Modeling and Analysis of Procedural Security in (e)Voting: the Trentino's Approach and Experiences

Komminist Weldemariam[1,2]
[2] *University of Trento*
Via Sommarive, 14 Povo (TN) 38100
Trento, Italy
sisai@fbk.eu

Adolfo Villafiorita[1]
[1]*Fondazione Bruno Kessler*
Via Sommarive, 18 Povo (TN) 38100
Trento, Italy
adolfo@fbk.eu

Abstract

This paper describes the experiences and the challenges we are facing within the ProVotE project, a four years project sponsored by the Autonomous Province of Trento that has the goal of switching to e-voting for local elections. One of the activities we are carrying out within ProVotE is the systematic analysis of the weaknesses and strengths of the procedures regulating local elections in Italy, in order to derive possible attacks and their effects. The approach we take is based on providing formal specifications of the procedures and using model checkers to help us analyze the effects of attacks. We believe such an analysis to be essential to identify the limits of the current procedures (i.e. under what hypotheses attacks are undetectable) and to identify more precisely under what hypotheses and conditions we can guarantee reasonably secure electronic elections. This paper presents the methodology and the techniques we are devising and experimenting with to tackle problem highlighted above.

1 Introduction

The organization of an election in Italy involves various offices of the Public Administration and private contractors, has a time-span of months, and has strict security and traceability requirements. Sensibility by citizens and politicians is very high, and litigation over, e.g., implementation of procedures and validity of results are not uncommon. As an example, one of the major candidates at the last general elections in Italy distributed — via the website of his party — a leaflet describing possible misconducts of poll-officers during elections and some counter-measures. The goal of the leaflet was help ensuring that poll-officers of the opposing party could not try and illegally alter the results of the election.

We are involved in a project named ProVotE [1] that it is related to the evaluation and possible introduction of e-voting for local elections held in the Autonomous Province of Trento, Italy. The e-voting system developed within the project has been used with experimental value by more than ten thousand citizens and with legal value in two small elections, making the initiative the largest experimentation of e-Voting in Italy, so far. The switch to electronic elections in Italy, however, is a long and difficult process, that requires extreme attention to all the aspects characterizing an (electronic) election, including a thorough understanding of the limits and the security breaches that are inherent in the procedures and/or that derive from a combination of procedures and systems adopted.

Therefore, one of the concerns of our work is understanding how the switch to the new technology changes security risks, with the ultimate goal of defining the laws and the procedures regulating electronic elections, that guarantee the same level or a higher level of security. Interestingly, paper voting and the procedures regulating paper elections are not immune to attacks, which can usually be carried out under the hypothesis of multiple "failures". For instance, if a ballot is stolen before the election and the polling officers do not report it, it is possible to control how voters vote in a polling station. The usage of electronic devices, however, shifts and "reshapes" some of the risks (see, e.g., [2, 3, 4, 5, 6, 7]).

Various approaches for specifying, modeling, analyzing, and assessing security have been proposed in the past and have been proven useful for zeroing the security lacks of the software systems under analysis (see, for instance, [8, 9, 10, 11]). However, these techniques are less or otherwise not effective in what we call *procedurally rich* scenarios, namely situations in which software systems are just part of a more complex organizational setting, in which procedures have to be executed on security-critical assets (belonging both to the digital and to the physical realm). This is exactly the case of (voting and) e-voting, in which even in those countries that have adopted a high level of automation, the execu-

tions of procedures and controls, carried out by people on physical assets (e.g. printouts of the digital votes), remains a necessary and unavoidable part (for instance, see detail analysis and evaluation report of the EVEREST project [7]).

We are approaching the problem by reasoning about the procedures and controls that regulate the usage of e-voting systems. We do so by providing formal models of the procedures, by "injecting" threats in such models and by analyzing, through the help of model checkers, the effects of such threats. We believe such an analysis to be essential to, first, identify the *security boundaries*, that is, the conditions under which procedures can be carried out securely and, secondly, to devise a set of requirement to be applied both at the organizational level and on the (software) systems used.

This paper is structured as follows. In the next section we introduce the ProVotE project, within which this work has been developed. In Section 3, we describe the context of procedural security in detail. In Section 4, we describe our methodology for procedural security analysis and illustrate the approach with snippet example. Section 5 discuses some related work and finally we draw our conclusion in Section 6.

2 The ProVotE Project and Motivations

ProVotE [1][1], a project sponsored by Provincia Autonoma di Trento (PAT), has the goal of ensuring a smooth transition to e-voting in Trentino, eliminating risks of digital divide and providing technological solutions which support, with legal value, the phases ranging from voting to the publication of the elected candidates.

