CS 591: Introduction to Computer Security

Lecture 2: Voting Machine Study Access Control

James Hook
Objectives:

- Review/Discuss Analysis of Diebold machine
- Introduce Access Control
Discussion

  – Reaction to the paper?
Discussion Questions

• What was the basic architecture of the voting machine?
• How did FHF steal votes?
• What other attacks did FHF consider?
• How did the viral propagation mechanism work?
Discussion Questions

- Is the analysis credible?
- Is the threat model credible?
- Is this representative of commercial systems today?
- Did Diebold follow best practices?
- Are the FHF results reproducible?
- Did Felton’s lab follow a round methodology in analyzing the machine?
Discussion Questions

• Having read the analysis of the Diebold machine, are you surprised that Sequoia used a threat of law suite to prevent Felten’s lab from analyzing their machine?

• Having seen this analysis of a fielded commercial system, are you more or less concerned about the discrepancies observed in Union County elections?
Discussion Questions

• Do you like Oregon’s vote by mail system?
Case Study

• We will use the FHF paper as a case study
• As we encounter concepts we will attempt to instantiate them in the context of the voting machine domain
Voting Machine Architecture

Touch Screen
Smart Card Reader
Audio jack
Removable Flash
Printer
On-board Flash
EPROM

Open
Key Access
Inside Box
Boot Process

- Boot device specified by hardware jumpers (inside box)
  - EPROM
  - on-board flash (default)
  - ext flash
- On Boot:
  - Copy bootloader into RAM; init hardware
  - Scan Removable flash for special files
    - “fboot.nb0” => replace bootloader in on-board flash
    - “nk.bin” => replace OS in on-board flash
    - “EraseFFX.bsq” => erase file system on on-board flash
  - If no special files uncompressed OS image
  - Jump to entry point of OS
• On OS start up:
  – run Filesys.exe
    • unpacks registry
    • runs programs in HKEY_LOCAL_MACHINE\Init
      – shell.exe (debug shell)
      – device.exe (Device manager)
      – gwes.exe (graphics and event)
      – taskman.exe (Task Manager)
  – Device.exe mounts file systems
    • \ (root): RAM only
    • \FFX: mount point for on-board flash
    • \Storage Card: mount point for removable flash
Boot (continued)

• Customized taskman.exe
  – Check removable flash
    • explorer.glb => launch windows explorer
    • *.ins => run proprietary scripts
      – (script language has buffer overflow vulnerabilities)
      – used to configure election data
    • default => launch “BallotStation”
      – \FFX\Bin\BallotStation.exe
BallotStation

- Four modes: pre-download, pre-election testing, election, post-election
- Mode recorded in election results file
  - \Storage Card\CurrentElection\election.brs
Stealing Votes

• Malicious processes runs in parallel with BallotStation

• Polls election results file every 15 seconds
  – If election mode and new results
    – temporarily suspend Ballot Station
    – steal votes
    – resume Ballot Station
Viral propagation

• Malicious bootloader
  – Infects host by replacing existing bootloader in on-board flash
  – subsequent bootloader updates print appropriate messages but do nothing

• fboot.nb0
  – package contains malicious boot loader
  – and vote stealing software
Access Control Model
Objectives

• Introduce the concept of Access Control
• Relate mechanism to Confidentiality, Integrity and Availability
Articulating Policy

• How do we articulate a security policy?
• How do we provide mechanisms to enforce policy?
• Voting
  – Different individuals in different roles
    • Voter, Poll worker, ...
  – Different actions
    • Vote, define ballot, start and stop election, ...
  – Logical and physical entities
    • Ballot, stored tally, final tally, voting machine, removable flash, on-board flash, ...
Ad hoc policies

• Discus
  – Only voters should vote
  – Only poll workers should start and start elections
Access Control Matrix Model

- Lampson ‘71, refined by Graham and Denning (‘71, ‘72)
- Concepts
  - **Objects**, the protected entities, O
  - **Subjects**, the active entities acting on the objects, S
  - **Rights**, the controlled operations subjects can perform on objects, R

- **Access Control Matrix**, $A$, maps Objects and Subjects to sets of Rights
  - State: $(S, O, A)$
Voting: Subjects, Objects, Rights

• Subjects: (Roles)
  – Voter, Poll worker, ...

