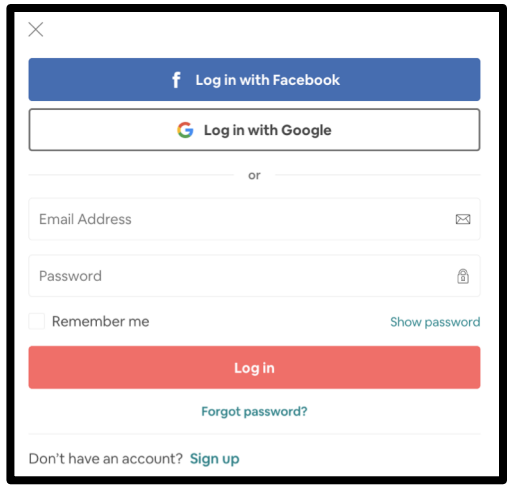
Unix-style file system permissions (rwx) implementation of this

User authentication mechanisms

- Web still mostly relies on password-based login
- Single-sign on (SSO)
- Usual approach:
  - Register username/password
  - Login via HTTP post
  - HTTP cookie set to identify logged-in session



Session handling and login



GET /login.html

Set-Cookie: AnonSessID=134fds1431

POST /login.html?name=bob&pw=12345

Cookie: AnonSessID=134fds1431

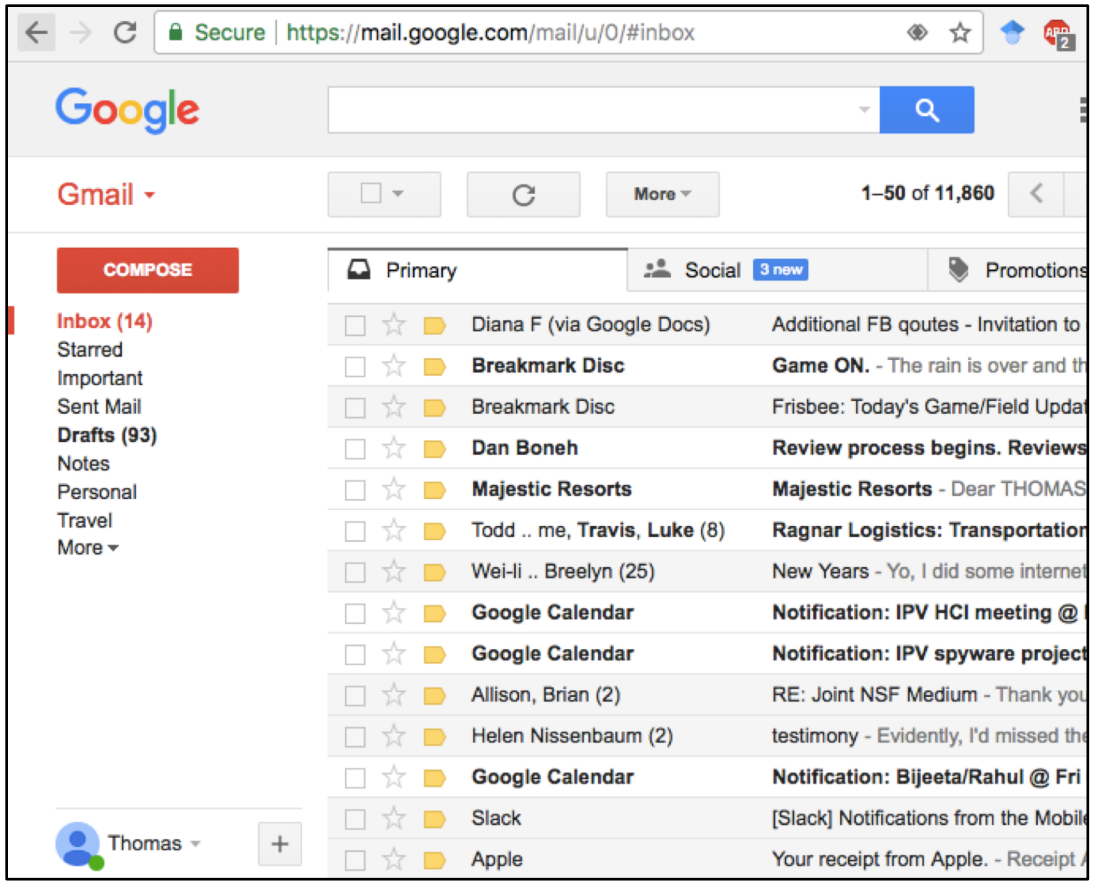
Set-Cookie: SessID=83431Adf

GET /account.html

Cookie: SessID=83431Adf

Need to check password matches registered version

My wife’s Airbnb compromise story



Hundreds of spam messages in her GMail

My wife’s Airbnb compromise story

Recovered account by resetting password

Proof of ownership of registered gmail account

Reset language setting back to English

Browser/Device	Location	Recent Activity	
Chrome Windows 7	New York, NY, United States	1 day ago	Logged Out
		about 1 hour ago less ▲	Log Out
		about 2 hours ago	Logged Out
		about 2 hours ago	Logged Out
Chrome Windows 10	Frankfurt am Main, HE, Germany	9 days ago	Logged Out
Chrome Windows 7	Balashikha, MOS, Russia	20 days ago	Logged Out

