Vote Selling, Voter Anonymity, and Forensic Logging of Electronic Voting Machines

Sean Peisert and Matt Bishop
Department of Computer Science
University of California, Davis
{peisert,bishop}@cs.ucdavis.edu

Alec Yasinsac
School of Computer and Information Sciences
University of South Alabama
yasinsac@usouthal.edu

Abstract

Much recent work has focused on the process of auditing the results of elections. Little work has focused on auditing the e-voting systems currently in use. The facilities for doing the former include the voter-verified paper audit trail; unfortunately, that VVPAT is not particularly helpful in tracking down the source of errors within e-voting systems. This paper discusses the need for a detailed forensic audit trail (FAT) to enable auditors to analyze the actions of e-voting systems, in order to demonstrate either the absence of problems or to find the causes of problems. We also discuss methods to prevent the use of the FAT as a covert channel for violating the necessary properties of secrecy of the ballot, so voters cannot sell their votes, and anonymity of the ballot, so a third party cannot associate a particular ballot with the voter who cast it.

1 Introduction

Highly public examples of electronic voting machine meltdowns in the United States continue to occur, and independent, *post mortem* analyses of the machines, including those used in the U.S. states of California, Florida, and Ohio [6, 7, 11, 16, 29], unearth flaw after flaw that demonstrate the lack of high assurance techniques in the development processes used by voting system vendors, and the poor quality of the standards they must meet. The intent of valid registered voters, recorded anonymously on a ballot and counted by election officials must be inviolate. There is no reason why this cannot be so. However, there is also no one solution. The U.S. National Institute of Standards and Technology's (NIST) 2007 Voluntary Voting System Guidelines (VVSG) [18]—substantial requirements for design and testing—will undoubtedly help somewhat if enacted, and appear to be headed in the right direction. In this paper, we focus on one key element of electronic voting machines—forensic logging and auditing—that, in concert with other elements of the VVSG2007, would significantly help determine and ensure the correct operation of electronic voting machines and thus ultimately, more fair and accurate elections.

Forensic logging has long been deemed as an essential element of maintaining and validating security [2, 8, 19]. More recently, Yasinsac and Bishop discussed why and how forensic audit trails are essential to validating the correct operation of electronic voting machines [27, 28], in addition to red teaming [6] and code analysis [29]. Further, they discussed how voter receipts are *not* sufficient as audit trails. Forensic logging of the voting machine *system* needs to be performed.

At face value, "auditing" seems to be a simple, well-defined term. Unfortunately, as the complexity of the system increases, so does the complexity of auditing. Given the often conflicting laws and requirements of elections, auditing elections is very difficult. Further, there are no widely accepted, fundamental, election-auditing standards. Therefore, election officials break new ground with each audit that they conduct.

However, there is a key complication in logging on electronic voting machines: if enough data is logged in order to analyze when and how a voting machine is operating incorrectly, how can vote selling and voter anonymity (related to voter coercion) be prevented? By law in most states in the United States, even an election official or forensic analyst performing a post mortem audit or recount should not be able to determine how a particular

978-0-7695-3450-3/09 $25.00 © 2009 IEEE

voter voted. The analyst should *only* be able to analyze the correctness of the machine, and the election official should *only* be able to conduct a recount, but not associate those votes with particular voters.

For example, in Florida, the initial results for an election for the Congressional District 13 (CD13) showed an anomaly: the number of undervotes[1] was an order of magnitude higher than expected for such a high-profile, contentious race. Further, the same anomaly occurred in only one other race out of 25. As there were no *voter-verified paper audit trails* (VVPATs) associated with the machines, the dispute was whether the voting machines correctly recorded the votes that the voters intended to cast. Some believed that the voters did indeed cast their votes for the race and so the source of the undervotes was malfunctioning voting machines. Others thought that the contentiousness of the race had caused voters to skip that particular race. A widely accepted explanation is that poor ballot design caused voters to miss the CD13 race on the ballot. Still, there is little physical evidence to support any hypothesis.

