An integrated application of security testing methodologies to e-voting systems

Marco Ramilli and Marco Prandini

University of Bologna, Italy
{marco.ramilli,marco.prandini}@unibo.it

Abstract. Various technical bodies have devised methodologies to guide testers to the selection, design, and implementation of the most appropriate security testing procedures for various contexts. Their general applicability is obviously regarded as a necessary and positive feature, but its consequence is the need for a complex adaptation phase to the specific systems under test. In this work, we aim to devise a simplified, yet effective methodology tailored to suit the peculiar needs related to the security testing of e-voting systems. We pursue our goal by selecting, for each peculiar aspect of these systems, the best-fitting procedures found in the most widely adopted security testing methodologies, at the same time taking into account the specific constraints stemming from the e-voting context to prune the excess of generality that comes with them.

Key words:eVoting, security, testing methodologies

1 Introduction

Testing is important to ascertain the adherence of an implemented system to its specification, but even more to prove that it exhibits sensible reactions to unexpected stimuli. Even the best design process cannot capture the latter property, since no explicit requisite can represent it; thus, testing contributes in an unique way to the development cycle of secure systems [1–3], notwithstanding the impossibility of guaranteeing the absence of any problem through it [4]. The scientific and technical communities have made various attempts at defining detailed procedures for security testing. Clearly, the goal pursued in these efforts was to devise generally-applicable guidelines, while at the same time providing as much detail as possible regarding the proper way of performing each step. Two limitations affect the proposals found in the literature. First, their application could result quite cumbersome, requiring a non-negligible effort in the preliminary phase of mapping the suggested procedures to the specificities of the system to be tested. Second, the "perfect" proposal does not exist, each one exhibiting areas of excellence and more neglected sections.

We aim at a twofold result for the mitigation of the aforementioned problems. The paper is structured as follows. In sections 2 and 3 we briefly outline the most widely adopted testing methodologies. We proceed (section 4) to synthetically review the commonly adopted e-voting architectures and to summarize

their common characteristics. By combining the elements of this preliminary exploration, we choose from the different methodologies the testing procedures that better suit the specific needs related to testing the components of e-voting systems. Taking into account the constraints that define the class of e-voting systems, as opposed to generic ones, we simplify those procedures by safely removing the unneeded details and more precisely driving the testing process.

2 Related work

We deem useful to recall a few concepts related to the different approaches to security testing, to e-voting, and to the relation between the two worlds that emerges from the most significant experiences to date.

2.1 Security testing approaches

There are significant differences between the many papers, from the academic as well as the technical world, that deal with the subject of security testing. A possible classification organizes the various proposals into three broad categories:

Toolkits implement in a convenient package a set of testing techniques, usually aimed at discovering specific classes of security problems. Toolkits represent the operating side of security testing. They are valuable companions to guidelines and methodologies, which in turn provide the strategies to effectively use them. The Open Vulnerability Assessment System [5] and the BackTrack Live CD [6] are significant examples among countless others.

Guidelines organize the process of security testing, by collecting sets of best practices and comprehensively listing items to be tested; they often distill the experiences gathered on the field by the technical community, but usually lack the level of detail that allows to design a precise test plan. Some examples of well-known guidelines come from NIST, namely: the Common Criteria for Information Technology Security Testing [7] and the Technical Guide to Information Security Testing and Assessment [8]. The Open-ended vulnerability testing (OEVT) [9] is being devised starting from the experiences gathered by the Electoral Assistance Commission in the U.S.A on the Voluntary Voting System Guidelines (VVSG) [10].

Methodologies represent the most structured approach to security testing. To different extents, every methodology defines: (a) an abstract model for the system, (b) an abstract model for the process of finding its vulnerabilities, and (c) a procedure for realizing a concrete test plan from the models, given the details of the system under test. A detailed discussion of the most widely adopted methodologies is illustrated in section 3.