My wife’s Airbnb compromise story

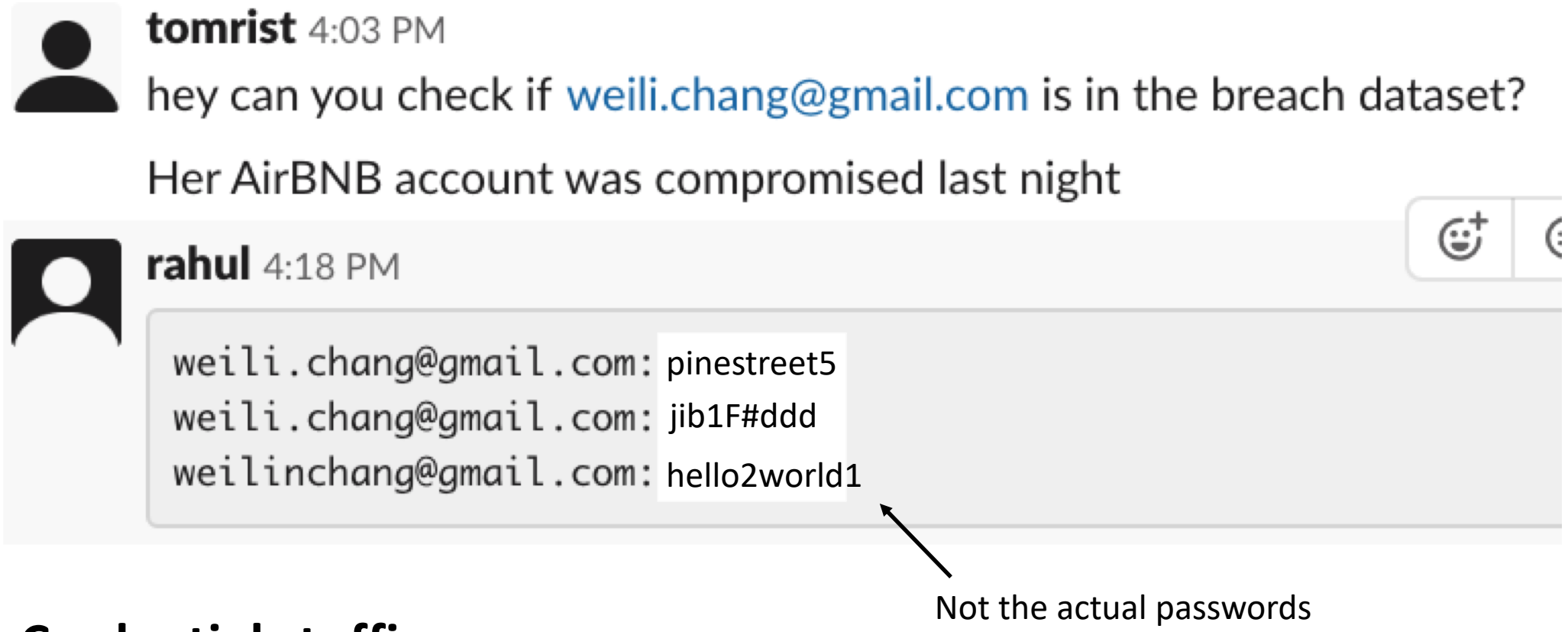
My account was hacked into, and I see several log-ins from Thailand and New York (where I live) that were not me. $2000 were charged for a Best of Vienna food tour, with receipts sent in Russian and Chinese. The tours have already passed, so I cannot just cancel, but I need a refund for these charges, otherwise I will dispute the charges through my credit cards. I thought that I usually have to authenticate my account using my
charges went through without me getting any texts

Спасибо за сообщение. Вам ответит первый осв
поддержки Airbnb.

Airbnb Support
8:36 AM
Have you recently made a booking with Luxury Retreats?
Please, fill up the following:

Name on card:
Type of card:
Transaction date:
Transaction amount:
Transaction currency:
First 6 and Last 4 digits of CC (not the full credit card number):
Is the CC shared? Yes or No:
Is the CC a corporate or private card? Yes or No:
Those are reservations for witch you were charged, right?

