Application Review

- Reviewing a mature (possibly complete) application
- A daunting task if the system is large
- And often you know little about it
  - Maybe you performed a design review
  - Maybe you read design review docs
  - Maybe less than that
- How do you get started?

Need to Define a Process

- Don’t just dive into the code
- Process should be:
  - Pragmatic
  - Flexible
  - Results oriented
- Will require code review
  - Which is a skill one must develop

Review Process Outline

1. Preassessment
   – Get high level view of system
2. Application review
   – Design review, code review, maybe live testing
3. Documentation and analysis
4. Remediation support
   – Help them fix the problems

- May need to iterate

Reviewing the Application

- You start off knowing little about the code
- You end up knowing a lot more
- You'll probably find the deepest problems related to logic after you understand things
- A design review gets you deeper quicker
  - So worth doing, if not already done
- The application review will be an iterative process

General Approaches To Design Reviews

- Top-down
  - Start with high level knowledge, gradually go deeper
- Bottom-up
  - Look at code details first, build model of overall system as you go
- Hybrid
  - Switch back and forth, as useful

Code Auditing Strategies

- Code comprehension (CC) strategies
  - Analyze source code to find vulnerabilities and increase understanding
- Candidate point (CP) strategies
  - Create list of potential issues and look for them in code
- Design generalization (DG) strategies
  - Flexibly build model of design to look for high and medium level flaws

Some Example Strategies

- Trace malicious input (CC)
  - Trace paths of data/control from points where attackers can inject bad stuff
- Analyze a module (CC)
  - Choose one module and understand it
- Simple lexical candidate points (CP)
  - Look for text patterns (e.g., `strcpy()`)
- Design conformity check (DG)
  - Determine how well code matches design

Guidelines for Auditing Code

- Perform flow analysis carefully within functions you examine
- Re-read code you've examined
- Desk check important algorithms
- Use test cases for important algorithms
  - Using real system or desk checking
  - Choosing inputs carefully

Useful Auditing Tools

- Source code navigators
- Debuggers
- Binary navigation tools
- Fuzz-testing tools
  - Automates testing of range of important values