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

Komminist Weldemariam[1,2], Adolfo Villafiorita[1]

[1]Fondazione Bruno Kessler
Center for Scientific and Technological Research (fbk-irst)
Sommarive 18 I-38050 Povo (TN) – Italy
{sisai|adolfo}@fbk.eu

[2]DISI, University of Trento
Sommarive 14 I-38100 Povo (TN) – Italy
weldemar@disi.unitn.it

Abstract: This paper presents a methodology for procedural security analysis in order to analyze and eventually try to make elections more secure. Our approach is based on modelling the electoral procedures in the form of business process models (which we write in a strict simplified subset of UML), systematically translate the models into executable formal specifications, and analyze the specifications against security properties. We believe such an analysis to be essential to identifying the limits of the current procedures (i.e. undetected attacks) and to identify more precisely under what hypotheses we can guarantee secure elections. This paper presents the approach and demonstrates with an example taken from the e-Voting procedures enacted within the ProVotE project, current trial of the Italian legislation.

1 Introduction

The organization of elections 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. The Autonomous Province of Trento who has autonomy over local election is evaluating the switch to e-voting and, to that extent, is sponsoring the ProVotE project [VF06].

The switch to electronic elections in Italy, however, is a long and difficult process that requires extreme attention, including a thorough understanding of the limits of the risks associated to the procedures or to the combination of the procedures and systems chosen for voting. (See, e.g., [ALRL04; Mya05; FM06; MFMP07; BLRS06; LKK+03; Ale04] for a discussion of security risks associated to the usage of ICT systems and elections.)

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 checker, the effects of such threats. We believe such an analysis to be essential to, first, identifying the security boundaries— that is the conditions under which procedures can be carried out securely and, secondly, devise a set of requirements, to be applied both at the organizational level and on the (software) systems used to make systems and system processes secure. In particular, the violation of security properties could provide clues about a sequence of actions that an adversary uses to construct attacks before or during the execution of procedures.

The main contribution of this paper is twofold. On one hand we are tackling the problem at the procedural level —namely, we are trying to understand weaknesses and strengths of the procedures regulating an election, in order to analyze possible attacks and their effects on the electoral system, and, more specifically, possible attacks and threats that can be realistically carried out on the e-voting machines. On the other hand, we are interested in devising techniques and tools to analyze security threats at the organizational/procedural level, and eventually make comparison between as-is and to-be election system procedures.

This paper refines and extends the work presented in [WVM07], and it is structured as follows. In the next chapter we explain the ProVotE project scopes under which this work has been developed. In Chapter 3, we describe the context of procedural security in detail. In Chapter 4, we describe our methodology for procedural security analysis and illustrate the approach with an example in Chapter 5. Finally, in Chapter 6, conclusions drawn from this work are discussed.

2 The ProVotE Project and Motivations

ProVotE [VF06], 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 (FBK, Faculty of Sociology of the University of Trento, Fondazione Graphitech), and local industries (Informatica Trentina). 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 (see also [VF06; CBF+06] for more details).

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 (between 2005 and 2006) held during local elections. Participation to the first phase has been quite high: about 10,000 citizens took part in the experimentation[30].

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 system.

For the third phase of the project, which could 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.

With respect to scope, population, and participation, ProVotE is among the largest, if not the largest, e-voting project in Italy.

3 The Context of Procedural Security Analysis

Procedural security deals with the identification, modelling, establishment, and enforcement of security policies about the procedures that regulate the usage of a system and system processes.

The situation is depicted in Figure 1. *Our target of evaluation* is a complex organization setting in which procedures transform and elaborate *assets,* which may not necessarily be just digital assets (like, e.g., paper documents are also sensitive assets). The procedures and the organization are meant to add value to the assets and high desire to protect them from attacks, which can either come from external sources or from insiders.

[30] 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.

In particular, we distinguish the following kind of attacks:

1. *Attacks on digital assets* (item 1 and item 3 in Figure 1). These 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 kinds 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. Opportunities for attacks are determined by the organizational settings only.