2.2 E-voting systems

The need for technological aids to make the voting process more efficient and accurate predates the availability of sophisticated computer architectures, but

usually the adopted solutions (for example, punch-card devices and optical scan machines) perform little more than substituting a supposedly more robust media for the traditional pencil-and-paper. More recently, comprehensive systems have been designed and implemented that exploit computers and networks to take care of every step of the voting process. The foreseen advantages have greatly increased, but the concerns raised by the trustworthiness and security aspects of such complex (and opaque) architectures have grown equally strong. Examples of commonly adopted approaches include:

Direct Recording Electronic Voting System (DRE). The voter chooses by simply touching the name of the candidate directly on screen and the machine casts the vote on its own storage device. At the end of the election, the machine produces an exhaustive report to be sent to the precinct for counting. The systems by leading vendors (Diebold, Election Systems and Software, Sequoia among the others), widely adopted for instance in the U.S.A., are mainly of this kind. However, they have harshly been criticized for the complete lack of independent correctness-checking capabilities, that leaves open the possibility of undetectable mistakes of malicious or accidental nature.

Voter Verified Audit Trail (VVAT) Electronic Voting Machine solve this problem by generating a proof which can be audited by the voter to ascertain the correct recording of her will. It can be a paper ballot which can be reviewed by the voter before confirming her intention to cast it, and then collected in a secure storage should a recount be needed, or a mathematical proof that pushes the concept even further, by allowing the voters to check whether their vote was accurately recorded by the electronic system (end-to-end verification), not only in the paper trail [11, 12].

2.3 E-voting Security Threats

Although security testing is an incomplete test, meaning that it does not ensure the absence of flaws, it is the *only* process able to prove threats. In sensitive systems like e-voting, the presence of threats might interfere with the correct election outcome compromising the democracy of the hosting country. Examples of the most important areas where security threats might be present are :

1. Secrecy. If the system does not assure secrecy, the system is at least vulnerable to covert channels attacks, where an attacker may buy or sell votes.
2. Integrity. If the system does not assure integrity, an attacker could compromise the election by replacing or modifying the integrity of the ballots or directly the integrity of the final counts.
3. Availability. If the system does not assure availability, the system can not assure the universal suffrage, becoming vulnerable at least to external quorum attacks, in which the attacker can modify the total number of voters denying the minimum voters requirements.
4. Authentication. If the system does not assure authentication controls, it is at least vulnerable to multiple vote attacks, where an attacker could vote multiple times for the preferred candidate.

Depending on the system implementation we may find different entry points where the security threats may appear. For example the integrity of the system might be threatened by malwares, or directly by the vendor introducing incorrect behaviors or backdoors on the voting platform; the authentication control might be threatened by wrong input validation, brute force attacks or buggy sessions. Since the range of the entry points is so large and so strongly platform dependent, the paper does not describe the details of each of them, but synthesizes the general features useful to devise an e-voting system testing methodology.

2.4 E-voting systems testing experiences

Oddly enough, to our knowledge, there is no documented application of the most complete testing methodologies to e-voting systems. Certification for official use, where it is mandatory, commonly follows guidelines like the VVSG, that are quite country- and technology-specific. *A posteriori* security reviews skillfully exploit various toolkits and attack techniques, not adopting structured approaches (but producing interesting results nonetheless). Notable examples of the latter category were the seminal Security Analysis of the Diebold AccuVote-TS Voting Machine [13] performed in 2006, the California Top-to-Bottom Review [16], performed by various Californian universities [14, 15] on all the voting systems used in 2007 for state and local elections, and the similar Evaluation & Validation of Election-Related Equipment, Standards & Testing (EVEREST) program undertaken in Ohio in the same year [17].

3 Existing security testing methodologies

In this section, we give a glimpse of the main security testing methodologies that we exploited to build a tailor-made testing process suited for e-voting systems.