The State of Florida audited the election. Part of that audit involved dynamic testing of the voting machines to see if any problems not revealed before the election arose in this post- election testing. A second part involved a source code audit of the voting system to determine if any software vulnerabilities or flaws could have caused or contributed to the undervotes. The audit concluded that the examined software did not cause or contribute to that problem [29].

During the analysis, the investigators commented that if the machines had a VVPAT, that the VVPAT "audit trail" would have been of little help in the analysis. If the VVPAT showed the undervote, it would be echoing the current results, and again the question would arise whether the undervote was due to the voter not casting a vote for the particular race, or was due to a programming error: exactly the situation without a VVPAT. However, if the VVPAT showed the voter having cast a vote for the race and the electronic count showed an undervote, then the investigators would know that an inconsistency occurred either before the vote was printed (in the case where a voter did not vote, and the VVPAT said she had) or after he vote was printed (in the case where a voter voted, and the electronic record of the ballot said she had not). Thus, the VVPAT could have confirmed that a problem existed without providing any guidance on the cause of the problem.

What auditing mechanisms would be useful to provide the information necessary to audit the system in a way that would uncover problems such as this? Specifically, what are the precise points in the system at which audit mechanisms should be placed and what data should those audit mechanisms record? For example, which system and/or function calls [20] should be saved? At this point, these are all open research questions.

The proper use of appropriate data gathering mechanisms would enhance both security and integrity, because they could be used to track events on the system to detect malicious activity (security) as well as tampering, alterations, and errors involving the recording and counting of votes (integrity). We call the log generated by these mechanisms a *forensic audit trail* (FAT), to distinguish it from the voter-verified audit trail (VVPAT).

But collecting a FAT also leads to a problem. Consider George, who wants to sell his vote. John says that he will pay George if George votes for a certain Presidential candidate named Thomas. John tells George to vote for Thomas by first voting for James, then changing his vote to Andrew, then back to James, then to Thomas. This sequence of votes (James/Andrew/James/Thomas) is highly unlikely to be accidental. If John sees this sequence in a FAT, he can guess that George probably cast that vote. This type of communication uses a covert channel, or a path through which people can communicate but that was not designed for such communication. This type of messaging is anathema to the U.S. election system in general because it enables the selling of votes. Any FAT must guard against this problem.

In this paper we focus on the two key issues of forensic logging of an electronic voting machine: recording enough data to evaluate the operation of a voting machine, but doing so in a way that disallows any manner of reverse-engineering the vote to a voter by using the audit trail.

2 Background

2.1 Forensics

Logging the appropriate data and analyzing it is hard even without anonymity requirements. Op-

[1] An undervote is when a ballot is cast with no valid selection for a particular race.

timizing and correlating the data that is being logged reliably are difficult, but both are essential to a successful analysis. And, as we discussed, in some cases, the needs of a forensic analyst may even directly contradict laws, such as the ability to reverse engineer a malfunctioning voting machine vs. the anonymity of the voter.

A systematic approach, that includes both the laws and some sort of security policy is needed [21]. However, the outcomes of cases in U.S. courts have not yet defined acceptable standards for the composition and admissibility of digital evidence. Therefore, we cannot apply existing legal forensic standards directly to election audits.

We have previously described a model of forensic logging, Laocoön [22], that uses attack graphs based on intruder goals to guide an optimal set of data to be logged and to assist in correlating that data during a post mortem analysis. We demonstrated how a mechanism using the model effectively reduces the possibility of failing to record necessary data, while minimizing the amount of unnecessary data being recorded. Thus, this approach is more effective than current ones and also more efficient for both the computer system recording the forensic data and for the human forensic analyst analyzing the data.