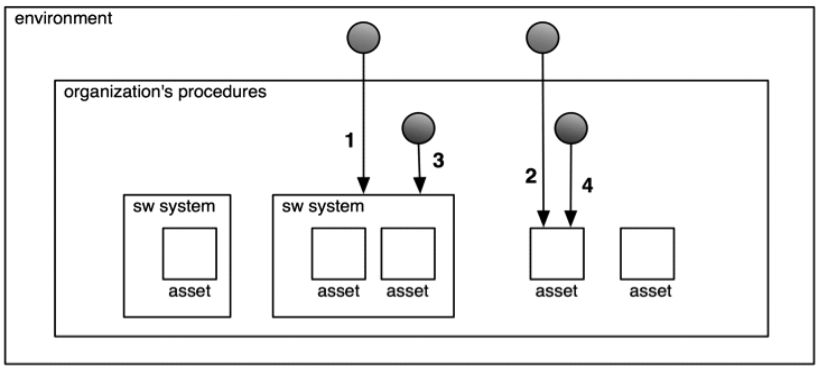

Figure 1: Procedural security Analysis.

Security assessment usually focuses on understanding items 1 and 3, namely, types and effects of attacks on (software) systems. In order to address the scenario depicted above in a systematic and tool-supported way, we *lift* the security assessment at the organizational level and we call *procedural security* analysis the usage of techniques and tools to understand 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 the assets manipulated by procedures in an unlawful way.

4 A Methodology for Procedural Security Analysis

We developed a precise methodology to perform formal procedural security analysis, based on the following steps (see also Figure 2):

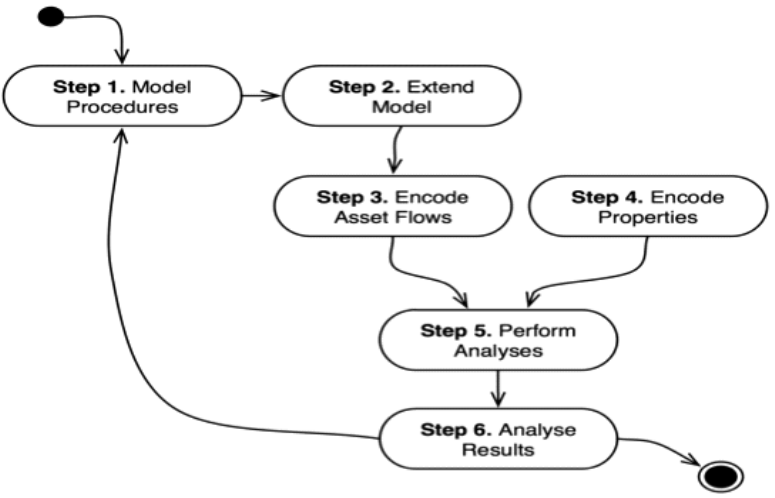

Figure 2: The process of formal procedural security.

1. ***Provide (business) models of the procedures under evaluation***. The starting point is a model that describes the process or the processes to be analyzed (Step 1 of Figure 2). In order to ease the task of translating the models into executable asset-flows, we defined and stuck to a subset of the UML activity diagrams. This allows us to describe the concepts like asset, processes, and accessory information (such as, location) in a strict and defined way. So far we managed to provide UML models of the electoral procedures in place in the Autonomous Province of Trento and in Regione Friuli Venezia Giulia. We use Visual Paradigm[31] as our reference-modelling tool. See some previous works [Man03; Mat06; Cia07] for more details about the notation, tool support, and the model themselves.

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

2. ***Inject Threat actions into the model***. We generate, from the models defined at the previous step, what we call *extended model* (Step 2 of Figure 2). The extended model is generated by "injecting" asset-threats in the nominal flow of the procedures. 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. The possible impact of threats depends upon the injection strategy that is chosen. The most general strategy is that of injecting all possible threats at all possible steps of the process (the model checker will 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.