3.1 ISSAF

The Information Systems Security Assessment Framework (ISSAF) [18] is a well-established penetration testing methodology, developed by OISS.org. It is designed to evaluate the security of networks, systems and application controls. The methodology outlines three well-defined action areas, and details the nine steps composing the main one, as follows:

- Planning and Preparation. The first phase encompasses the steps needed to set the testing environment up, such as: planning and preparing test tools, contracts and legal protection, definition of the engagement team, deadlines, requirements and structure of the final reports.
- Assessment. This phase is the core of the methodology, where the real penetration tests are carried out. The assessment phase is articulated in nine activities: (1) Information Gathering; (2) Network Mapping; (3) Vulnerability Identification; (4) Penetration; (5) Gaining Access & Privilege Escalation; (6) Enumerating Further; (7) Compromise Remote Users Sites; (8) Maintaining Access; (9) Covering Tracks.

– Reporting, Clean-up and Destroy Artifacts. During this phase, at the very end of the active parts of the methodology, testers have to write a complete report and to destroy artifacts built during the Assessment phase.

ISSAF has a clear and very intuitive structure, which guides the tester through the complicated assessment steps.

3.2 OSSTMM

The Open Source Security Testing Methodology Manual (OSSTMM) [19] is the de-facto standard for security testers. It describes a complete testing methodology, offering fairly good tools to report the result set. In the OSSTMM ontology, the complex made by the target and the logical infrastructure to access it is named *scope*. The scope of application of OSSTMM encompasses any interaction, with any asset within the whole operating security environment, including the physical components of security measures as well. The methodology mandates that all the threats within the scope must be considered possible, even if not probable. On the access paths side, the scope is organized in three channels: COMSEC (communications security), PHYSSEC (physical security), and SPECSEC (spectrum security). Channels are the means of interacting with assets, where an asset is defined as anything of value to the owner. Fig. 1 represents the scope extension. The three main channels are split into 5 sub-channels:

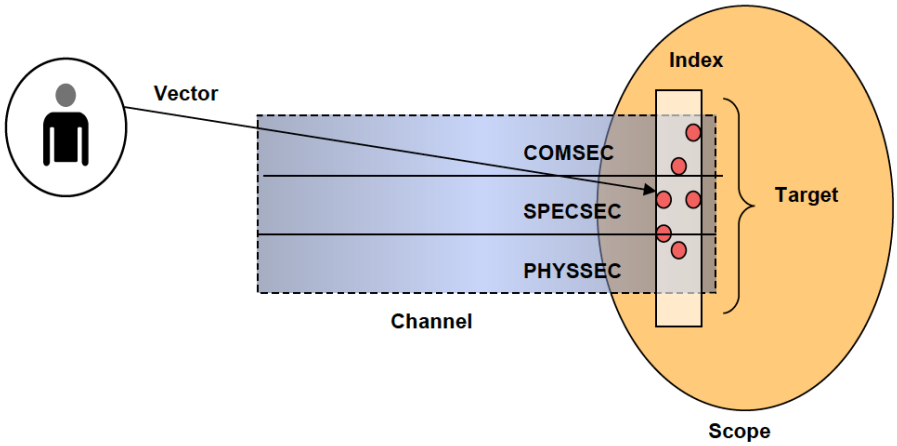

Fig. 1. OSSTMM: The Scope Process.

– Human. It comprises all the human elements of communications
– Physical. It comprises the tangible elements of security where interaction requires physical effort or an energy transmitter to manipulate.

- Wireless Communication. It comprises all the electronic communications, signals and emanations which take place over the known EM spectrum.
- Data Networks. It comprises all the electronic systems and data networks where interactions take place over dedicated cables and wired network lines.
- Telecommunication. It comprises all the telecommunication networks, digital or analog, where the interaction takes place over shared network infrastructures, such as telephone lines.

OSSTMM describes 17 modules to analyze each of the sub-channels. Consequently, the tester has to perform 17*5 = 85 analyses before being able to write the final report.