The project includes partners from the public administration (PAT, Regione Trentino/Alto-Adige, Consorzio dei Comuni Trentini, Comune di Trento, IPRASE), research centers and academia (Fondazione Bruno Kessler (FBK), Faculty of Sociology of the University of Trento, Fondazione Graphitech), and local industries (Informatica Trentina) and is co-led by the Electoral Service of the Autonomous Province of Trento and by FBK-IRST. Project leadership by the Public Sector, in our opinion, among other advantages, helps tackling the issue of potential conflicts of interests by private industries; see e.g. [12]. The technological solution (both software and some hardware components) has been developed in house, providing integration with some commercial components.

The project is multi-phased and is organized in various lines of activities that strictly interact. For instance, in the first phase of the project, some functional and non-functional requirements of the e-voting prototype have been provided with a strict round-trip between the sociological and the technological line, with the normative line ensuring compatibility with the electoral laws. See [1, 13] for more details and [14] for some considerations related to the sociological aspects of e-voting in Europe. Development proceeds incrementally, and each phase defines milestones to check the goals set in each different line.

The first phase had the goal of testing prototypes, evaluating acceptance by citizens, ease of use, and some organizational aspects. Verification of the results achieved in the first phase was conducted through four different trials (May 2005, November 2005, May 2006, November 2006) held during local elections. During the experimentation polling stations were equipped with one or more e-voting machines and citizens were asked to vote on paper, repeat their vote using the electronic systems, and provide feedback about the system to interviewers. About ten thousand citizens took part in the first phase of the project. (With the first experimentation having the lowest participation, due also to the fact that we had one e-voting machine per polling station. In subsequent experimentations we equipped most polling stations with two machines and got higher participation[2]).

During the second phase of the project we used the electronic systems in two elections, with legal value. The first election was the election of student representatives in a local high school and it involved 1,298 students. The second election — conducted in the towns of Campolongo al Torre and Tapogliano in Friuli-Venezia Giulia (November 2007), a neighboring region with autonomy similar to that of PAT — was a poll to unify the two municipalities; 561 people used the systems. In both cases, logistics, procedures, and laws governing the elections were relatively simple and can be considered a simplified version of the other kinds of elections we intend to use our systems for.

For the third phase of the project, which will lead to a large-scale introduction of the new voting system, aspects related to procedures, logistics, and organization become more relevant, as they will serve both as the basis for the deployment of the solution and for the definition of the laws that will govern the electronic election.

To our knowledge, little has been done in Italy to try and introduce e-voting. We are aware of one eletronic election, with legal value, in Valle d'Aosta, that it involved a small number of voters. The election remained an "isolated" case and little or no information is available on the matter. With respect to scope, population, and participation, ProVotE is among the largest, if not

[1] http://ed.fbk.eu/index.php/EVoting/HomePage

[2] Detailed results of all the experimentations and elections conducted within the ProVotE project are available on the Internet at: http://www.provincia.tn.it/elezioni and http://referendum2007.regione.fvg.it/index.html.

the largest, e-voting project in Italy.

3 Procedural Security Analysis

Procedures in an election system are best practices mandated by law locally (within the province) or at the country level. These practices are intended to ensure that elections is carried out correctly and securely. From a practical point of view, procedures are often as important as the technical security features of the systems used in elections, since procedures define how critical assets are to be managed, elaborated, and transformed.

During elections, incorrect or malicious "deviations" from the procedures defined by the law may results in violations of fundamental rights of citizens (e.g. secrecy of vote) or, even, in threats to the integrity of electoral data and of the election results. Lawmakers, therefore, need to carefully consider and analyze what happens when a procedure is not followed as prescribed and have to define mechanisms to ensure that violations can be detected.

We are interested in providing methodologies and tools to help assessing security of procedures and the effects of deviations from the "nominal" behaviors, with the goal of highlighting security vulnerabilities. *Procedural security*, therefore, deals with the identification, modeling, establishment, and enforcement of security policies about the procedures that regulate the usage of a system and system processes. The breach of security objectives during the execution of the procedures is known as *threat to the procedures* (or *procedural threats*). We call *procedural security analysis* the process of understanding the impact and effects of procedural threats, namely courses of actions that can take place during the execution of the procedures, and which are meant to alter, in an unlawful way, the assets manipulated by procedures.

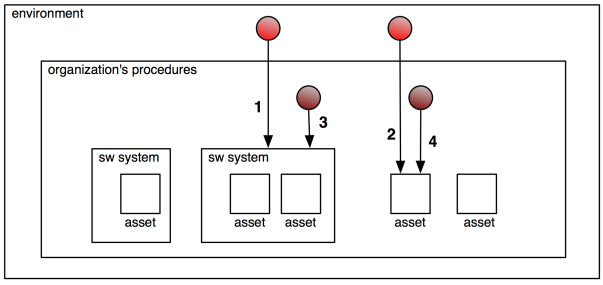

Figure 1: Procedural Security Analysis

The situation is depicted in Figure 1. Our *target of evaluation* is a (complex) organizational setting in which procedures transform and elaborate *assets*, which may not necessarily be just digital (e.g. a printed ballot). The procedures are meant to add value to the assets and to protect them from attacks, which can either come from external sources or from insiders. In particular, we distinguish the following kinds of attacks:

1. *Attacks on digital assets* (item 1 and item 3 in Figure 1). These kinds of attacks are meant to alter one or more of the digital assets of an organization. Attacks can either be carried out from external sources (the environment) or from internal sources. Opportunities for attacks are determined by the organizational setting and by the security provided by the digital systems.