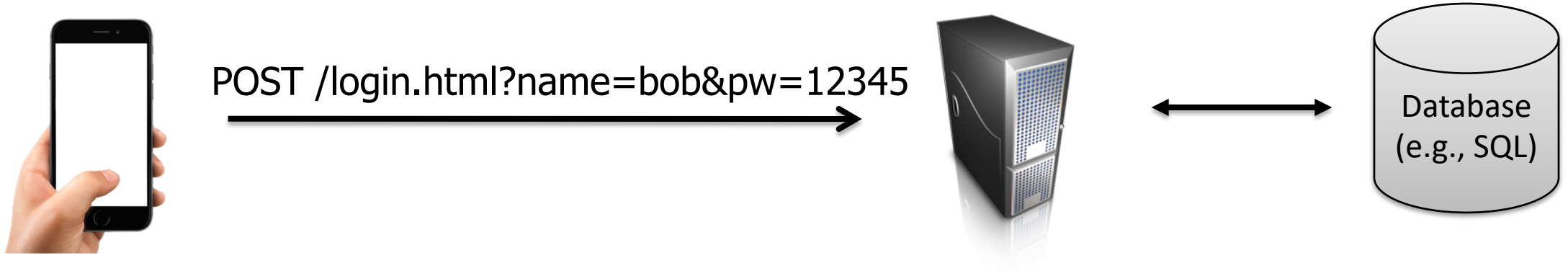
Meanwhile…



Credential stuffing:
try to compromise an account by submitting publicly disclosed passwords

My skype account was victim to this as well several years ago

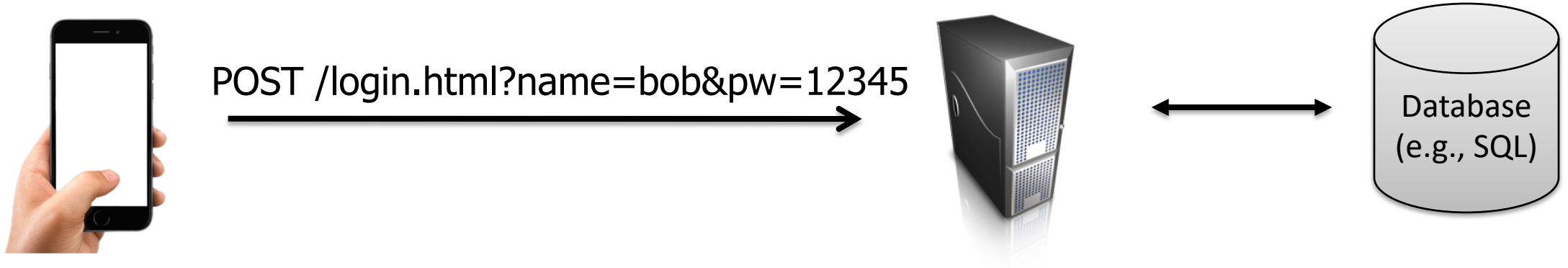
Password management



- 5 minutes to brainstorm ideas for how to improve password-based authentication handling
- Talk to your neighbor(s):

What can we do to prevent credential stuffing attacks?

Password management



Countermeasure	Purpose
Password hashing	Database leak doesn't immediately reveal user passwords. Slows ***offline guessing attacks***
Strength meters	Nudge / force users to pick stronger passwords to mitigate guessing attacks
Lockout after X failed attempts	Prevent remote guessing attacks (X typically 10, 100, 1000); slows down / prevents ***online guessing attacks***
Compromised credential checks	Check if password is in known breaches

Password hashing

- Cryptographic hash function H maps message to short digest (e.g., 256 bit string)
- ***One-way:*** Given y = H(M) hard to compute M
- ***Collision-resistant:*** Can't find M, M' s.t. H(M) = H(M')

- Cryptographers have designed good hash functions
  - SHA256, SHA512, SHA-3
  - Deprecated hashes: MD5, SHA-1

Brute-force password cracking

Given hash h = H(pw) for unknown pw, find pw

Dictionary:
List of probable passwords
or way of generating them

Count	password
290729	123456
79076	12345
76789	123456789
59462	password
49952	iloveyou
33291	princess
…	

Training alg. for language model

Model p

p is a probability model:
p(pw) is estimate of likelihood someone picks pw

Generate sequence of guesses

$pw_1, pw_2, pw_3, \ldots$ s.t.