3.3 Black Hat

Most attackers follow a sort-of-coded procedure to exploit systems, made of four steps, as described in the following list:

- Bugs Information Discovery. In this step the attacker, using automatic and manual analysis, gathers information about system bugs.
- Exploration. In this step the attacker filters the informations obtained in the previous step, obtaining a list of vulnerabilities (i.e. exploitable bugs).
- Assessment. The attacker figures out the most profitable vulnerability.
- Exploitation. The attacker, using both known and improvised techniques, begins the exploitation.

While the apparent order of this procedure has led many to call it "the Black Hat Methodology" (BHM), it is not formally defined anywhere, nor general enough to be used for penetration testing. The main difference between attacking a system and performing penetration testing is the final goal: to attack a system the attacker needs only one vulnerability, to protect the system the tester needs to find all the vulnerabilities. The non-cyclic control flow present in the methodology does not help the tester to find each vulnerability, but only the first one.

3.4 GNST

The Guideline on Network Security Testing (GNST) [20] issued by NIST, notwithstanding the name, is the first methodology to introduce a formal process for reporting and to take advantage of inducted hypotheses. It follows four steps:

- Planning. System analysis finds out the most interesting test targets.
- Discovery. The tester searches the system, looking for vulnerabilities.
- Attack. The tester verifies whether the found vulnerabilities can be exploited.
- Reporting. In the last step, every result is reported.

Each step has an input vector representing known facts and an output vector representing the complete set of results deriving from the performed actions. GNST introduces an attempt at considering inducted hypotheses, where the output vector from a step can become part of the input vector of another one.

4 Applying methodologies to e-voting systems

There are many different kind of tests to be performed on voting systems, for which the authors believe that a specific methodology is needed, such as: usability testing, performance testing, and proof of correctness. With an overall perspective, the tester needs to verify the good behavior checking each election requirement. Testing the election requirements mens checking:

R.1) Voter Validation. The voter should reach the state where he is authenticated, registered and he has not yet voted.
R.2) Ballot Validation. The voter must use the right ballot, and the ballot captures the intent of the voter.
R.3) Voter Privacy. The voter cannot be associated with the ballot, not even by the voter herself.
R.4) Integrity of Election. Ballots cannot change during the election time and the casted votes are accurately tallied.
R.5) Voting Availability. Voters must be able to vote, all enabling materials must be available.
R.6) Voting Reliability. Every voting mechanisms must work.
R.7) Election Transparency. It must be possible to audit the election process.
R.8) Election Manageability. The voting process must be usable by those involved.
R.9) System State Requirements. The systems must meet the State certification requirements.
R.10) State Certifications. The voting system must have the certification of the State where the election takes place (whether it considers the afore-listed requirements or a different set).

Focusing on the security aspects of e-voting systems testing, we may consider as the common and implicit "*testing goal*" of the process the overall security of the system. Considering that in security the composability property does not hold (security(a) $\cup$ security(b) != security(A $\cup$ B)), except in unrealistically simple situations and after an unusually complex design process, the tester must verify every component and the whole system in two separate views. This means that tester has to test at least a fixed object called *Voting System* and many different objects called *Voting Objects*.

4.1 Testing Voting System and Voting Objects

The voting objects vary according to the analyzed system, but for the sake of clarity some examples include: touch screen monitors, printers, network cables and routers, power supplies, software and so forth. For each defined Voting Object the tester needs to verify that it is not possible to:

– Compromise the Hardware, i.e. insert, remove, substitute or damage physical devices. An example of denial of service attack performed through the hardware occurs when an attacker cuts the edges of a resistive touchscreen monitor (RTM). The attack analysis shows that the vulnerability resides in

the technology that place the touch sensors on the surface of the screen, and suggests to adopt as a countermeasure the substitution of RTM with capacitive touchscreen monitors, which have glass-hidden sensors.
- Compromise the Firmware, i.e. alter drivers, hardware BIOS or embedded code. An example of election hijacking performed through firmware alteration occurs when the attacker modifies a router, choosing it because it is a rarely tested COTS component, substituting its firmware with a custom one which allows to dump or to manage the network communications between machines and the ballot box, thus greatly increasing the chances of compromising the election system.
- Compromise the Software, i.e. insert new code, modify the existing code, delete existing code or force an unexpected behavior. For example, an attack vector of this kind on the Unix platform could be an unsecured boot process allowing an attacker to find a privileged login through single-user-mode, or an unsecured terminal where by shutting down the graphic user interface the attacker can operate on the local file system.