2. *Attacks on other kind of assets* (item 2 and item 4 in Figure 1). These attacks are meant to alter one or more of the non-digital assets of an organization. Attacks can either be carried out from external sources (the environment) or from internal sources or a combination of both that it forms coordinated attacks. Opportunities for attacks are determined by the organizational settings only.

Security assessment (like [8, 10]) usually focuses on understanding items 1 and 3, namely, types and effects of attacks on (software) systems. In the next section we propose a tool-supported methodology to tackle also points 2 and 4 above, namely types and effects of attacks on assets that are not (necessarily) digital and that derive from the way in which procedures are implemented and carried out.

4 The Methodology and a Case Study

4.1 The Methodology

We devise the following methodology to perform our analyses (See also Figure 2):

The first step consists of *providing (business) models of the procedures* under evaluation (Step 1 of Figure 2). These models are meant to describe the process or the processes to be analyzed, and to define how assets are elaborated and transformed by such procedures. In order to ease the task of translating the models into executable specification, we decided to stick to a subset of the UML, that allows us to describe the domain concepts in a strict and defined way (see [15]). The model contains information about the procedures (workflows), the assets used, their features, and the actors and their role when participating to different workflows.

So far we managed to provide UML models of the electoral procedures in place in the Autonomous

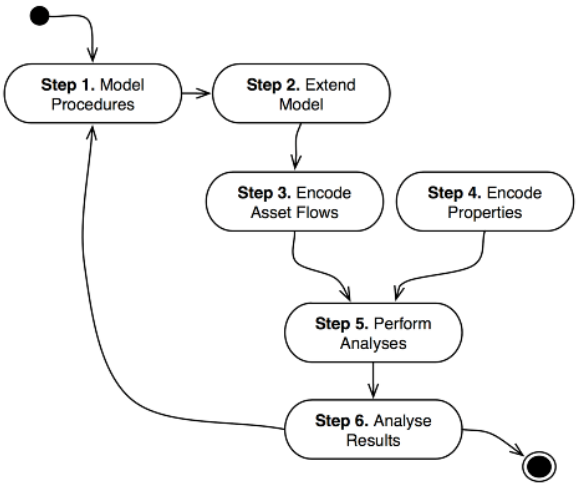

Figure 2: The process of formal procedural security.

Province of Trento and in Regione Friuli Venezia Giulia. We use Visual Paradigm [3] as our reference modeling tool. See [16, 17] for more details about the notation, tool support, and the model themselves.

The second step consists of *"injecting" threat actions* into the model and generate what we call *extended model* (Step 2 of Figure 2). A threat is an action that alters some features of an assets or allows some actors privileges (e.g. a "read" privilege) on some assets. Thus, in the extended model, not only assets are modified according to what the procedures define, but they can also be transformed by the (random) execution of one or more threat actions. Since attacks depend on what threat-actions are carried out, the effectiveness of the analysis depends upon the threat actions that are defined and the injection strategy that is chosen. With respect to the first point (threat action), we allow attackers any possible privilege and action to assets. With respect to the second point, it turns out that the best and most general strategy is that of injecting all possible threat-actions at all possible steps of the nominal procedures (the model checker will then take care of "pruning" useless threats, namely threats which do not lead to any successful attack). The construction of the extended model, whose generation can be automated, is currently performed by hand.

The third step is *encoding the asset model into executable specification* (Step 3 of Figure 2). From the extended models defined at the previous step we derive executable specification that define how each asset is manipulated by the procedures. Asset flows are represented in the NuSMV input language [18]. The NuSMV model of the asset flows is based on the definition of "program counters" that ensure that procedures are executed according to the specifications, and by defining one module per asset with one state variable per asset-feature. The state variables encode how features change during the execution of the procedures. Accessory information, such as actors responsible for the different activities can be used, e.g., to express security properties, which, in turn, are verified against the asset model.

The fourth step is *specifying security properties to model check*. The specification of the desired (procedural) security properties, namely, the security goals that have to be satisfied are then encoded using LTL/CTL formula. Asset flows and security properties are then the input to NuSMV.