$p(pw_1) \geq p(pw_2) \geq p(pw_3) \ldots$

$h_1 = H(pw_1)$

$h_2 = H(pw_2)$

$h_3 = H(pw_3)$

…

If $h = h_i$ then $pw = pw_i$

Brute-force password cracking

```
Sun Sep 08 16:20:42:cs5435-fall2019$ openssl speed sha256
Doing sha256 for 3s on 16 size blocks: 1319113 sha256's in 2.95s
Doing sha256 for 3s on 64 size blocks: 672037 sha256's in 2.94s
Doing sha256 for 3s on 256 size blocks: 273646 sha256's in 2.93s
Doing sha256 for 3s on 1024 size blocks: 82038 sha256's in 2.93s
Doing sha256 for 3s on 8192 size blocks: 10923 sha256's in 2.94s
```

~450,000 hashes per second

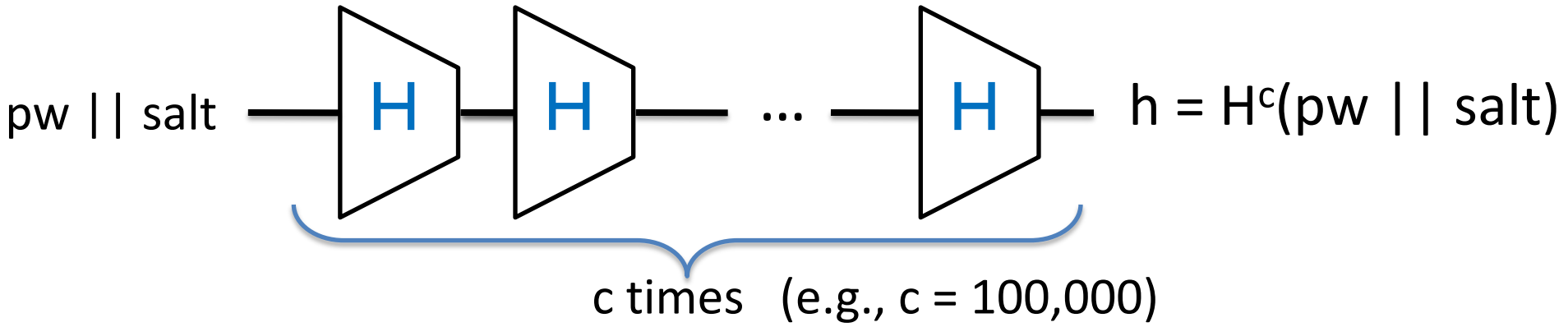
How many guesses / hashes needed to crack a password?

Rainbow tables:

precompute huge number of hashes to make quick-to-execute look-up table

Password hashing

- Make hashing slower to slow down cracking attacks
- Use random per-user salts to prevent use of rainbow tables
- PKCS#5 approach:



- Memory-hard hashing: Scrypt and argon2 require lots of memory to compute as well as time

Measuring password strength

- Hashing slows down, but doesn't prevent guessing attacks
- How do we measure password strength?
  - Old, now-considered-incorrect approaches:
    - NIST entropy estimates (ad hoc heuristic)
    - Shannon entropy
  - Nowdays:
    - Strength meters based on ***guess ranks***

Internet users ditch “password” as password, upgrade to “123456”

Contest for most commonly used terrible password has a new champion.