Assessing the absence of the afore-listed attack opportunities does not mean that the analyzed system can be considered safe. The best way for a tester to identify all the possible flaws is to consider the most favorable situation for the attacker (the worst situation for the system), assuming a White Open Box point of view, where everyone knows how the system works (through documentation), how the system has been written (through source code) and where the tester can simulate both internal and external attacks. We define the posture of tester as "*Voting System Tester Point of View*", which is unique for all the systems. Flaws hypotheses and induction flaws hypotheses may be applied in the same way as most of the methodologies show. Properly documenting the evidence regarding what the tester has found, and reporting every relevant action performed during the test is a common provision of most of the methodologies. Summing up, the new methodology should have three new basic assumptions as follow:

A.1) Testing Goals = the entire security of electronic voting system
A.2) Testing Objects = Voting System + Voting Objects
A.3) Tester Point Of View = Voting System Tester Point of View = Internal/External Open White Box

Adding assumptions means decreasing the procedure's complexity because the final methodology has three less steps to follow. Fig. 2 shows the transition from the discussed methodologies assumptions to the new ones. On the left of Fig. 2 "*testing goals*" are defined. ISSAF defines the testing goals in the "Planning and Preparation" section, OSSTMM in the "Scope" section and GNST in the "Planning" section. The meaning of the arrows between left boxes and the central one is that each "*testing goal*" is an instance of "Security of Voting System" as previously discussed. On the right of Fig. 2 "*Testing Objects*" are defined. ISSAF define the testing objects in the "Planning and Preparation" section, GNST in the "Planning" section, while OSTMM classifies the testing objects in the three known channels. The meaning of the arrows between the right boxes and

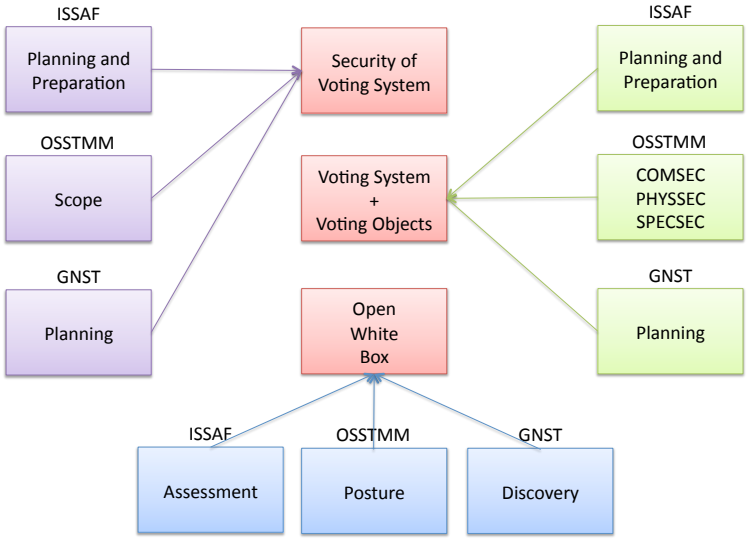

Fig. 2. Transition from old to new assumptions

the central one is that each "*Testing Objects*" should be collapsed into "Voting System + Voting Objects". Finally on the bottom of Fig. 2 "*Voting System Tester Point of View*" is represented. ISSAF defines the Voting System Tester Point of View into the "Assessment" section, OSSTMM in the "Posture" section and GNST in the Discovery section. Again, the meaning of the arrows between the bottom boxes and the center one is that each "*Voting System Tester Point of View*" should be fixed to "Open White Box" to ensure a safe, worst-case-scenario analysis.