In step five (Step 5 of Figure 2) we *perform analyses*, namely, we run the model checker and collect results (counter-examples). Counter-examples of security properties found by the model checker encode sequences of actions that, if executed, pose a threat to security of one or more assets. In standard situations, the counter-example will contain the execution of one (or more) asset threat. A counter-example in which no asset threats need to be executed would show an inherent weakness in the *nominal* workflows or, otherwise, an error in the specifications.

The last step consists in *analyzing the obtained results* (Step 6 of Figure 2), by looking at the counter-examples generated by the model-checker. In particular some of the information we are interested in is:

1. The **Actor-Play-Role** namely, the roles actors have in the execution of the attack and the privileges they get on assets.
2. **Reachability** the sequence of actions leading to the violation of a security goal, with particular respect to the execution of asset-threats.

This step allows us to achieve two goals. First, it allows us to understand what are the hypotheses and conditions under which a given security goal is achieved or breached. Second, it provides information to try and modify the existing procedures, so that security breaches can be taken care of.

The analysis approach we take is very similar to that of FSAP/NuSMV-SA [19], for safety analysis, whereby a system specification is "enriched" with information about faults and analyzes are carried out to understand the effect and impact of faults on safety requirements expressed in the form of LTL/CTL formulae. Analogously to what happens in safety analysis when analyzing, e.g.,

[3] http://www.visual-paradigm.com/

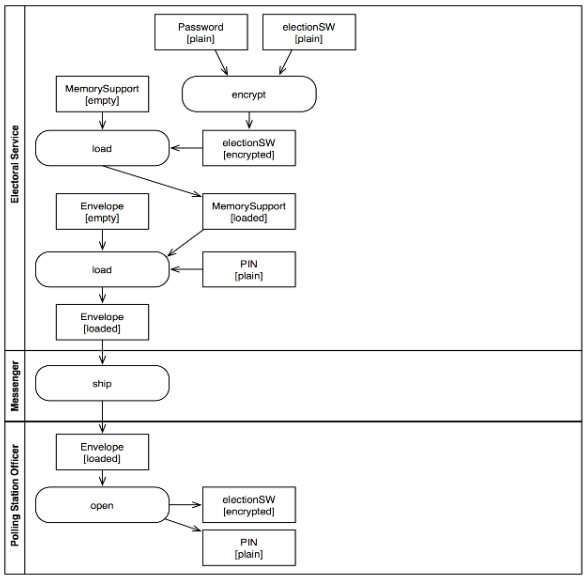

Figure 3: An example of asset flows

the loss of a critical functions, enhancing the procedures results in reducing the probability of an attack or making the attack more complex, rather than eliminating it.

4.2 The Case Study

In this section we illustrate the methodology through a simple snippet example, which is derived from the procedure in place for e-voting experimentations. Note that this work is the extension of the work presented in [15].

4.2.1 Model Procedures and Inject Threats

A subset of UML diagrams is used to model (an excerpt of) the procedure that is followed during project trials for delivering the voting software to the polling stations (See Figure 3). The diagram shows how, before the election, the Electoral Office encrypts the e-voting software and creates a memory support which contains the final software release. The responsible person at the Electoral Office then prepares an envelope with the PIN code (that it is used to activate the voting functions) and the memory support. A messenger (e.g. a police officer) takes the envelope and delivers it to the polling station, where the polling officers, once verified that the enveloped is sealed, open it, insert the memory support in the voting machine, insert the PIN and start the voting operations.

In order to analyze possible threats to this procedure, we inject threats into the model of the procedures and generate the extended model. Figure 4 depicts the extended model resulting from the injection of some *delete* and *replace* threat actions in the example of Figure 3 (threat actions are stereotyped with the "threat-action" stereotype and marked in color). Note that the semantics of delete and replace actions may slightly vary when applied to different kinds of assets.

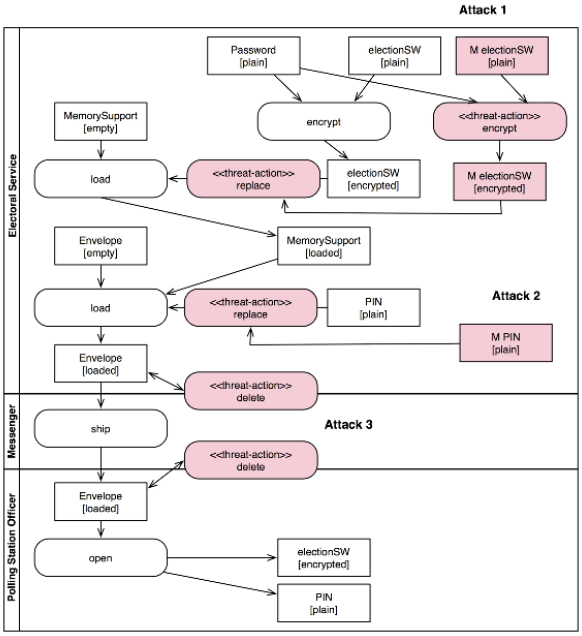

Figure 4: An example of extended model

4.2.2 Model the asset-flow in NuSMV

The next step is modeling the asset flows into executable NuSMV specification. In particular, the translation is performed as follows:

1. we define a module that "works" as the "program counter", that is, it encodes the sequence of actions defined by the workflow. The "program counter" ensures that the order in which activities are executed (let them be "nominal" or the execution of "threat actions") is the one defined by the UML diagrams.