Constructing the attack graphs can be time consuming and difficult for a general-purpose computing system. However, electronic voting machines are not general purpose computers. They have highly restricted modes of operation and well-defined security policies. For example, many violations of security policy on an electronic voting machine (e.g., changing or discarding cast votes) are easy to define precisely. The benefit of the model is that deriving or knowing the *methods* for violating the policies—which is known to be difficult—need not be pre-defined. The result is that only the information that is necessary to analyze to understand the violations or suspected violations is recorded. Without the model, system designers would either have to guess about what is important (creating the risk of missing something) or record everything (thus simultaneously violating anonymity/confidentiality rules and also overwhelming both the auditing system and the human analyst).

Thus electronic voting machines and the related election management systems make ideal environments for the use of the model. Most importantly, Laocoön provides a technological framework for balancing secrecy and anonymity needs with auditing. By using Laocoön to systematically characterize both attacker and defender goals, one can then more easily characterize the threats and targets. By understanding the context of attacks and failures better, an analyst can weigh forensic metrics against privacy metrics to evaluate what information can be revealed to the auditors and therefore what, of the requirements given by Laocoön, can and cannot be safely recorded. In this manner, if a particular state does not want auditors to be able to see how voters cast their ballots, the model leads to one set of constraints. Alternatively, if a state has no such laws, and determines that the auditors are trusted, or establishes procedural mechanisms to provide protection to preserve anonymity, the model leads to a different set of privacy constraints.

2.2 Practical Matters

It is also clear that Laocoön alone is insufficient, and needs to be combined with a procedural model [21]. For example, the system should be built in such a way that the logging mechanism cannot be bypassed and the logs are tamperproof, or measurably tamper-resistant [26]. This requires the use of a reference monitor-like component and write-once storage devices. However, even such low-level mechanisms are not 100% reliable, and so they also must be designed to fail in a safe way.

For example, one way to store the logs is to use a transaction-based system that records each transaction [30] to a separate system with a separate (and perhaps even more rigorous) security domain. This would work in a way that a transaction on the primary system would not proceed without acknowledgment of receipt by the logging system. However, if the link between the two is broken, the primary system should stop functioning. This is vulnerable to a denial-of-service attack, of course, but certain jurisdictions may find this more desirable than over-votes or other actions proceeding without being logged. (For example, they could revert to paper ballots at that point.)

The model must also account for procedural elements outside of any technical context. Doing so can help to verify where to focus effort on additional training for or redundancy in the duties of poll workers. Indeed it is at least as important to know the most critical points to train humans so that effort is not wasted on technological solutions when such a solution is not appropriate. Therefore, the design of an audit system should begin with voting machine requirements, which we have previous discussed. The research questions that we previously posed also drive the construction of the forensic model by giving a starting point for understanding what the system looks like when it's running as it is intended, when it is running in violation of its original intention, and what elements of the system could cause a transition between those two state spaces.

But even system requirements—the policies—for an electronic voting system vary among specific types of machines; a precinct-count optical scanner has very different requirements than a touch-screen DRE,[2] for example. Yet certain requirements apply to all election systems. Thus, the design of an auditing system should begin by refining voting requirements into policies that voting systems must meet. These policies serve as input to the model that guide which data is to be logged. For the remainder of this paper, we focus only on the technological elements, and assume the common elements that all such systems share and face for auditing.

3 Methods for Forensic Logging

The basic requirements for all elections held in the United States are (in no particular order) accuracy, convenience, efficiency, and security. We focus on the first and last items. For our purposes, "accuracy" means that the vote totals reflect the votes cast, and "security" means that no one has tampered with the system in a way that could affect the recording or counting of votes, or compromise the anonymity of the voter or the secrecy of the ballot. How can we detect violations of these requirements, and reconstruct what happened?

The context in which events occur is critical to their correct interpretation. Logging sequences of touches are not enough: the context—the contents of the screen—provides the information that enables the analysts to interpret the log in order to reconstruct the series of events. This allows the analysts to decide whether the touches are meaningful, and if so what they mean. Combined with the other data, such as source code, the auditors will have available, the analysts can use the FAT to reconstruct the voting system's "state" at any point. The system's state includes technical characteristics that are commonly associated with computer systems, such as how long the system has been running or which (anonymous) "user" started it, and characteristics that reflect a voter's perception of the system, such as which language it is displaying information in.