by **Jon Brodkin** - Jan 20 2014, 4:00pm GMT

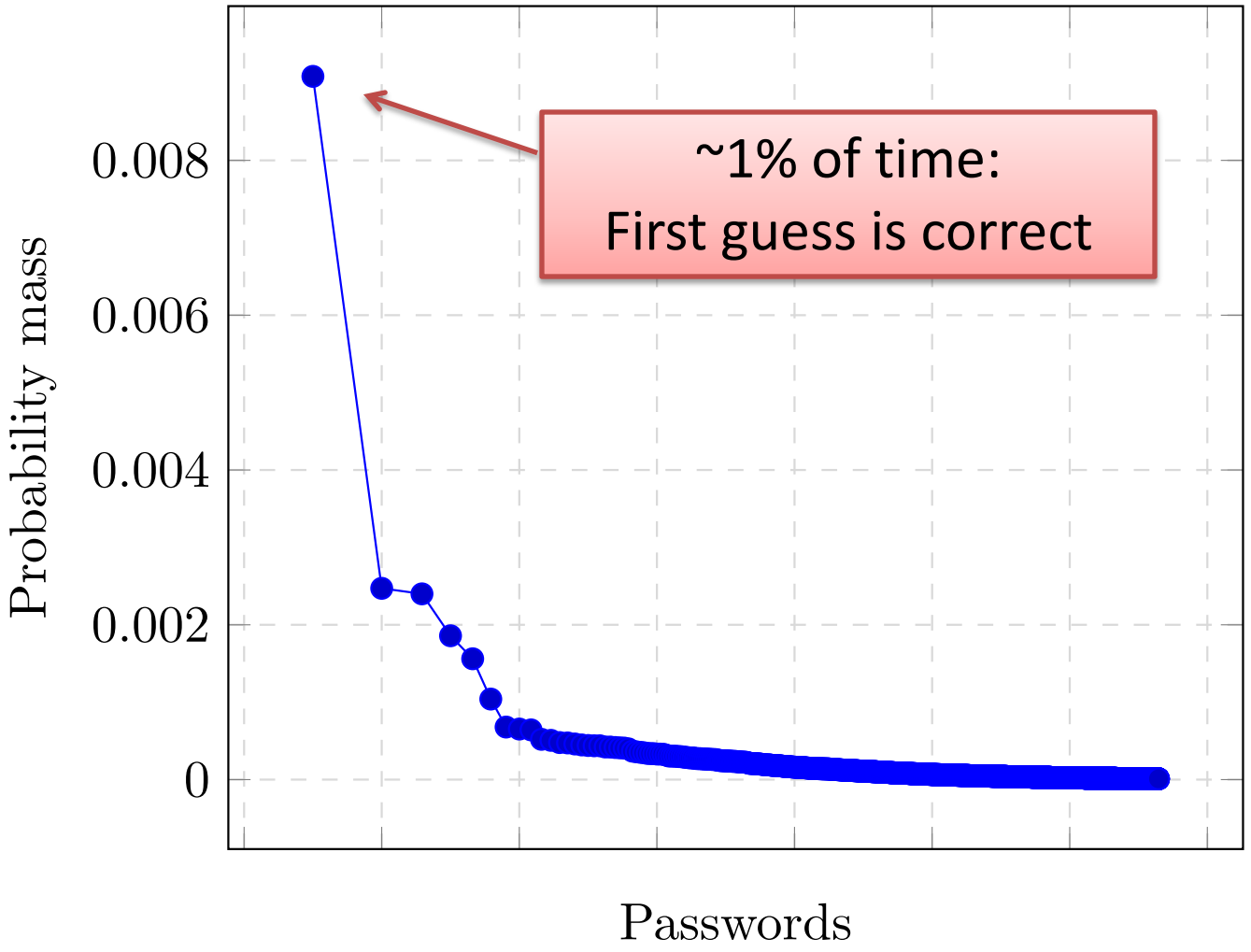
290729 123456
79076 12345
76789 123456789
59462 password
49952 iloveyou
33291 princess
21725 1234567
20901 rockyou
20553 12345678
16648 abc123
16227 nicole
15308 daniel
15163 babygirl
14726 monkey
14331 lovely

Rockyou data breach:
32 million social gaming accounts

Most common password used by almost 1%

[Bonneau 2012]
69 million Yahoo! Passwords
1.1% of users pick same password

Rockyou empirical probability mass function



(Only first 5,000 points shown)

Shannon entropy

Let $\mathcal{X}$ be password distribution.

Passwords are drawn iid from $\mathcal{X}$

N is size of support of $\mathcal{X}$

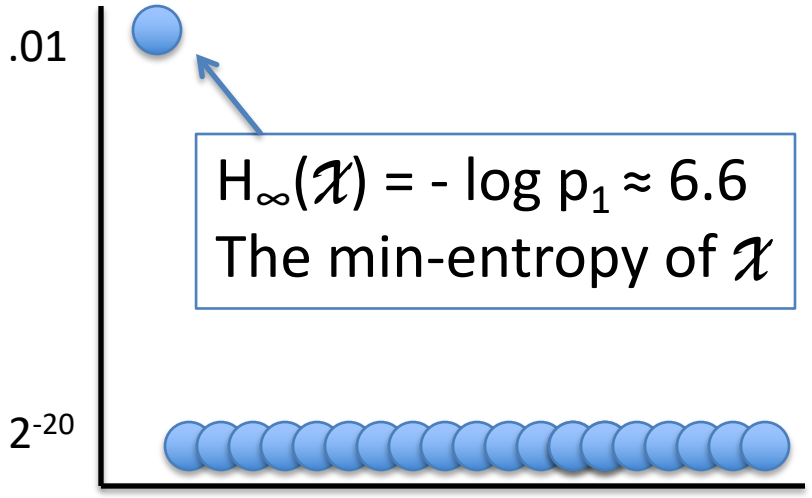
p_1 , p_2 , …, p_N are probabilities of passwords in decreasing order

Shannon entropy:

$$H_1(\mathcal{X}) = \sum_{i=1}^{N} -p_i \log p_i$$

Shannon entropy is poor measure (for password unpredictability)

N = 1,000,000
$p_1 = 1 / 100$
$p_2 = (1 - 1/100)/999{,}999 \approx 1 / 2^{20}$
…
$p_N = (1 - 1/100)/999{,}999 \approx 1 / 2^{20}$