2. we define a module for each asset that it is specified in the UML diagram. Assets have features (properties): each feature of the asset is defined as a state variable of the module encoding the asset. The transition from one asset state to the next is determined by the "program counter" (which represents the execution of an action of the workflow) and some accessory information such as role(s) (which encodes the relationship between the workflow activity and actors who are assigned to perform the execution of the workflow).

3. finally, we define some boolean variables to capture malicious asset flows and the execution of threat actions.

For instance, the following snippet of code defines the asset type `electionSW` and some of its features, named `status` (that is, the states in which the `electionSW` can be), `value` (the relative weight of an asset assigned based on the criticality of the asset, basically the analyst decide and assign this value), and `content` (that is, the qualitative value of the `electionSW` can be).

```
MODULE electionSW ( ... )
VAR
status  : {plain, encrypted};
value   : {noValue, lowValue,
highValue,inf};
content : {sw,esw,eesw};
```

Similarly, other modules with their corresponding feature variables are declared (e.g., `MODULE Key(...)`, `MODULE memorySupport (...)`). "Accessory" information (not strictly necessary to execute the workflows), such as the actors responsible for each activity, is encoded in the model through `DEFINE`s in the main module, such as in the following snippet:

```
DEFINE
 ElectoralServiceActive := pc.pc  =
 loadMemSup || [...]
```

Evolution of assets' properties are encoded using state machines, which are encoded in NuSMV with the `next` construct (which specifies the value of a variable at step $n+1$, given the value at step n). Notice that asset flows are defined both in terms of the "program counter" (e.g. the current step of the workflow) and the value of the asset features. Figure 5 shows a model of feature `content` variable of `electionSW` asset where it's state changes according to the program counters and some other variables; its corresponding snippet NuSMV code is also given below:

```
[...]
next(sw.content) := case
(pc.pc = encrypt &&
 content =  sw) ||
(content = eesw &&
 pc.pc =  openEnvelope &&
 POfficersActive)
:esw;
(pc.pc = loadEnvelope &&
(TechnicianTwoActive ||
ElectoralServiceActive) && content
```

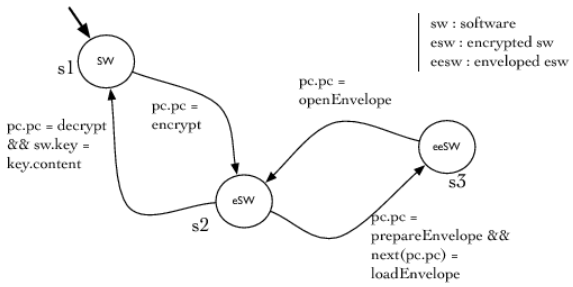

Figure 5: A simple example of state transition model for `content` feature of `electionSW`.

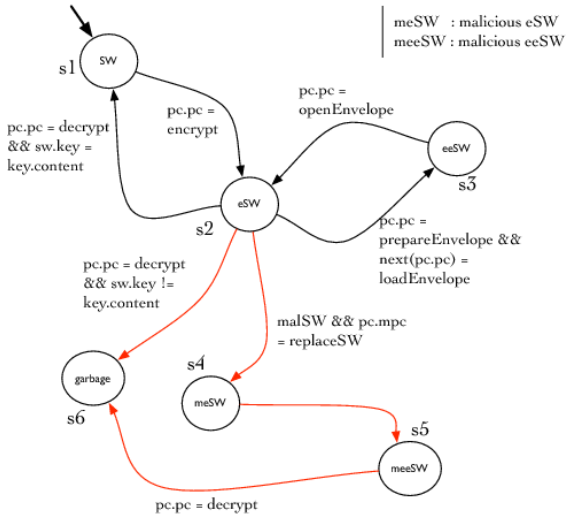

Figure 6: An extension of Figure 5.

```
= esw)
:eesw;
pc.pc =decrypt &&POfficersActive
&&(status = encrypted
|| content = esw) && !fakeKey
:sw;
[...]
```

Note that in the code above we have left some detail specification (such as location) for the matter of presentation purpose.

Threat injection (model extension) corresponds to augmenting the state machine of the asset flow with new transitions corresponding to the execution of threat actions. Figure 6, for instance, shows an asset flow with some threat actions that may alter a feature of an asset (e.g. `content`), in some undesired way.