5 Tailoring the methodologies to the e-voting context

In this section we finally discuss how to choose the most appropriate procedures from the illustrated methodologies, adapting and simplifying them to fit the scenario of e-voting systems testing. The description is necessarily kept at a rather high level of abstraction, because the amount of details involved in the accurate description of each set of procedures could never fit a conference paper.

5.1 ISSAF Adaptation

ISSAF can be exploited as follows, taking advantage of the three new assumptions introduced in section 4. Referring to the Fig.2 the main ISSAF "Planning and Preparation" steps are:

- Identification of contact individuals from both sides.
- Opening meeting to confirm the scope, approach and methodology.
- Agreement on specific test cases and escalation paths.

By fixing the assumptions A.1 and A.2, the tester does not really need to perform the first two steps, which are time and money consuming and often require organizational skills that do not belong to the tester. In the presented scenario there is no way to discuss the scope of the security test; it cannot be other than "the entire security of electronic voting system". Similarly, there is only one set of testing objects that must be tested, as shown in point A.2, thus freeing the tester from the need to define agreements of specific tests cases and escalation paths. Fixing assumption A.3 simplifies the process shown in section 3.1, allowing to avoid the following 3 steps out of the proposed 9:

- Information Gathering.
- Gaining The First Access.
- Privilege escalation.

Notice that the tester does not need to verify the absence of privilege escalation or of remote/local access to the machine, not because these are irrelevant; on the contrary, the starting assumption means that the tester directly operates on the worst-case scenario assuming the attacker already owns this information.

5.2 OSSTMM Adaptation

OSSTMM provides a comprehensive concept of scope, allowing a vast variety of scenarios. For its application to the e-voting domain, it is possible to reduce the space of possible testing procedures by taking into account the assumption A.1 and A.2 as described in section 4.1. These allow to prune the the Scope Definition process, composed by the regulatory phase (cfr. page 25, sec. A.1 and A.2, OSSTMM light edition) and definition phase (page 26, sec. B.4 to B.7, ibid.). Another simplified step regards the information phase (cfr. pages 26-27, sec. C.8 to C.13, ibid.) where the tester should acquire as much information as possible about the system. According to the section 4.1 we reduce the information phase into the assumption A.3, freeing the tester from to the heaviest part of the information gathering task.

5.3 GNST Adaptation

GNST does not provide a detailed set of actions to define what it calls "Planning". It suggests to define rules, to acquire management approvals, to find financing and finally to set up the testing goals and testing objects. Although no strong guidelines are presented, each of the aforementioned steps is superfluous in the e-voting domain, where testing is clearly mandated and financed and testing objects have been previously clarified: the entire GNST Planning phase can be substantially collapsed by applying the constraints deriving from A.1 and part of A.2. GNST's discovery phase has been defined as follow:

- Network Scanning.
- Domain Name System (DNS) interrogation.
- InterNIC (whois) queries.
- Search of the target organization's web server(s) for information.
- Search of the organization's Directory server(s)for information.
- Packet capture (generally only during internal tests).
- NetBIOS enumeration (generally only during internal tests).
- Network Information System (usually only during internal tests).
- Banner grabbing.

By assuming a tester point of view according to A.3, the whole "discovery phase" can be taken as an assumption, allowing insider and external security tests. Following the general methodology, if the tester cannot find a way to remote access the system, he skips all the insider attacks. Assuming A.3, even in this case the tester will perform the tests related to threats originating from a potential insider attacker.