3. ***Encode the Asset Flows***. From the extended models defined at the previous step we derive the asset flows of each asset manipulated by the procedures (Step 3 of Figure 2). Asset flows are represented in the NuSMV input language. 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 enrich the language used to express security properties. The necessity of modelling actors' roles in NuSMV depends upon the target of the security analysis. Note that from the list of activities executed to carry out, e.g., an attack, we can derive the list of actors involved, simply by looking at the UML activity diagrams.

4. ***Specify Security*** Properties to Check. The specifications of the desired (procedural) security properties, namely, the security goals that have to be satisfied, are then encoded using LTL/CTL formulas (Step 4 of Figure 2), which then (together with the model) are given as input to NuSMV.

5. ***Perform Analysis***. We finally run the model checker to perform the analyses (Step 5 of Figure 2). Counterexamples of security properties encode the sequence of actions that have to be executed in order to carry out an attack on an asset.

6. ***Analyze Results***. The last step is analyzing the obtained results (Step 6 of Figure 2). Counterexamples are used to achieve the following two goals. First, they allow to understand what are the hypotheses and conditions under which a given security goal is achieved or breached. Second, they provide information to try and modify the existing procedures, so that security breaches are taken care of. Analogously to what happens in safety analysis when analyzing, e.g., the loss of critical functions, enhancing the procedures results in reducing the probability of an attack or making the attack more complex, rather than eliminating it [Mar07].

5 A Case Study Example

Modelling Asset-flow, Step 1. Figure 3 shows a fragment of the procedure that is followed during project trials for the transfer of election results from polling stations to Electoral Office. The diagram abstracts away those details that are irrelevant for the sake of presentation, e.g. details related to the alternative modelling choices for carrying out the data transfer process are omitted. We also hide some actors' responsibilities by collapsing, e.g. Secretary, Scrutinizers, them into a single actor. See in [Vol07] for detail strategies of data transfer process and how the alternative choices are modelled.

The diagram illustrates (see Figure 3), after the election, the Section President (one of the Poll Officers) deactivates the voting machines, extracts (from the voting machine) printed votes, the USB key with the results, and other artifacts, and prepares a package containing votes and various reports, to be delivered to the Electoral Office. Electronic data are transmitted through a VPN and the USB key with the electronic results delivered to the Electoral office via a "messenger" (e.g. a police officer).

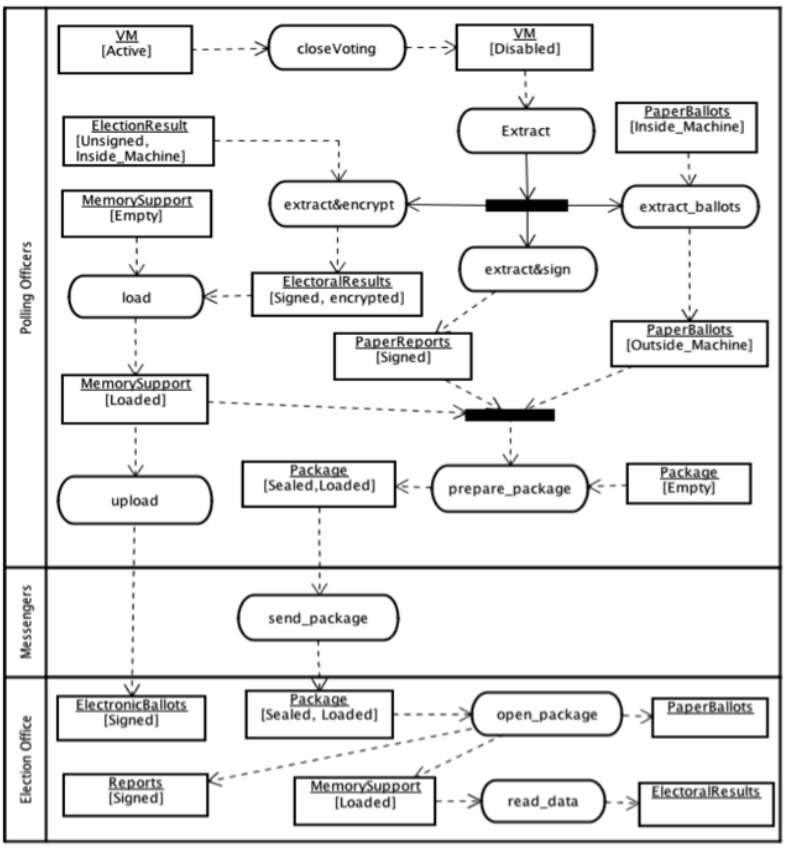

Figure 3: An example of asset flow.

