Engineering Secure Software

BLACK BOX TESTING

The Power of Objectivity

- Black box testing
  - No knowledge of the source code
  - Cursory knowledge of the architecture and protocols
  - Often performed by a third-party
- Why limit our own knowledge?
- Objectivity
  - Different blind spots than the developers
  - Assumptions are closer to attackers
- Assessment
  - Persuasive to upper management and customers
  - If a naïve tester can get this far, who else can get this far?

Exploratory & Penetration Testing

- Exploratory testing
  - Scope: entire attack surface
  - Goals
    - Enumerate areas of attack
    - Find that low-hanging fruit
    - Mostly outer defenses
  - Drawback: results not always actionable

- Penetration testing
  - Scope: small area of the attack surface
  - Goals:
    - Focus intently on a few features
    - Determine exploitability (construct actual exploits)
    - Evaluate defense in depth
  - Drawback: proof by existence
    lack of vulnerabilities found != lack of vulnerabilities

Areas of Expertise

- Know the environment
  - Networking (webapps, low-level protocols)
  - Operating systems (mobile, desktop, & server)
  - Database management
- Know where to look
  - “Where there’s smoke, there’s fire”
  - Bug in security feature is often a vulnerability
  - Common assumptions
- Recognize a potential vulnerability
  - Vague, system error messages
  - Strange behavior
- Constructing exploits

In Practice...

- Exploratory and penetration testing are done simultaneously
  - Start with exploratory testing
  - Drill down with penetration testing
- When teams have security problems, they often just hire pen-testers
  - Despite this being the least efficient approach (and too late)
  - Many people confuse security expertise with penetration testing expertise
- Requires many hours of practice
  - Start with known vulnerabilities
    - How could I have recognized it?
    - Try to construct an exploit for it
  - Build your own tools

Exploratory: Fuzz Testing

- Goal:
  - Reveal the attack surface
  - Look for candidates of pen-testing
  - Examine how the system reacts, without going too deep
- Simulate the user
  - Reverse-engineer common behavior
    - Record a normal sequence of operations
    - Identify & modify the inputs
  - Explore beyond common behavior
  - Automate the process
- Output:
  - Vulnerability candidates
  - Automatically constructing exploits is usually too much work

Tools of Exploratory Testing

- Web applications
  - HTML viewers & parsers: Firebug, developer tools
  - Automate common operations: Greasemonkey
  - Web client tampering tools
- Networking
  - Discover local ports: netstat
  - Discover remote ports: nmap
  - Sniff the network: Wireshark, ngrep, tcpdump
- Operating system
  - Find open file handles: lsof
  - Process tree: ps

Tools of Penetration Testing

- Reverse engineering
  - Disassembly & Decompilation: IDA-pro, javap,
  - Debuggers: gdb
- Networking
  - Proxies: for intercepting & tampering traffic
  - Custom fault injectors
- Password cracking
  - Online guessing
  - Offline cracks
- Buckets of tools
  - Metasploit: database of specific exploits
  - BackTrack: Linux distribution with tons of tools
  - http://sectools.org/

Before You Start

- Define what is sensitive
  - Test data: e.g. config files, sensitive records
  - Environment information: e.g. server versions
  - Database schema
- Control the environment
  - e.g. use your own server
  - Be ready to try different configurations
  - Use virtualization to contain your exploits
- Ask for the keys
  - To study defense in depth
  - e.g. multiple authorizations to a test system
- Be ready to dive in… this will take a while

Don’t Forget the Feedback!

- Transfer your knowledge back to developers
  - Highlight the assumptions that they made
  - Discuss what could have been done to avoid it
- Don’t just poke holes, help with the mitigation
  - Egos are often bruised, at that point it’s time to help
  - Fix the vulnerability, not block the exploit
- Focus on process improvement
  - Checklists for future inspections
  - Newly-identified assets
  - Produce tests for similar features