6 Conclusion

Security testing is a fundamental phase in the life cycle of almost any system. Sensitive systems like those used for e-voting undergo particularly severe testing to attain certification of their security properties before usage into a real election. This exacting process should be based on one of the state-of-the-art methodologies described in section 3 of this paper. These exist to manage the planning and execution of testing procedures, taking into account the complex interrelations between the different parts and the huge amount of detail involved, on any kind of system. However, before being usable on peculiar systems, any methodology has to be adapted to the specific context. This paper described the common-denominator aspects, constraints and problems that characterize the whole class of e-voting systems, across their different instantiations (DREs, VVPATs, etc.). With this knowledge, it was possible to identify the procedures of the different methodologies that are most fit to this specific domain, and to provide some guidelines to instantiate them in the most effective way, by removing as many unnecessary steps as possible. A key step in this direction was fixing some unequivocal assumptions, as described in section 4.1. Assumptions work by explicitly stating the context elements that the tester can assume to hold without the need for verifying them, thus removing some degrees of freedom that otherwise leave manifold testing paths open, and eventually allowing to reduce the complexity of the testing phase. The (inital) result should be of help to prospective testers, strongly kick-starting the unavoidable phase of adaptation to the exact system they are dealing with. The ongoing work regards the refinements of practical details and the preparation of a case study to demonstrate the effectiveness of the proposed work on a real system.

References

1. Arkin, B., Stender, S., McGraw, G.: Software penetration testing. Security & Privacy, IEEE **3**(1) (Jan.-Feb. 2005) 84–87
2. Bonver, E., Cohen, M.: Developing and retaining a security testing mindset. Security & Privacy, IEEE **6**(5) (Sept.-Oct. 2008) 82–85
3. Potter, B., McGraw, G.: Software security testing. Security & Privacy, IEEE **2**(5) (Sept.-Oct. 2004) 81–85
4. Thompson, H.: Why security testing is hard. Security & Privacy, IEEE **1**(4) (July-Aug. 2003) 83–86
5. Brown, T., Anderson, W., et al.: Open vulnerability assessment system (dec 2009)
6. Moser, M., Aharoni, M., Muench, M.J., et al.: Backtrack (jun 2009)
7. Horlick, J.: HB 150-20 Information Technology Security Testing: Common Criteria. National Institute of Standards and Technology (October 2005)
8. Scarfone, K., Cody, A., Souppaya, M., Orebaugh, A.: SP 800-115 Technical Guide to Information Security Testing and Assessment. National Institute of Standards and Technology (September 2008)
9. Technical Guidelines Development Committee, ed.: 5.4. In: Voluntary Voting System Guidelines Recommendations to the Election Assistance Commission. U.S. Election Assistance Commission (August 2007)
10. U.S. Election Assistance Commission, ed.: Voluntary Voting System Guidelines. U.S. Election Assistance Commission (2005)
11. Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A., Vora, P.: Scantegrity: End-to-end voter-verifiable optical- scan voting. Security Privacy, IEEE **6**(3) (may-june 2008) 40–46
12. Adida, B.: Helios: web-based open-audit voting. In: SS'08: Proceedings of the 17th conference on Security symposium, Berkeley, CA, USA, USENIX Association (2008) 335–348
13. Feldman, A.J., Halderman, J.A., Felten, E.W.: Security analysis of the diebold accuvote-ts voting machine. In: EVT'07: Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology, Berkeley, CA, USA, USENIX Association (2007) 2–2
14. Bishop, M., Wagner, D.: Risks of e-voting. Commun. ACM **50**(11) (2007) 120–120
15. Balzarotti, D., Banks, G., Cova, M., Felmetsger, V., Kemmerer, R., Robertson, W., Valeur, F., Vigna, G.: Are your votes really counted?: testing the security of real-world electronic voting systems. In: ISSTA '08: Proceedings of the 2008 international symposium on Software testing and analysis, New York, NY, USA, ACM (2008) 237–248
16. Wagner, D.: Report of the california voting system review (USENIX Security Symposium 2007) http://www.usenix.org/events/sec07/tech/.
17. Ohio secretary of state (pub): Evaluation & validation of election-related equipment, standards & testing - http://www.sos.state.oh.us /SOS/elections/voterInformation/equipment/VotingSystemReviewFindings.aspx
18. Open Information Systems Security Group: Information systems security assessment framework (2006)
19. Institute for Security and Open Methodologies: Open source security testing methodology manual (2009)
20. Wack, J., Tracy, M., Souppaya, M.: SP 800-42 Guideline on Network Security Testing. National Institute of Standards and Technology (October 2003)