Threat Injection, Step 2. The next step is *injecting*, that is, extending the model with threat actions and generate the *extended model*. Figure 4 shows some examples of threat-actions injected into the nominal model of Figure 3. In the extended model, threat actions are marked with the stereotype "threat-action". Impact of the attacks depend upon the asset they target and the position, in the procedure, where the attack take place.

For instance, replacing the results of a polling station in a USB key has no effect after the result have been generated. (On the other hand it may change the results of the election if performed before the results have been computed.)

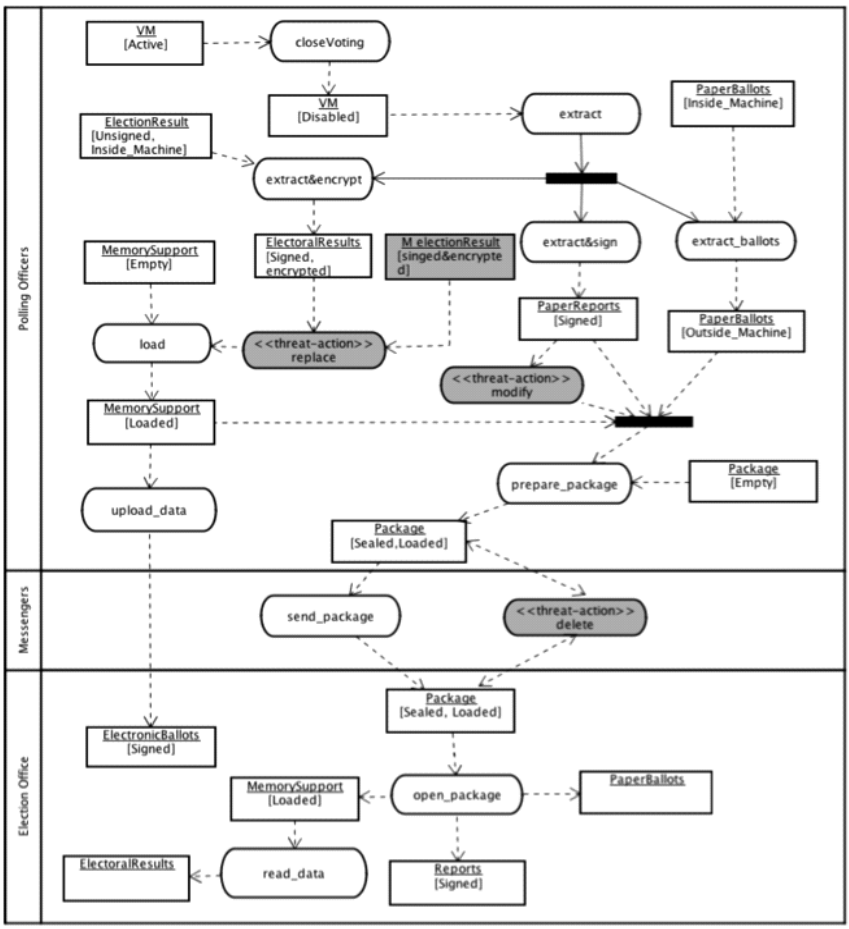

Figure 4: An example of extended model.

