A Methodology for Assessing Procedural Security: A Case Study in E-Voting

Komminist Weldemariam
&
Adolfo Villafiorita

email: sisai@fbk.eu, adolfo@fbk.eu

The 3rd International Conference on E-Voting, 2008
Introduction
Currently, in Italy voting is paper-based

Trentino is an autonomous Region

In 2003, the Local Government decided to automatize the electoral procedures

The ProVotE Project started
The prototype

- **DRE VVPAT**
- **Components:**
  - Touch screen
  - Printer with cutter
  - Smart card reader
  - Signal system
  - Removable memory support
  - UPS
  - Ballot box

Remark: remote voting is illegal in Italy
E-Voting elections

4 trials
+2 with legal value
+20 precincts

11500 voters

Trento, Fondo, Coredo
Lomaso, Baselga di Pinè
Daiano, Peio, Cavedine
Campolongo al Torre, Tapogliano

10 Municipalities

8 mag 200
6 nov 2005
28 mag 2006
5 nov 2006
20 nov 2006
25 nov 2007
## Results

<table>
<thead>
<tr>
<th>Event</th>
<th>Election Type</th>
<th>Date</th>
<th>Value</th>
<th>Municipalities</th>
<th>Precincts</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Election</td>
<td>Town up to 3000 and with more than 3000 inhabitants</td>
<td>8 may 2005</td>
<td>Trial</td>
<td>Baselga di Pine, Coredo, Fondo, Lomaso, Trento</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Election</td>
<td>Town up to 3000 inhabitants</td>
<td>6 nov 2005</td>
<td>Trial</td>
<td>Daiano</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Election</td>
<td>Town up to 3000 inhabitants</td>
<td>28 may 2006</td>
<td>Trial</td>
<td>Peio</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Election</td>
<td>Town up to 3000 inhabitants</td>
<td>5 nov 2006</td>
<td>Trial</td>
<td>Cavedine</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Election</td>
<td>Students representatives election</td>
<td>20-21 nov 2006</td>
<td>Legal</td>
<td>Liceo Leonardo Da Vinci di Trento</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Referendum</td>
<td>Consultative Referendum</td>
<td>25 nov 2007</td>
<td>Legal</td>
<td>Campolongo al Torre (FVG) Tapogliano (FVG)</td>
<td>2</td>
<td>Evento</td>
</tr>
</tbody>
</table>

- **Different election types**
- **On more days**
- **Trials (both paper & EVS) and only electronic voting system**
- **More municipalities at the same time**
- **Collaborations with other autonomous regions**
The ProVotE project goals

- To provide the Trentino's e-voting technological solution
- To ensure a smooth transition from paper-based to e-voting (reducing the digital-divide)
- Social studies
- *Electoral laws & regulations*

**Scenario:**
Full automation of the voting “chain”
(from expression of the vote to dissemination of the results)
Full automation of “accessory” functions
(registration, paper logs, …)
Procedural security context
Issues

• Switch to e-voting is a long and difficult process in Italy:
  – Strict security and traceability requirements
  – Relatively strong resistance (high sensibility)
    • by citizens (e.g. doubts related to anonymity)
    • by politicians (e.g. trust)
    • by public administration (e.g. alteration of data)
Expectations

- Understand vulnerabilities
- Mitigate *actual* and *perceived* threats
- Describe the procedures in the law in an accurate way
- Increase procedural transparency
Approach

- Which problems are solved with e-voting?
- Which new problems are introduced? Is it possible to introduce new procedures to tackle them?
- Which threats are independent from the voting system?
Security Modeling & Analysis

• Various “state of the art of security computing”
  – Focus on avoidance, detection, prevention, recovery, ...
  – Shown useful for zeroing the security lacks of the software systems under analysis

• Various Red-Team approaches (e.g. the California Top-Bottom Voting System Review)
  – Emphasize how the goals of an attacker can be successfully achieved
  – Identify, model, assess and document the sequence (or coordination) of threat actions
Characterizing aspects

- **Asset mobility**: assets and sensible data related to an election are handled by different actors with different responsibilities over time in *different locations*:
  - Asset *mobility* is intrinsic in the domain: without it, you could not have elections

- **Asset evolution**: assets related to a system (or an election) change their *value* over time;

- **Number of instances**: various election assets need to be replicated (e.g. pin code)

- **Non digital assets**: essential for various tasks (e.g. documents for auditing)
Procedure Matters!

- In “procedurally rich” scenarios, software systems are just part of a complex organizational setting!
- A vulnerability in the procedures:
  - Makes it possible to perform operations that violate the security policies of the system;
  - May compromise the CIA properties of non-digital assets;
- Security achieved by strict definition of procedures and controls embedded in procedures.
- Example: “is the practice of electronic voting machine "sleepovers" appropriate?”
Previous work

- Assets-centered procedural analysis: procedures as assets transformation functions
- Asset-state: the asset condition at a given procedural step
- Asset-flow: formalism to represent the asset evolution
- Location: asset environment at a given procedural step
The methodology
Motivation of this work

• There is no structured approach in security assessment that considers security procedures as a target for security analysis;

• The existing approaches are not very effective in what we call procedurally rich systems;

• There is a high demand for systematic procedural security analysis for e-voting systems to discover where incorrect or malicious deviations from the procedures defined by the law may result in fundamental security and privacy features of the system.
What is the main goal?