The triggering of a threat action is "monitored" through boolean variables that are set to true when the

action takes place, as illustrated by the following pieces of code. We first declare one boolean variable per threat:

```
can_mesw, can_meesw: boolean;
can_garbageSW: boolean;
can_mPPin, can_mePPin: boolean
```

The above variables are initially set to false. When a variable is set to true (either because constrained to do so by the model or, more often, at random[4]), a transition in the state machine encoding the asset flow is triggered and the value of the asset flow changed according to the threat-action (rather than to the nominal flow), as illustrated by the following piece of code:

```
next(can_mesw) := case
(malS && pc.mpc = replaceS &&
(next(pc.pc)=loadMemSup))
|| (can_meesw && pc.pc =
openEnvelope)
:1;
1: can_mesw;
esac;
```

4.2.3 Represent properties and Model Check them

After encoding the relevant information of asset-flows, we specify security properties using LTL/CTL formulae. We use LTL (Linear Temporal Logic) to reason on the computational path scenarios of an asset (e.g., what can happen as asset travels along different locations) and CTL (Computational Tree Logic) to reason about the existence of specific states (e.g., is there any particular state in which an asset can be altered in an undesired way).

In the example we have shown (in the extended model), for instance, it is possible to implement at least three different attacks. The first one consists in replacing the software which is sent to the polling stations. By reading the `key` (or name it also `password`) with which the `electionSW` is encrypted and substituting a modified version of the software in the `MemorySupport` it is possible, for a malicious actor, to deliver a modified copy of the software to the polling station. The second one consists in replacing the `PIN`. A malicious actor with access to the `PIN` code may substitute the `PIN` which is loaded in the `envelope` and thus, have a wrong `PIN` delivered to the polling station (eventually causing a denial of service — namely the voting functions cannot be activated by the polling officers). The third attack consists in deleting (or destroying) the `envelope` during transportation, possibly causing another denial of service.

Below we show two examples of properties that allows us to highlight such attacks.

[4]Corresponding to modeling that threats can take place at any time

	t_0	t_1	t_2	t_3	t_4	...	t_9	t_{10} ...
pc.pc	—	—	encrypt	loadMem	preEnv	...	openEnv	decrypt
sw.content	sw				garbage	...		
malSW	$\bot$		$\top$	$\bot$		...		
replaceSW	$\bot$		$\top$	$\bot$		...		
sw.can_mesw	$\bot$			$\top$		...		
sw.can_garbage	$\bot$				$\top$	...		
sw.is_sw	$\top$				$\bot$	...		

Figure 7: Example 1: A counter-example showing the alternation of election software at poll station.

i) Delivery of election software. In this example, we want to check that: *"It is never the case that poll officers receive an altered version of the election software that can be run on the machines"*. In other words, we want to verify that the procedures guarantee the integrity of the software. The property is specified in CTL as:

```
AG ! (sw.content = garbageSW &&
sw.location = pollStation &&
PollOfficersActive )
```

When checking the above property on the extended model with NuSMV, the property proves to be false — namely, it is possible for an attacker to have delivered the wrong software to the polling stations . The generated counter-example is shown in Figure 7. The key point here is that, by having access to the password to encrypt the software and by being able to substitute the software before it is put in the envelope, a modified or useless version of the software can be delivered to the polling station (See 7 for the NuSMV counterexample in which the `malSW` and `replaceSW` variables set to true highlight the execution of threat actions).

The counter-example thus highlights a possible attack and the resources that are needed to carry the attack out. Based on this information, analysts can try and suggest possible counter-measures. In the example we have shown, the attacks can be performed by having access to two assets (password and election software), when the assets are under the responsibility of the Electoral Office. The information provided by the counter-example, thus, allows analysts to focus on what assets need to be protected or on where additional controls are more effective to detect the attack.

ii) Denial of Service. In this case, we are interested in checking whether a denial of service attack could happen in a polling station. As an example, consider the following property: *"It is never the case that poll officers get the wrong PIN code"*. (Notice that the PINs are used by poll officers to activate the voting machines and voting functions.) This property is expressed in CTL formula as:

```
AG ! (PIN.can_garbage &&  PIN.location
```

	t_0	...	t_4	t_5	...	t_7	...	t_{10}
pc.pc		...	preEnv	loadEnv	...	openEnv	...	
malPiN	$\bot$	...	$\top$	$\bot$	...		...	
replacePiN	$\bot$	...	$\top$	$\bot$	...		...	
pinReady	$\bot$	...		$\top$	...		...	
can_mPPiN	$\bot$	...			...	$\top$	...	
pin.can_garbage	$\bot$	...			...		...	$\top$

Figure 8: Example 2: Denial of service attack counter-example.

```
= pollStation)
```

We give the above property on the extended model to check that the property holds. However, the tool generates the counter-example depicted in Figure 8.

