Authentication factors

- Something you know (password, PIN)
- Something you have (e.g., smart card)
- Something you are (biometrics)
- CAPTCHAs, time and location, …
- Multi-factor authentication

Passwords: love to hate

- Many problems for users, sysadmins, researchers
- But familiar and near-zero cost of entry
- User-chosen passwords proliferate for low-stakes web site authentication

Password entropy

- Model password choice as probabilistic process
- If uniform, \( \log_2 |S| \)
- Controls difficulty of guessing attacks
- Hard to estimate for user-chosen passwords
  - Length is an imperfect proxy

Password hashing

- Idea: don’t store password or equivalent information
- Password ‘encryption’ is a long-standing misnomer
  - E.g., Unix crypt(3)
- Presumably hard-to-invert function \( h \)
- Store only \( h(p) \)

Dictionary attacks

- Online: send guesses to server
- Offline: attacker can check guesses internally
- Specialized password lists more effective than literal dictionaries
  - Also generation algorithms (s → $, etc.)
- ~25% of passwords consistently vulnerable

Better password hashing

- Generate random salt \( s \), store \((s, h(s, p))\)
  - Block pre-computed tables and equality inferences
  - Salt must also have enough entropy
- Deliberately expensive hash function
  - AKA password-based key derivation function (PBKDF)
  - Requirement for time and/or space
Password usability

- User compliance can be a major challenge
  - Often caused by unrealistic demands
- Distributed random passwords usually unrealistic
- Password aging: not too frequently
- Never have a fixed default password in a product

Backup authentication

- Desire: unassisted recovery from forgotten password
- Fall back to other presumed-authentic channel
  - Email, cell phone
- Harder to forget (but less secret) shared information
  - Mother’s maiden name, first pet’s name
- Brittle: ask Sarah Palin or Mat Honan

Backup auth suggestion: use time

- Need for backup often comes for infrequently-used accounts
- May be acceptable to slow down recovery if it reduces attack risk
  - Account recovery is a hassle anyway
- Time can allow legitimate owner to notice malicious request

Centralized authentication

- Enterprise-wide (e.g., UMN ID)
- Anderson: Microsoft Passport
- Today: Facebook Connect, Google ID
- May or may not be single-sign-on (SSO)

Biometric authentication

- Authenticate by a physical body attribute
  - Hard to lose
  - Hard to reset
  - Inherently statistical
  - Variation among people

Example biometrics

- (Handwritten) signatures
- Fingerprints, hand geometry
- Face and voice recognition
- Iris codes

Outline

User authentication
Announcements intermission
Error rate trade-offs
Web authentication
TLS and certificates
Names and identities

Some small updates

- There will be no problem set 2
- There will be labs both this week and next
- SRTs are open, and I’ll allocate lecture time for them a week from today
Outline
User authentication
Announcements intermission
Error rate trade-offs
Web authentication
TLS and certificates
Names and identities

Imperfect detection
Many security mechanisms involve imperfect detection/classification of relevant events
Biometric authentication
Network intrusion detection
Anti-virus (malware detection)
Anything based on machine learning

Detection results
True positive: detector says yes, reality is yes
True negative: detector says no, reality is no
False positive: detector says yes, reality is no
False negative: detector says no, reality is yes
Note: terminology may flip based on detecting good or bad

Why a trade-off?
Imperfect methods have a trade-off between avoiding FPs and avoiding FNs
Sometimes a continuous trade-off (curve), e.g. based on a threshold
E.g., spam detector “score”
May need to choose both a basic mechanism and a threshold

Two ratios to capture the trade-off
True positive rate:
\[ \text{TPR} = \frac{TP}{P} = \frac{TP}{TP + FN} = 1 - \text{FNR} \]
False positive rate:
\[ \text{FPR} = \frac{FP}{N} = \frac{FP}{FP + TN} = 1 - \text{TNR} \]

ROC curve intro

Error rates: ROC curve

Extreme biometrics examples
exact_iris_code_match: very low false positive (false authentication)
similar_voice_pitch: very low false negative (false reject)
Where are these in ROC space?