• The usage of techniques and tools to understand the impact and effects of procedural threats:
  – In particular, the actions that can take place during the execution of the procedures, and which are meant to alter assets manipulated by procedures in an unlawful way
A procedural threat is a breach of security objectives during the execution of a procedure;

Kinds of attacks:
- Attacks against digital assets (item 1 & 3)
- Attacks against other kinds of assets (item 2 & 4)
A Structured Methodology

**Step 1. Model Procedures**

- **Step 2. Extend Model**
  - **Step 3. Encode Asset Flows**
  - **Step 4. Encode Properties**
    - **Step 5. Perform Analyses**
    - **Step 6. Analyse Results**
Step 1: model procedures

- The procedural business model:
  - Describes the processes to be analyzed;
  - Defines how assets are elaborated and transformed;
- The model contains information such as:
  - Assets used, their features, the actors and their role when participating to the workflows.
- A subset of UML diagrams is used:
  - To ease the task of translating the models into executable specifications
  - To ease the communication with the domain experts
Step 1: example

Procedure

Asset

Asset (state changed)
Step 2: extend model

- A **threat-action** alters some features of an asset (allows some privileges, change its value, ...)
- The **extended model** is built “injecting” some threat-actions in the nominal flow of the procedures;
- The effectiveness of the analysis depends on:
  - *The definition of threat-actions*
    We allow to the attackers all possible privileges and action to an asset
  - *The injection strategy*
    The most general is to inject all possible threat-actions at all possible procedural steps
Step 2: example

Original

Extended
Towards step 3: MC

- Model Checking (MC) is a formal verification technique to verify **reactive systems** (e.g. safety-critical, real-time embedded, ...)
  - Step 3.1: encode the model
  - Step 3.2: encode properties

- **NuSMV**: symbolic model checker (http://nusmv.irst.itc.it/)
  - Synchronous and asynchronous finite state systems
  - Computation Tree Logic (CTL), Linear Temporal Logic (LTL)

Figure: A. Cimatti and M. Pistore, ESSLLI, July 5-9, 2002, Trento (Italy)
Reactive system

- Reactive systems can be represented with:
  - State variables
  - Initial values for state variables
  - Transitions among states

  For a formal definition see “Kripke structure”.

- For convenience, complex systems can be combined by sub-systems (called **modules** in NuSMV)
SMV language

As data types, the SMV language provides:

- boolean (a : boolean)
- enumerative (a : {empty, full})
- bounded integers (a : 1..7)

MODULE simpleModule (...)
VAR
  value : boolean;
ASSIGN
  init(value) := 0;
  next(value) := !value;

MODULE main
VAR
  a : process simpleModule;
  b : process simpleModule;
[...]

Some constraints can be added

Some parameters can be added

value = 0

value = 1
Step 3: encode asset flow

- A “program counter” module ensures the procedures are executed according to the model order.

- **A module per asset;**
  - The “state” variable encode how asset properties change during the execution a procedure.

```plaintext
MODULE electionResult ( ... )
VAR
  state : {plain, unsigned, signed, signed_&_encrypted};
  content : {null, data, encrypted_data, garbage};
  location : {polling_place, electoralOffice}
ASSIGN
  init(content) := null;
  next(content) := case (content = data && pc.pc = encrypt_proc) : encrypted_data
```

Step 3: encode asset flow

- “Accessory” information

  Actors responsible for different activities can be used to express security properties.

  DEFINE

  ElectoralServiceActive := pc.pc = CloseVoting || [...]

Step 4: encode properties

- Some classes of **security properties** to understand:
  - **Undetected attacks**: attacks which succeed in altering one or more assets and for which the procedures provide **no check to highlight the alteration**;
  - **Denial of Services**: attacks which alter one or more assets in such a way that procedures have to be stopped.
- The security properties are specified in the LTL / CTL temporal logic
Step 4: example

- "It is never the case that poll officers receive an altered election software".

AG ! (electionResult.content = garbage & electionResult.location = electoralOffice)
Step 5: perform analyses
Step 6: analyze results

- To understand what are the hypotheses and conditions under which a given security goal is achieved or breached;
  - Are single attacks managed correctly?
  - Are there reasonable combinations of attacks which result in a major problems?
- To provide information to try and modify the existing procedures, so that security breaches can be taken care of.
  - Does the change or the introduction of some procedures solve the problem?
Conclusion

- **Security modeling and analysis motivation**
  Since procedures in election defines how the election is to be run by humans, they should be as important as technical security features of the election systems;

- **Procedural security is as essential as technical security to increase the security of E-Voting systems;**

- **We presented a structured methodology to perform procedural security analysis** and a threat injection strategy based on modeling procedures, injecting threat actions and using verification techniques;

- **A simple example taken from real eVoting trials was used to test the approach.**
Future work

- **Automation**
  Devise techniques and automatic tools to analyze security threats at procedural level;
  *(e.g. algorithms to automatically generate specification and perform threat injection)*

- **Standardization** of various aspects of the property and threat representation;

- **More and bigger experiments** to evaluate the significance of the proposed approach and the computational requirements;
  *(e.g. how many assets/procedures can be analyzed within reasonable time-limits?)*

- Further analyze the **concept of location**;

- Deal with the **number of instances** problem.
  *The procedures describe the behavior on a single asset (e.g. voting machine). What problems should be faced if we scale to consider more asset of the same type?*
Thank you!