The attack is similar to the previous: during the preparation of the envelopes with software and PINs, a PIN is replaced with a wrong one, as shown by figure 8. Once again, the information can be used by analysts to devise counter-measures.

The examples, although trivial, show how — by reasoning on the extended model — it is possible to explicitly represent the attacks that can be carried out, determine what assets are needed by the attackers and when, and who can carry the attacks. Similarly to what happens in model checking, we do not provide any quantitative information about the likelihood of the attacks. However, even in this simple case, we believe that the output of the attacks can provide experts the information and the requirements to enhance the current procedures, to eliminate certain attacks or, at least, to make them more difficult to implement.

5 Related Work

Various approaches for specifying, modeling, analyzing, and assessing security have been proposed in the past (see, for instance, [8, 9, 10, 11]). These approaches mainly focus on ways to build secure (software) systems by providing methodologies and techniques to develop and analyze systems, subsystems, and their execution environment. However, in *procedural rich* scenario, namely in situation in which security breaches may be carried out on outputs and assets which may be produced by ICT systems, (most of) the proposed approaches do not fit well.

To our knowledge, so far, formal procedural security analysis is quite an unexplored area. However, various work have been going on the representation and effective implementation of e-voting procedures using business process notations. In this area, the work closest in spirit to ours can be found in [20, 21], where the authors argue the need for procedural security in electronic elections and provide various examples of procedural risks occurred during trials in UK. The same authors in [22] also investigate the need for applying business process re-engineering to electoral process. Our focus, however, is on the technical machinery to automate analyses.

In [23, 24], the authors stress the importance to define roles and responsibilities within the e-voting process in order to come with a better understanding of electoral processes. Our approach complement and possibly extend these works by providing tools to support such analyses.

Last but not least, Volha et al. [25] presented an approach to reason on security properties of the *to-be* models (which are derived from *as-is* model) in order to evaluate procedural alternatives in e-voting systems. In particular, using formal approach (using **Datalog** [26] and its underlying theorem prover) they express and verify security concerns (such as, delegation of responsibility among untrusted parties, trust conflict and so on). The aim is that of understanding problematic trust/delegation relationships and eventually finding ways to adopt a solution to the detected security properties violations.

6 Conclusion and Future Work

This paper described a methodology to perform procedural security analysis based on explicit reasoning on asset flows — notably, by building a model to describe the nominal procedures under analysis and, injecting possible threat actions (by assuming that any combination of threats can be possible in all steps) into the model. We also outlined encoding strategies using NuSMV input language —that it is amenable for formal analysis allowing to reason on different properties about the procedure on the extended model such as, the "actor-play-role" principle and "reachability" of (un)desired state of an asset.

Among the advantages, a structured approach to analyze security of procedures and a general threat injection strategy that allows a high level of generality in defining the attacks. The usage of model-checkers, moreover, allows for reasoning about threat composition and to highlight, e.g. the level of coordination and resources required to carry out certain attacks.

The work presented in this paper is on-going. Among the areas of development we mention automation (e.g. algorithms to automatically perform threat injection) and better tool support. Another area of interest relates to provide guidelines that can be incorporated in the Common Criteria [27], both methodologically and tool supported way to automate the analysis.

References

[1] Adolfo Villafiorita and Giorgia Fasanelli. Transitioning to e-Voting: the ProVotE Project and the Trentino's Experience. In *EGOV-06, Krakow, Poland*, 2006.

[2] Bernard van Acker. Remote e-Voting and Coercion: a Risk-Assessment Model and Soluions. In Prosser and Krimmer [28], pages 53–62.

[3] T. Kohno, A. Stubblefield, A.D. Rubin, and D.S. Wallach. Analysis of an Electronic Voting System. In *IEEE Symposium on Security and Privacy*, 2004.

[4] J W. Bryans, B Littlewood, P Y. A. Ryan, and L Strigini. E-voting: Dependability Requirements and Design for Dependability. In *ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security*, pages 988–995, Washington, DC, USA, 2006. IEEE Computer Society.

[5] Matt Bishop and David Wagner. Risks of e-voting. *Commun. ACM*, 50(11):120–120, 2007.

[6] Ryan Gardner and Sujata Garera and Aviel D. Rubin. On the Difficulty of Validating Voting Machine Software with Software. In *EVT'07: Proceedings of the USENIX/Accurate Electronic Voting Technology on USENIX/Accurate Electronic Voting Technology Workshop*, pages 11–11, Berkeley, CA, USA, 2007. USENIX Association.

[7] P. McDaniel, M. Blaze, and G. Vigna. EVEREST: Evaluation and Validation of Election-Related Equipment, Standards and Testing. Ohio Secretary of State's EVEREST Project Report, December 2007.

