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;

Testing and security

“Testing shows the presence, not the absence of bugs” – Dijkstra
Easy versions of some bugs can be found by targeted tests:
- Buffer overflows: long strings
- Integer overflows: large numbers
- Format string vulnerabilities: %x

Random or fuzz testing

Random testing can also sometimes reveal bugs
Original ‘fuzz’ (Miller): program </dev/urandom
Even this was surprisingly effective

Mutational fuzzing

Instead of totally random inputs, make small random changes to normal inputs
Changes are called mutations
Benign starting inputs are called seeds
Good seeds help in exercising interesting/deep behavior

Grammar-based fuzzing

Observation: it helps to know what correct inputs look like
Grammar specifies legal patterns, run backwards with random choices to generate
Generated inputs can again be basis for mutation
Most commonly used for standard input formats
- Network protocols, JavaScript, etc.
What if you don't have a grammar?

- Input format may be unknown, or buggy and limited
- Writing a grammar may be too much manual work
- Can the structure of interesting inputs be figured out automatically?

Coverage-driven fuzzing

- Instrument code to record what code is executed
- An input is interesting if it executes code that was not executed before
- Only interesting inputs are used as basis for future mutation

AFL

- Best known open-source tool, pioneered coverage-driven fuzzing
- American Fuzzy Lop, a breed of rabbits
- Stores coverage information in a compact hash table
- Compiler-based or binary-level instrumentation
- Has a number of other optimizations

Outline

- ROC curve exercise, cont’d
- Testing and fuzzing
- Announcements intermission
- Usability and security
- Usable security example areas

Last parts of the course

- Today is the last lecture
- Monday 5/2 is the last lab, also:
  - Due date for Project 2
  - Last date to submit SRTs
- No meetings or assignments during finals

Outline

- ROC curve exercise, cont’d
- Testing and fuzzing
- Announcements intermission
- Usability and security
- Usable security example areas

Users are not 'ideal components'

- Frustrates engineers: cannot give users instructions like a computer
  - Closest approximation: military
- Unrealistic expectations are bad for security

Most users are benign and sensible

- On the other hand, you can't just treat users as adversaries
  - Some level of trust is inevitable
  - Your institution is not a prison
- Also need to take advantage of user common sense and expertise
  - A resource you can't afford to pass up
Don’t blame users

- “User error” can be the end of a discussion
- This is a poor excuse
- Almost any “user error” could be avoidable with better systems and procedures

Users as rational

- Economic perspective: users have goals and pursue them
  - They’re just not necessarily aligned with security
  - Ignoring a security practice can be rational if the rewards is greater than the risk

Perspectives from psychology

- Users become habituated to experiences and processes
  - Learn “skill” of clicking OK in dialog boxes
- Heuristic factors affect perception of risk
  - Level of control, salience of examples
- Social pressures can override security rules
  - “Social engineering” attacks

User attention is a resource

- Users have limited attention to devote to security
  - Exaggeration: treat as fixed
  - If you waste attention on unimportant things, it won’t be available when you need it
  - Fable of the boy who cried wolf

Research: ecological validity

- User behavior with respect to security is hard to study
- Experimental settings are not like real situations
- Subjects often:
  - Have little really at stake
  - Expect experimenters will protect them
  - Do what seems socially acceptable
  - Do what they think the experimenters want

Research: deception and ethics

- Have to be very careful about ethics of experiments with human subjects
  - Enforced by institutional review systems
- When is it acceptable to deceive subjects?
  - Many security problems naturally include deception

Outline

ROC curve exercise, contd
Testing and fuzzing
Announcements intermission
Usability and security
Usable security example areas

Email encryption

- Technology became available with PGP in the early 90s
- Still an open “challenge problem”
- Also some other non-UI difficulties: adoption, govt. policy
Phishing
- Attacker sends email appearing to come from an institution you trust
- Links to web site where you type your password, etc.
- Spear phishing: individually targeted, can be much more effective

Phishing defenses
- Educate users to pay attention to X:
  - Spelling → copy from real emails
  - URL → homograph attacks
  - SSL "lock" icon → fake lock icon, or SSL-hosted attack
- Extended validation (green bar) certificates
- Phishing URL blacklists

SSL warnings: prevalence
- Browsers will warn on SSL certificate problems
- In the wild, most are false positives
  - foo.com vs. www.foo.com
  - Recently expired
  - Technical problems with validation
  - Self-signed certificates (HA2)
- Classic warning-fatigue danger

SSL warnings: effectiveness
- Early warnings fared very poorly in lab settings
- Recent browsers have a new generation of designs:
  - Harder to click through mindlessly
  - Persistent storage of exceptions
- Recent telemetry study: they work pretty well

Modern Firefox warning

Modern Firefox warning (2)
**Spam-advertised purchases**

- “Replica” Rolex watches, herbal Vi@gr@, etc.
- This business is clearly unscrupulous; if I pay, will I get anything at all?
- Empirical answer: yes, almost always
- Not a scam, a black market
- Importance of credit-card bank relationships

**Advance fee fraud**

- Why do Nigerian Scammers say they are from Nigeria? (Herley, WEIS 2012)
- Short answer: false positives
  - Sending spam is cheap
  - But, luring victims is expensive
  - Scammer wants to minimize victims who respond but ultimately don’t pay

**Trusted UI**

- Tricky to ask users to make trust decisions based on UI appearance
  - Lock icon in browser, etc.
- Attacking code can draw lookalike indicators
  - Lock favicon
  - Picture-in-picture attack

**Smartphone app permissions**

- Smartphone OSes have more fine-grained per-application permissions
  - Access to GPS, microphone
  - Access to address book
  - Make calls
- Phone also has more tempting targets
- Users install more apps from small providers

**Permissions manifest**

- Android approach: present listed of requested permissions at install time
  - Can be hard question to answer hypothetically
    - Users may have hard time understanding implications
  - User choices seem to put low value on privacy

**Time-of-use checks**

- iOS approach: for narrower set of permissions, ask on each use
  - Proper context makes decisions clearer
  - But, have to avoid asking about common things
  - iOS app store is also more closely curated

**Trusted UI for privileged actions**

- Trusted UI works better when asking permission (e.g., Oakland’12)
- Say, “take picture” button in phone app
  - Requested by app
  - Drawn and interpreted by OS
  - OS well positioned to be sure click is real
  - Little value to attacker in drawing fake button