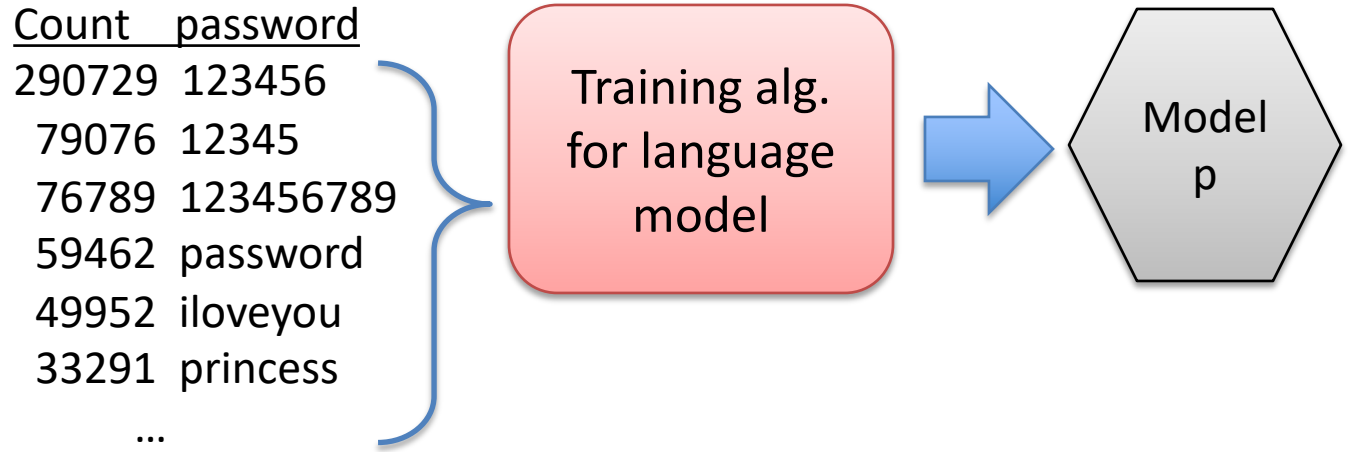
$H_1(\mathcal{X}) \approx 19$

19 bits of "unpredictability". Probability of success about $1/2^{19}$?

What is probability of success if attacker makes one guess?

Shannon entropy is almost never useful measure for security

Strength meters



$pw_1, pw_2, pw_3, \ldots$ s.t. $p(pw_1) \geq p(pw_2) \geq p(pw_3) \ldots$

Guess rank is index of password in this list

Strength meters: take min guess rank under various models p
lower the guess rank the weaker the password

Work well if guess ranks capture best known guessing attacks

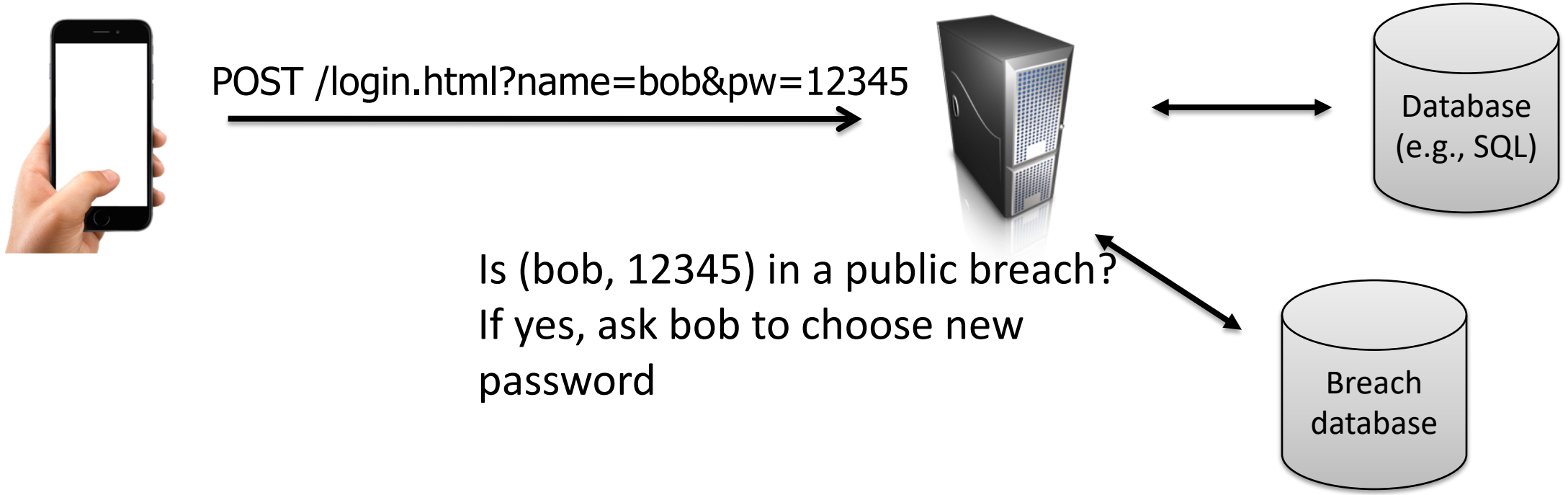
Zxcvbn is a good, fast one; Melicher et al. give one based on deep learning