Asset-Flow encoding, Step 3. Below we show a snippet of the code that defines the asset type *electionResult* and some of its feature variables, named *state* (the states in which the *electionResult* can be) and the content (the qualitative value of the *electionResult* can be).

```
MODULE electionResult ( ... )
VAR
state     : {plain,unsigned,signed,signed_&_encrypted};
content : {null,data,signed_&_encrypted_data,garbage};
```

The evolution of assets' properties is 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). Below, for instance, we show a piece of NuSMV code that illustrated how the content variable of *electionResult* asset changes:

```
init(content) := null;
next(content) := case
pc.pc = closeVoting && next(pc.pc) = extract_&_encrypt : data;
content = data && pc.pc = extract_&_encrypt && state = signed: signed_&_encrypted_data;
[...]
```

Threat injection (model extension) corresponds to augmenting the state machine of the asset flow with new transitions (e.g., adding a transition that leads to a *garbage* state of *content*) corresponding to the execution of threat actions. The triggering of a threat action is "monitored" through boolean variables that are set to true when the action takes place, as illustrate by the following pieces of code:

```
next(can_malElectionRes) := case
  (malElectionRes && pc.mpc = replaceElectionRes &&
  next(electionResult.content) = malEnSignedData) || [...]  :1;
  1: can_malElectionRes;
  esac;
```

Note that in the codes above we have left some detail specification (such as location) for the matter of presentation purposes. Analogously, the remaining asset flows and model extension encodings can be encoded.

Specify Security Properties and Perform Analysis, Step 4 & 5. We use temporal logic formulas to represent the properties of interest and model check them using the NuSMV tool. In particular, security properties are specified using LTL/CTL logic language. LTL is used to reason on the computational path scenarios of an asset (e.g., what can happen as asset travels along different locations), while CTL 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).

Among the property classes we are interested in is that of verifying a property about "*Safe transfer of election result.*" A desirable property, for instance, that we want to specify and analyze can be described in plain text as: "*It is never the case that election officials receive modified election data before computing the final result.*" This property is expressed in CTL formula as:

```
AG ! (ElectionResult.can_garbage &&  ElectionResult.location = electoralOffice)
```

We give the above property to NuSMV tool to check that the property holds. However, the tool generates a counter-example showing the violation of the given property. Upon analyzing the generated counter-example, the election result is replaced (i.e., a replace attack is in place) following the introduction of a wrong election data into the asset flow, which, in turn, causes wrong delivery of election result to the electoral office. Among the possible scenarios that we analyzed, at some time a malicious election data is introduced while poll officer is preparing the data to transfer to electoral office. At the same time, an attacker implements replace attack before loading the memory support.

6 Related Work

Various approaches (for specifying, modelling, analyzing, and assessing security) have been proposed in the past and proven useful for zeroing the security lacks of the analyzed systems (see, for instance, [FM06; BDL+03; VWW06; Wim05]).

To our knowledge, however, formal procedural security analysis is quite an un-explored area. The work closest in spirit to ours can be found in [XM04, XM05], where the authors argue the need for procedural security in electronic elections and provide various examples of procedural risks occurred during trials in the UK; in [LKK+03, XM06] where the authors highlight the importance of defining roles and responsibilities in e-voting and in [Ale05] where the need for applying business process re-engineering to the electoral process is emphasized. Our focus, however, is on the technical machinery to automate analyses.

Volha et al. [Vol07] presents 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 using Tropos.

Finally, Alexander et al. in [PKKU04] also highlighted a comprehensive way of overviewing attacks against sensible assets in all stages of e-voting.

7 Conclusion

In this paper we presented a methodology to perform procedural security analysis based on explicit reasoning on asset-flows — notably, by building a model to describe the nominal procedures implementation, enriching this model with possible threat actions, and encoding the extended model to suit for model checking techniques which, in turn, allows to reason on different aspects of the procedures such as, the "actor-play-role" principle and some reachability analysis for some undesired state of an asset. Among the advantages of our approach, the possibility of getting a better comprehension of the effect and impact of combined attacks to the assets of an election.

The model checker runs that were made on the current version of the specification have not revealed much interesting results though seemed useful; therefore, much work needs to be done in order to see if the model can be fully verified or if any interesting results can be uncovered. Moreover, we need to consolidate our approach and provide guidelines that can be incorporated in the Common Criteria [cc07], both methodologically and in a tool supported way to automate the analysis.