As an example, suppose a voter touches the screen on an electronic voting machine 103 times during a voting session. Those touches are all actions affecting the state of the system, so the positions of these touches are recorded as forensic information. But that information alone does not provide the context for interpreting what those touches meant. So, assume the analyst knows which page of the ballot was being displayed. If 20% of the touches did not connect precisely to a "hot spot," but were near a hot spot, and if immediately after those touches the nearest hot spot was touched precisely, then a forensic analyst could reasonably conclude that the portion of the screen that is supposed to detect a voter's touches was mis-aligned. The additional context of the screen positioning helps the analyst to understand how to interpret the touches, and whether the non-connecting touches were the voter idly tapping the screen, or indicate some other problem.

Fundamentally, the electronic voting machine is also a computer. It might be connected to other electronic voting machines and/or a "supervisor" machine. It may even be able to receive inputs from other hardware sources, such as removable storage devices that can upload new firmware or in some way alter the behavior of the machine. Thus, many of the characteristics of an electronic voting machine can be captured in the same way as for a general-purpose computer system on a network, using logging appropriate for such a situation. This could include capturing "network" traffic, insertion of new software, and replacement of existing software. It could even include recording system and dynamic library calls [20] to an operating system, as is often necessary for forensic analysis of general-purpose computers, unless the system is an embedded system with extremely narrow functionality dictated by the hardware design itself.

This raises the question of limits. How much

[2] A Direct Recording Electronic Voting Machine (DRE) records votes digitally on electronic components (e.g., hard disks or flash memory), as opposed to an Electronic Ballot Marker (EBM), which contains touch-screens and other similar components to a DRE, but the vote is ultimately recorded on a paper ballot, not an electronic object.

information needs to be recorded? How much can be recorded without causing problems, either mechanical (such as running out of memory) or in the ability of the election to meet its requirements (violating ballot secrecy, for example)?

The design of an audit system should begin with voting machine requirements. The requirements are translated into policies and the policies are used to define attack graphs. The attack graphs are formed by starting with an ultimate goal of the attackers—the policy violations—and working backward to—the point of "entry" into the system. As mentioned earlier, we cannot define, let alone enumerate, all *methods* of violation. The model tolerates this by using data about steps toward the policy violation that we do know something about to place bounds on the steps that we do not know anything about. These attack graphs are then translated into specifications and implementations that Laocoön uses to guide logging—what data to log and where to place the instrumentation to log the data. Finally, that data is used by a human analyst to conduct forensic analysis in a rigorous and systematic way.

For example, consider how Laocoön would handle over-voting. Over-voting occurs when more candidates are selected than are allowed in a given race, and electronic voting machines generally disallow over-votes. There are a limited number of ways in which such an event could ultimately be recorded, either on a VVPAT or on electronic media. At some point, the value of a variable or a (set of) bit(s) changes somewhere. There are a limited number of data points that can be manipulated to do this. Describing in advance *how* the data is manipulated is not necessary (and in fact may be impossible). It is sufficient to understanding the possible paths that lead to this manipulation. The paths begin at entry into the system (e.g., touchscreen, supervisor machine, hardware manipulation) and end at the data. These paths must be monitored. This automatically places bounds on the intermediate steps that happen along the path. This allows a system to collect enough information for a forensic analyst to analyze the violation—the traversal of that attack graph by an attacker—and either understand what happened, or determine how to analyze the system further to obtain any necessary information not recorded in the FAT.

