Voting System Testing and Evaluation Addressing Resolution 17-05

John Kelsey
Computer Scientist
NIST Computer Security Division
March 9, 2005 TGDC Meeting

NOTE: This is all very preliminary!

Overview

- The Resolution
- Checklists, Testing, and Open-Ended Evaluation
- Evaluating a Design
- Evaluating The Pieces
- Some Issues to Resolve

Resolution 17-05

The TGDC directs NIST to research and draft standards documents requiring testing of voting systems that includes a significant amount of ***open-ended research for vulnerabilities....***

Why Do We Need Open-Ended Evaluation?

- Can't rely on simple checklist approach
  - Lots of ways to use strong components in weak ways
  - Sometimes underlying design is fatally flawed
- Evaluation needs to be adversarial
  - Goal is to find weaknesses before system is fielded
  - Try to find a way to fail the system
- Can't trust vendor assertions w/o verification
  - Vendor insiders may be in on an attack

Background: An Adversarial Model of the World

- We have to assume existence of serious attackers
  - Money (hundreds of millions spent on 2004 elections)
  - Access (history of insider attacks in voting systems)
  - Risk Tolerance (activists and extremists willing to do major crimes for their causes)
- Don't underestimate attackers!
  - Full access to system internals
  - Possible insider access
  - Intelligence and expertise at least equal to designers

Open-Ended Search for Weaknesses

- What we're looking at:
  - Top-level system design and architecture
  - Documentation and procedures
  - Software and OS configuration
  - Hardware
  - Communications
- Assumptions:
  - Insider access
  - Full knowledge of system
  - Large budget and risk tolerance

Open-Ended Evaluation: Evaluating a Design

- Using system documentation, look at the whole design and see if it really is secure
  - Many attacks found at this stage don't work, because of details that weren't mentioned in the docs
  - Sometimes find very fundamental flaws in assumptions or designs
  - Check to see if procedures in documentation really address problems
  - Result: List of possible attacks, attack patterns, and things to more closely check in later stages

New requirements on system documentation!

Evaluating the Pieces

- We get list of requirements on software security from high-level evaluation and documentation:
  - Evaluate S/W, OS config, physical config based on requirements from above evaluation step
- Evaluate:
  - Software (custom and COTS)
  - OS configuration
  - Physical configuration
  - Communications
- All: Mix of checklists, automated tools, open-ended search for weaknesses

Evaluating the Pieces (2)

- Communications:
  - Secure communications (encryption, auth, orig ident)
  - Protect from communications (lock down box)
  - Test, watch communications, check S/W & config
- Software:
  - Development Environment (version ctrl, security,QA)
  - Testing / Vulnerability Scans
  - Code Review

Verify claims in system documentation and do open-ended search for problems with all these.

Evaluating the Pieces(3)

- OS/COTS configuration security
  - Checklists for known systems (NIST-Win XP)
  - Check of version and known problems in DBs
  - Verify all unneeded stuff removed, box locked down
- Physical configuration security
  - Unused HW or ports removed or irreversibly disabled
  - Vulnerable but used ports locked / sealed
  - Verify security of scheme for applying locks/seals.

Verify claims in system documentation and do open-ended search for problems with all these.

Cost and Incentive Issues

- All this evaluation can get costly
  - Estimate a minimum of 2-3 weeks of highly skilled team members' time (see Maryland “red team eval”)
  - Cost could easily go over $100,000
    - Team of 4 at $200/hr for 2 wks = $64,000.
  - Good architecture may be able to reduce this
- Incentive and Financing Issues
- Availability of Reports
  - Free rider problems

Extra Details

Evaluating Communications Security

- Securing the Communications
  - Based on risks noted in other eval steps
  - Strong cryptography, good auditing, good protocols
  - Applies to networks, phones, memory cards, paper, etc.
- Securing the Box from Communications
  - Every communications channel into a machine is a potential avenue of attack
  - Remove services, lock down box, use firewalls and single-purpose protocols
- Again, open-ended evaluation needed

Evaluating Software Security

- Development Environment
  - Version control, security on development network, internal testing and review, coding standards, tools
- Testing
  - Automated testing for function, automated searches for known issues (buffer overruns, race conditions, non-cannonical representations, etc.)
- Review of Code
  - Adherence to coding and documentation standards
  - Open-ended search for problems

Evaluating OS/Configuration Security

- OS and other COTS software used:
  - Source code often not available
  - Even with source, no coding standards, development environment not known in detail, etc.
- Common approach: Checklists for securing OS
  - NIST provides guidance for securing Windows XP
  - General Rule: Remove everything not needed, lock down everything left
- Still need open-ended search for problems
  - Verify what documentation claims about installed services, drivers, etc.

Evaluating Physical Configuration

- Based on top-level evaluation, other evaluations, and voting system documentation
- Obvious stuff:
  - Block access to or remove unused H/W
  - Make sure design supports effective locks and seals
  - Verify claims in documentation, requirements from other evaluations
- Again, open-ended search for problems, not just checklist!