Outline

User authentication, cont’d
Error rate trade-offs
Good technical writing (pt. 1)
Web authentication
Names and Identities

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

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

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

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, cont’d
Error rate trade-offs
Good technical writing (pt. 1)
Web authentication
Names and Identities

Writing in CS versus other writing

- Key goal is accurately conveying precise technical information
- More important: careful use of terminology, structured organization
- Less important: writer’s personality, persuasion, appeals to emotion

Extreme biometrics examples

- exact_iris_code_match: very low false positive (false authentication)
- similar_voice_pitch: very low false negative (false reject)
Still important: concise expression

- Don't use long words or complicated expressions when simpler ones would convey the same meaning.
  Examples:
  - necessitate
  - utilize
  - due to the fact that
- Beneficial for both clarity and style

Know your audience: terminology

- When technical terminology makes your point clearly, use it.
- But provide definitions if a concept might be new to many readers.
  - Be careful to provide the right information in the definition.
  - Define at the first instead of a later use.
- On other hand, avoid introducing too many new terms.
  - Keep the same term when referring to the same concept.

Precise explanations

- Don't say "we" do something when it's the computer that does it.
  - And avoid passive constructions.
- Don't anthropomorphize (computers don't "know").
- Use singular by default so plural provides a distinction:
  - The students take tests.
  - Each student takes a test.
  - Each student takes multiple tests.

Provide structure

- Use plenty of sections and sub-sections.
- It's OK to have some redundancy in previewing structure.
- Limit each paragraph to one concept, and not too long.
  - Start with a clear topic sentence.
  - Split long, complex sentences into separate ones.

Plagiarism and citations

- Never use someone else's writing to make it look like your own.
  - Overlaps with but different than cheating.
- Give proper credit for ideas that you get from somewhere else.
  - For 4271, mostly don't need to credit course resources.
  - We have no specific requirements about citation format.

Know your audience: Project

- For projects in this course, assume your audience is another student who already understands general course concepts.
  - Up to the current point in the course.
  - I.e., don't need to define "buffer overflow" from scratch.
- But you need to explain specifics of bcimgview.
  - Make clear what part of the program you're referring to.
  - Explain all the specific details of a vulnerability.

Inclusive language

- Avoid words and grammar that implies relevant people are male.
- My opinion: avoid using he/him pronouns for unknown people.
- Some possible alternatives:
  - "he/she"
  - Alternating genders.
  - Rewrite to plural and use "they" (may be less clear).
  - Singular "they" (least traditional, but spreading).

Outline

- User authentication, cont'd
- Error rate trade-offs
- Good technical writing (pt. 1)
- Web authentication
- 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
  - E.g., limited validity window
- If encoding data in ID, must be unforgeable
  - E.g., 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, cont’d
- Error rate trade-offs
- Good technical writing (pt. 1)
- Web authentication
- 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
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