[8] Igor Nai Fovino and Marcelo Masera. Through the Description of Attacks: A Multidimensional View. In Janusz Górski, editor, *SAFECOMP*, volume 4166 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 2006.

[9] David Basin, Jürgen Doser, and Torsten Lodderstedt. Model Driven Security for Process-Oriented Systems. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 100–109, New York, NY, USA, 2003. ACM.

[10] Monika Vetterling, Guido Wimmel, and Alexander Wisspeintner. A Graphical Approach to Risk Identification, Motivated by Empirical Investigations. *Lecture Notes in Computer Science*, pages 574–588, Thursday, November 23 2006.

[11] Guido Oliver Wimmel. *Model-Based Development of Security-Critical Systems*. PhD thesis, Technische Universität München, June 2005.

[12] Margaret McGaley and Joe McCarthy. Transparency and e-Voting: Democratic vs. Commercial Interests. In Alexander Prosser and Robert Krimmer, editor, *Electronic Voting in Europe*, volume 47 of *LNI*, pages 153–163. GI, 2004.

[13] Letizia Caporusso and Carlo Buzzi and Giolo Fele and Pierangelo Peri and Francesca Sartori. Transition to Electronic Voting and Citizen Participation. In Robert Krimmer, editor, *Electronic Voting*, volume 86, pages 191–200. GI, 2006.

[14] Anne-Marie Oostveen and Peter Van den Besselaar. Security as Belief User's Perceptions on the Security of E-Voting Systems. In Prosser and Krimmer [28], pages 73–82.

[15] Komminist Weldemariam, Adolfo Villafiorita, and Andrea Mattioli. Assessing Procedural Risks and Threats in e-Voting: Challenges and an Approach. In Ammar Alkassar and Melanie Volkamer, editors, *VOTE-ID*, volume 4896 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2007.

[16] Andrea Mattioli. Analysis of Processes in the Context of Electronic Election. Master's thesis, University of Trento, Italy, 2005-2006. In Italian.

[17] Aaron Ciaghi. From Laws to Models: Tools and Methodologies. Master's thesis, University of Trento, Italy, 2006-2007. In Italian.

[18] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. NuSMV 2: An Open Source Tool for Symbolic Model Checking. In *Proc. of International Conference on Computer-Aided Verification*, 2002.

[19] Marco Bozzano and Adolfo Villafiorita. The FSAP/NuSMV-SA Safety Analysis Platform. *Int. J. Softw. Tools Technol. Transf.*, 9(1):5–24, 2007.

[20] Alexandros Xenakis and Ann Macintosh. Procedural Security Analysis of Electronic Voting. In *ICEC '04: Proceedings of the 6th international conference on Electronic commerce*, pages 541–546, New York, NY, USA, 2004. ACM Press.

[21] Alexandros Xenakis and Ann Macintosh. Procedural Security and Social Acceptance in E-Voting. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 5*, page 118.1, Washington, DC, USA, 2005. IEEE Computer Society.

[22] Alexandros Xenakis and Ann Macintosh. Using Business Process Re-engineering (BPR) for the Effective Administration of Electronic Voting. *The Electronic Journal of e-Government*, 3(2), 2005.

[23] Alexandros Xenakis and Ann Macintosh. A Generic Re-engineering Methodology for the Organized Redesign of the Electoral Process to an E-electoral Process. In Robert Krimmer, editor, *Electronic Voting*, volume 86 of *LNI*, pages 119–130. GI, 2006.

[24] Costas Lambrinoudakis, Spyros Kokolakis, Maria Karyda, Vasilis Tsoumas, Dimitris Gritzalis, and Sokratis Katsikas. Electronic Voting Systems: Security Implications of the Administrative Workflow. In *DEXA '03: Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, page 467, Washington, DC, USA, 2003. IEEE Computer Society.

[25] Volha Bryl, Fabiano Dalpiaz, Roberta Ferrario, Andrea Mattioli and Adolfo Villafiorita. Evaluating Procedural Alternatives. A Case Study in e-Voting. *Proceedings of METTEG07*, 2007. An extended version has been published as a Technical Report DIT-07-005, Informatica e Telecomunicazioni, University of Trento.

[26] Thomas Eiter and Georg Gottlob and Heikki Mannila. Disjunctive Datalog. *ACM Trans. Database Syst.*, 22(3):364–418, 1997.

[27] The Common Criteria (Volume 3.1), September 2006. http://www.commoncriteriaportal.org/.

[28] Alexander Prosser and Robert Krimmer, editors. *Electronic Voting in Europe - Technology, Law, Politics and Society, Workshop of the ESF TED Programme together with GI and OCG, July, 7th-9th, 2004, in Schloß Hofen / Bregenz, Lake of Constance, Austria, Proceedings*, volume 47 of *LNI*. GI, 2004.