Password selection requirements

- [Komanduri et al. ‘11], [Vu et al. ‘07], [Proctor et al. ‘02] and others studied with Amazon Mturk
  - General consensus that it increases resistance at least somewhat
  - Decreases usability (character class requirements very painful)

- Password expiration policies [Zhang et al. ’10]
  - High-level bit: they don’t work
  - Attacker with old password can crack new one very often

Credential stuffing countermeasures



Third-party services for making such queries:

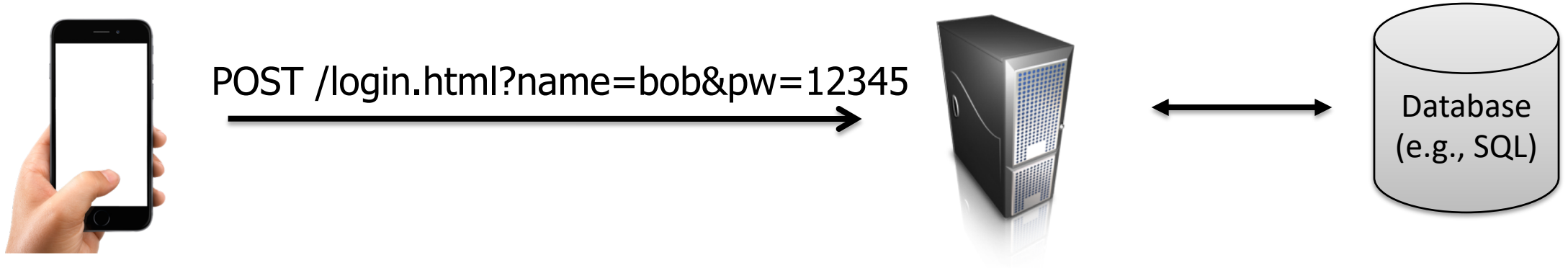
- HaveIBeenPwned
- Google password checker

Credential tweaking attacks

- Suppose bob changes password to 12345**6**
- Credential stuffing no longer works, but remote guessing attacker could try variants of (leaked) 12345
  - We call this a credential tweaking attack
- Deep learning mechanisms to learn conditional probability distribution
  - p(pw' | pw) where pw is leaked password, pw' is variant
  - Trained from leak data to capture typical password variants
- Experiments showed that 1,316 Cornell accounts vulnerable

[Pal et al. 2019]

Password management

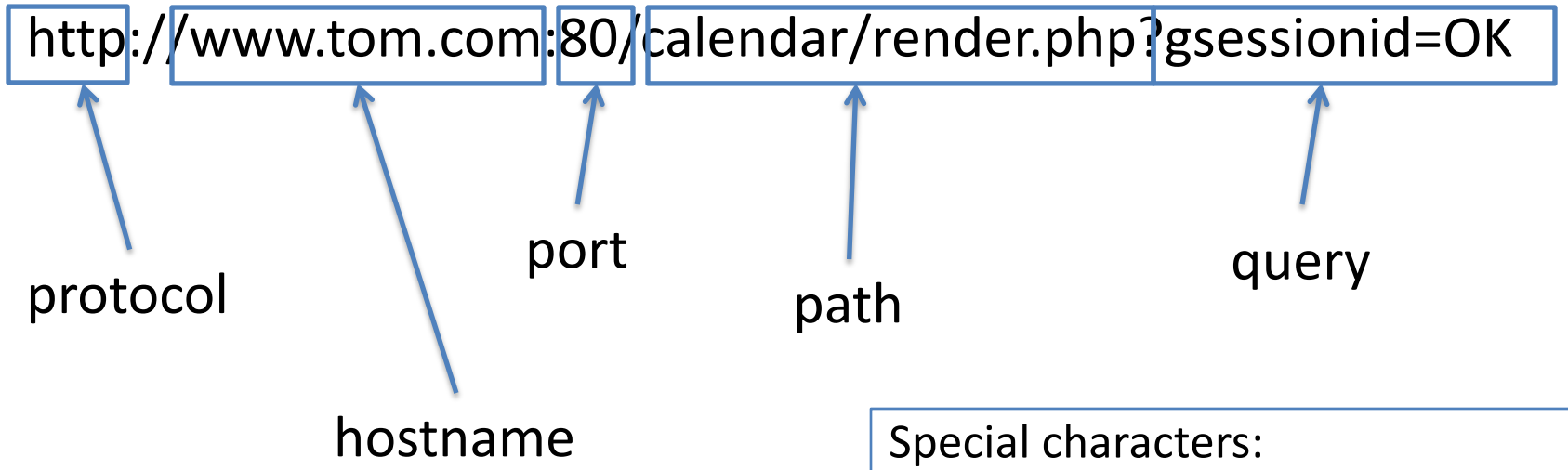


Countermeasure	Purpose
Password hashing	Database leak doesn't immediately reveal user passwords. Slows ***offline guessing attacks***
Strength meters	Nudge / force users to pick stronger passwords to mitigate guessing attacks
Lockout after X failed attempts	Prevent remote guessing attacks (X typically 10, 100, 1000); slows down / prevents ***online guessing attacks***
Compromised credential checks	Check if password is in known breaches