References

[Ale04] Alexandros Xenakis and Ann Macintosh. Levels of Difficulty in Introducing e-Voting. Electronic Government, 3183/2004, November 05 2004. LNCS, Springer.

[Ale05] 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.

[ALRL04] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing, 01(1):11–33, 2004.

[BCP+02] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. NuSMV 2: An Open Source Tool for Symbolic Model Checking. In Proceeding of International Conference on Computer-Aided Verification, 2002.

[BDL+03] 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.

[BLRS06] J W. Bryans, B Littlewood, P Y. A. Ryan, and L Strigini. E-Voting: Dependability Requirements and Design for Dependability. ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security (ARES'06), 0:988–995, 2006.

[CBF+06] Letizia Caporusso, Carlo Buzzi, Giolo Fele, Pierangelo Peri, and Francesca Sartori. Transition to Electronic Voting and Citizen Participation. In Robert Krimmer, editor, Electronic Voting, volume 86 of LNI, pages 191–200. GI, 2006.

[cc07] Common Criteria. 2007. http://www.commoncriteriaportal.org/.

[Cia07] Aaron Ciaghi. From Laws to Models: Tools and Methodologies. Master's thesis, University of Trento, Italy, 2006-2007. In Italian.

[FM06] Igor Nai Fovino and Marcelo Masera. Through the Description of Attacks: A Multidimensional View. In Computer Safety, Reliability, and Security, 25th International Conference, SAFECOMP 2006, Gdansk, Poland, September 27-29, 2006, Proceedings, pages 15–28, 2006.

[LKK+03] 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.

[Man03] Andrea Mattioli. From Processes to Information Systems: Tools for Sharing Models. Master's thesis, University of Trento, Italy, 2002-2003. (In Italian)

[Mar07] Marco Bozzano and Adolfo Villafiorita. The FSAP/NuSMV-SASafetyAnalysisPlatform. Int. J. Software Tools Technology Transfer, 9(1):5–24, 2007.

[Mat06] Andrea Mattioli. Analysis of Processes in the Context of Electronic Election. Master's thesis, University of Trento, Italy, 2005-2006. (In Italian)

[MFMP07] Daniel Mellado, Eduardo Fernández-Medina, and Mario Piattini. A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems. Comput. Stand. Interfaces, 29(2):244–253, 2007.

[Mya05] Myagmar, S. and Lee, A. and Yurcik, W. Threat Modelling as a Basis for Security Requirements. In StorageSS '05: Proceedings of the 2005 ACM workshop on Storage security and survivability, pages 94–102,, New York, NY, USA, 2005. ACM Press.

[PKKU04] Alexander Prosser, Robert Kofler, Robert Krimmer, and Martin Karl Unger. Security Assets in E-Voting. In Electronic Voting in Europe, pages 171–180, 2004.

[VF06] Adolfo Villafiorita and Giorgia Fasanelli. Transitioning to e-Voting: the ProVotE Project and the Trentino's Experience. In EGOV-06, Krakow, Poland, 2006.

[Vol07] Volha Bryl, Fabiano Dalpiaz, Roberta Ferrario, Andrea Mattioli and Adolfo Villafiorita. Evaluating Procedural Alternatives. A Case Study in e-Voting. Proceedings of MET-TEG07, 2007. An extended version has been published as a Technical Report DIT-07- 005, Informatica e Telecomunicazioni, University of Trento.

[VWW06] 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.

[Wim05] Guido Oliver Wimmel. Model-Based Development of Security-Critical Systems. PhD thesis, German umlauts Institut f¨r Informatik der Technischen Universität München, February 2005.

[WVM07] 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.

[XM04] 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.

[XM05] 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, Wash- ington, DC, USA, 2005. IEEE Computer Society.

[XM06] Alexandros Xenakis and Ann Macintosh. A Generic Re-engineering Methodology for the Organized Redesign of the Electoral Process to an E-electoral Process. In Electronic Voting, pages 119–130, 2006.