A if (iris()) return REJECT; else return ACCEPT;
B return REJECT;
C if (iris()) return ACCEPT; else return REJECT;
D if (iris() && pitch()) return ACCEPT; else return REJECT;
E return ACCEPT;
F if (rand() & 1) return ACCEPT; else return REJECT;
G if (pitch()) return ACCEPT; else return REJECT;
H if (iris() || pitch()) return ACCEPT; else return REJECT;

Outline

User authentication
Announcements intermission
Error rate trade-offs
Web authentication
TLS and certificates
Names and identities

Per-website authentication

Many web sites implement their own login systems
+ If users pick unique passwords, little systemic risk
  - Inconvenient, many will reuse passwords
  - Lots of functionality each site must implement correctly
  - Without enough framework support, many possible pitfalls

Building a session

HTTP was originally stateless, but many sites want stateful login sessions
Built by tying requests together with a shared session ID
Must protect confidentiality and integrity

Session ID: what

Must not be predictable
- Not a sequential counter
Should ensure freshness
- Eg, limited validity window
If encoding data in ID, must be unforgeable
- Eg, data with properly used MAC
- Negative example: crypt(username || server secret)

Session ID: where

Session IDs in URLs are prone to leaking
Including via user cut-and-paste
Usual choice: non-persistent cookie
Against network attacker, must send only under HTTPS
Because of CSRF, should also have a non-cookie unique ID

Session management

Create new session ID on each login
Invalidate session on logout
Invalidate after timeout
- Usability / security tradeoff
- Needed to protect users who fail to log out from public browsers

Account management

Limitations on account creation
- CAPTCHA? Outside email address?
See previous discussion on hashed password storage
Automated password recovery
- Usually a weak spot
- But, practically required for large system
Client and server checks

- For usability, interface should show what's possible
- But must not rely on client to perform checks
- Attackers can read/modify anything on the client side
- Easy example: item price in hidden field

Direct object references

- Seems convenient: query parameter names resource directly
  - E.g., database key, filename (path traversal)
- Easy to forget to validate on each use
- Alternative: indirect reference like per-session table
  - Not fundamentally more secure, but harder to forget check

Function-level access control

- E.g., pages accessed by URLs or interface buttons
- Must check each time that user is authorized
  - Attack: find URL when authorized, reuse when logged off
- Helped by consistent structure in code

Outline

- User authentication
- Announcements intermission
- Error rate trade-offs
- Web authentication
- TLS and certificates
- Names and identities

HTTPS hierarchical PKI

- Browser has order of 100 root certs
  - Not same set in every browser
  - Standards for selection not always clear
- Many of these in turn have sub-CAs
- Also, “wildcard” certs for individual domains

Hierarchical trust?

- No. Any CA can sign a cert for any domain
- A couple of CA compromises recently
- Most major governments, and many companies you’ve never heard of, could probably make a google.com cert
- Still working on: make browser more picky, compare notes

CA validation standards

- CA's job to check if the buyer really is foo.com
- Race to the bottom problem:
  - CA has minimal liability for bad certs
  - Many people want cheap certs
  - Cost of validation cuts out of profit
- "Extended validation" (green bar) certs attempt to fix

HTTPS and usability

- Many HTTPS security challenges tied with user decisions
- Is this really my bank?
- Seems to be a quite tricky problem
  - Security warnings often ignored, etc.
Outline
User authentication
Announcements intermission
Error rate trade-offs
Web authentication
TLS and certificates
Names and identities

Accounts versus identities
"Identity" is a broad term that can refer to a personal conception or an automated system.
"Name" is also ambiguous in this way.
"Account" and "authentication" refer unambiguously to institutional/computer abstractions.
Any account system is only an approximation of the real world.

Real human names are messy
• Most assumptions your code might make will fail for someone
  • ASCII, length limit, uniqueness, unchanging, etc.
• So, don’t design in assumptions about real names
• Use something more computer-friendly as the core identifier
  • Make “real” names or nicknames a presentation aspect

Zooko’s triangle
Claims (2001) it is hard/impossible for a naming scheme to be simultaneously:
• Human-meaningful
• Secure
• Decentralized
Too imprecise to be definitively proven/refuted
• Blockchain-based name systems are highest-profile claimed counterexamples
• A useful heuristic for seeing design tensions

Identity documents: mostly unhelpful
“Send us a scan of your driver’s license”
• Sometimes called for by specific regulations
• Unnecessary storage is a disclosure risk
• Fake IDs are very common

Identity numbers: mostly unhelpful
Common US example: social security number
• Variously used as an identifier or an authenticator
  • Dual use is itself a cause for concern
• Known by many third parties (e.g., banks)
• No checksum, guessing risks
• Published soon after a person dies

“Identity theft”
The first-order crime is impersonation fraud between two other parties
• E.g., criminal trying to get money from a bank under false pretenses
• The impersonated “victim” is effectively victimized by follow-on false statements
  • E.g., by credit reporting agencies
  • These costs are arguably the result of poor regulatory choices
• Be careful w/ negative info from 3rd parties