Summary

- Authentication of users a fundamental challenge in security
- We still rely on passwords as primary means of authenticating people
  - Offline brute-force cracking attacks
  - Remote guessing attacks
  - Credential stuffing and tweaking attacks
  - Web services can implement various defenses
- For users, password managers can help
- Next time: second factors, account recovery, single sign on

Uniform resource locators (URLs)

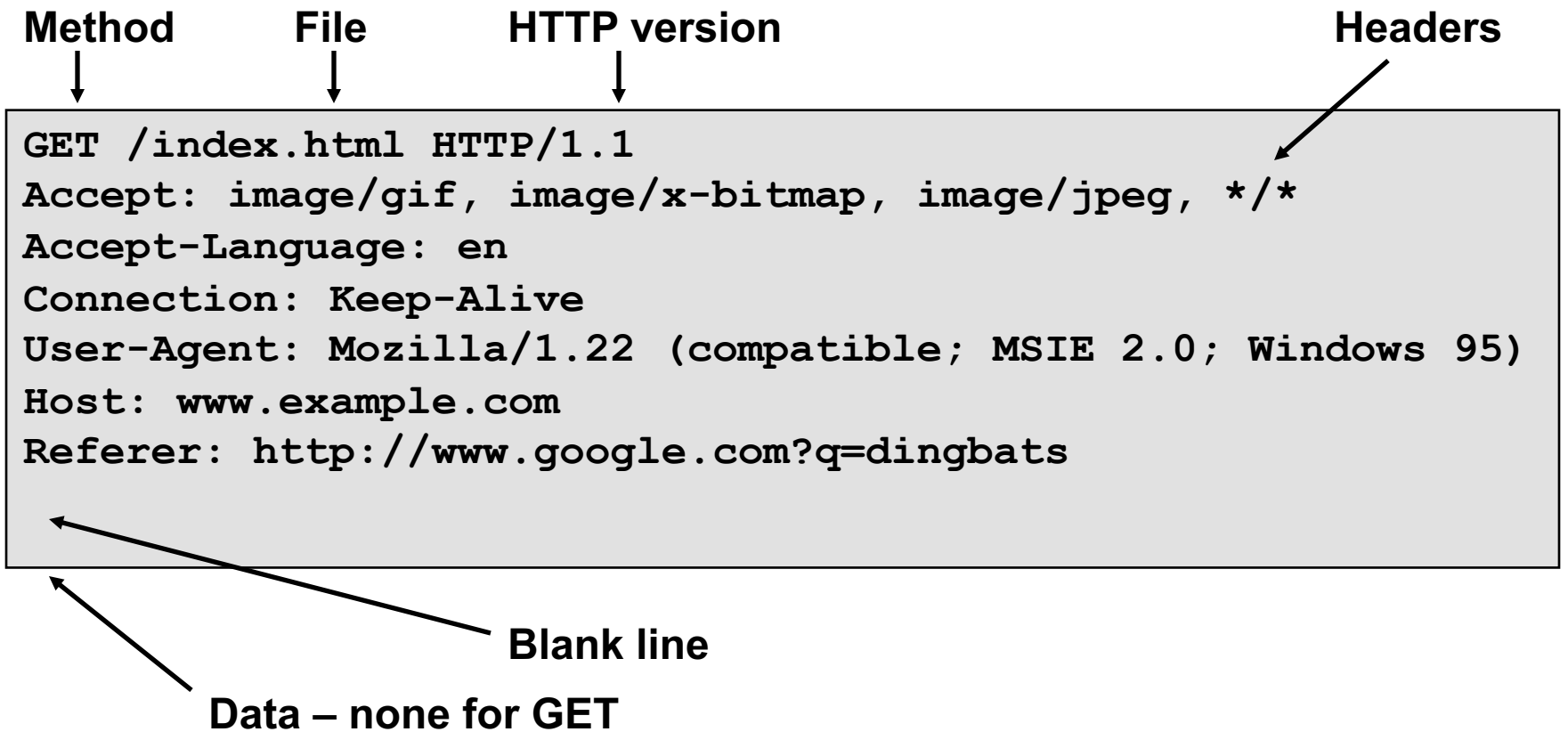


URL's only allow ASCII-US characters.
Encode other characters:

%0A = newline
%20 = space

Special characters:
+ = space
? = separates URL from parameters
% = special characters
/ = divides directories, subdirectories
= bookmark
& = separator between parameters

HTTP Request



GET : no side effect	POST : possible side effect

HTTP Response