• Rights: (Actions)
  – Vote, define ballot, start and stop election, ...

• Objects: (Logical and physical entities)
  – Ballot, stored tally, final tally, voting machine, removable flash, on-board flash, ...

• Question: Is every voter a subject? Or is the role of voter a subject? One-person-one-vote?
Exercise

- Sketch Access Control Matrix (ACM) for Voting

<table>
<thead>
<tr>
<th></th>
<th>Ballot</th>
<th>Stored Tally</th>
<th>Final Tally</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voter</td>
<td>read</td>
<td>increment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Poll Worker</td>
<td></td>
<td></td>
<td>print</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Questions

• What about modes?
  – Once the election starts the ballot should not change
  – Voters should only vote when the election is happening
Questions

• Levels of abstraction
  – Some objects are physical, some are logical
  – When considering the programming model you now have processes and files (and possibly modes of operation)

• Exercise:
  – Sketch ACMs with processes as subjects and files as objects for voting and post-election modes
Exercise

• Compare the ACMs for files and processes with the original ACM
• Is every operation specified in the original feasible in the refined ACMs?
• Is every feasible operation in the refined ACMs allowed in the original?
Mechanisms

• Policy specifies abstract goals
• Mechanisms are concrete devices, algorithms, or processes that assist in implementing a policy
• For example, passwords are a mechanism that can support an authentication policy
  – Mechanisms are not always perfect!
Access Control Mechanisms

• Most operating systems provide some mechanisms for supporting access control

• Typically:
  – Processes are associated with users (or user identification numbers), which are the subjects
  – Files are objects
  – Rights are: read, write, append, execute, search, ...

Applying the Mechanism

• Can a generic Access Control mechanism help make the Voting machine more trustworthy?
• What about modes?
  – Mode is not part of typical AC mechanisms
  – However rights can be changed, but this is heavy weight
  – Analysis of systems that actively change rights is potentially difficult
Limitations on Mechanisms

• Simple mechanisms are preferred
• All computational mechanisms must be decidable
• In general, useful mechanisms must be computationally cheap
Limitations on ACM

• Given an ACM mechanism with dynamic rights management, can a software tool say yes or no, in all cases, to the question:
  – Does object O ever acquire right r?
• In 1976 Harrison, Ruzzo and Ullman showed that
  – in general this is an undecidable problem
  – In restricted cases it is decidable
• These results are presented in Bishop Chapter 3
Why should I care?

• Although the specifics of the Bishop recounting of the HRU results may seem tedious, the take home message is critical:
  – Security is a non-trivial property of computer systems
  – Any reasonably expressive security mechanism when coupled with a general purpose programming system will lead to undecidable language problems
  – There will never be a post-hoc “lint-like” tool that takes a security spec and an arbitrary program and definitively says “secure” or “insecure”
So can I give up?

- Don’t give up!
- Build security in from the start
- Design systems so that security properties are manifest
- Use simple mechanisms, like access control, in straightforward ways
- Architect for verification and validation
Returning to Access Control

• Is Access Control biased to
  – Confidentiality
  – Integrity
  – Availability

• Exercise
  – Develop scenarios in which a confidentiality (integrity, availability) property is expressed using an access control matrix
Model vs. Mechanism

• Earlier I presented the model of the AC Matrix
• Does UNIX implement the full AC Matrix?
  – What key simplifications does UNIX adopt?
  – Why?
• Is the full ACM mechanism a good idea?
  – Is it a good model?
A Good Model

- ACM is a good model because any mechanism of compatible granularity can be described in terms of how it approximates the ACM model