One goal of forensics is to reconstruct the state of a machine in sufficient detail that an analyst can understand the sequence of events leading up to the attack, and the attack itself.[3] Suppose such a reconstruction requires the FAT to record enough information to allow a voter to "mark" her ballot so others can determine how she voted. Two obvious techniques are to vote by touching the screen in a predetermined and unusual way (which is recorded in the FAT), or writing in an unusual candidate name such as "Frisbey Q. Perhassenpfeffer." This marks the ballot so election officials or members of an audit team can identify the ballot. This is a classic insider attack [4], unless the state allows the contents of the FAT to become public or untrusted observers are allowed to examine the FAT during its review. Indeed, it has recently been shown with certain datasets that with very little information about a particular record, it is possible to reverse-engineer the identity of the voter [17]. Given the information that *could* be collected by a voting machine, there is substantial cause for concern about how easy it is to defeat anonymization techniques which simply simply strip the names from that data and hope it is anonymous.

As a more detailed example, consider a presidential election with candidates James, Andrew, and Thomas. George wishes to reveal to John how George voted, in a way that John will believe. For example, if John is going to pay George $1,000 to vote for Thomas in the election, John needs to be sure that George actually cast his vote for Thomas before paying him. So they agree to a signaling mechanism by which George will uniquely mark the record of his vote, so that John can verify it is indeed George's vote.

The key idea is that George need not mark his ballot uniquely. As the FAT allows complete reconstruction of all events on the system that George votes on, George need only take a series of options that populates the portion of the FAT (thus, marking it) that corresponds with his voting. John then locates the part of the FAT that corresponds to George's voting. He then reconstructs the sequence of votes, and from that can determine how George voted.

Continuing with our example, George and John mutually agree that before touching the screen to vote for Thomas, few voters will initially vote for James, then change their vote to Andrew,

[3] The other goal is to enable the analyst to present the information to others, such as judges, lawyers, and the body politic.

change it back to James, and finally to Thomas. So George agrees that he will touch the voting machine's screen to vote for candidates in that order. As the system allows the user to change his selections up until the ballot is actually cast, that sequence of touches will cause the final vote for president to be cast for Thomas, as John wants.

Specifically, George enters the polling station and is assigned a voting machine. He walks over and begins to vote. When he reaches the presidential race, he taps the icon next to James. The system marks a vote for James. George then taps the icon next to Andrew; the screen changes the mark to reflect a vote for Andrew, and cancels the mark for James. George touches the icon by James again, changing the vote back to James; then he touches the icon for Thomas. The final vote marked is "Thomas."

After the election is over, an audit is called for because there are indications that a system malfunctioned and votes may have been lost. So the forensic auditors begin auditing the system that George used to vote. John is one of the observers of the audit. He sees the FAT generated for the machine, and looks for the sequence he and George agreed to. When he finds it, he knows that George has voted as he promised, and pays the $1,000. This type of attack exploits a *covert channel*—a path whose primary purpose is not communication, but one that can be used to communicate. In this case, the goal of the FAT is to record events for analysis; it is not designed to be a communication channel in the sense of electronic mail or a telephone line. But that is exactly how George and John are using it.

4 Methods for Enforcing Anonymity

The area of covert channels arises with any shared medium, and it has been studied extensively in computer security, particularly in conjunction with systems that handle different levels of classified materials (e.g., classified, secret, and top secret) [13, 15, 23, 25]. Two general categories of approaches for reducing the amount of communication that can be communicated via covert channels have been identified: enforcing regularity and adding noise.

In this context, the first approach suggests modifying the FAT log entries in some way to make them more regular, so an analyst cannot glean information from the differences. Communication requires changes to information. For example a steady "hum" conveys nothing. If the "hum" starts and stops, the points of starting and stopping can communicate something (much as Morse code is used to communicate something). If the hum varies in intensity, the changes and different intensities can communicate something. Similarly, one analyzes the FAT entries to determine how information can be transmitted using them. In our above example, reordering the screen touches would close the channel that depends on the order of the touches.

This raises two problems. First, the specific nature of the covert channel through the resource controls the countermeasure. The above assumes information is encoded in ordering of touches. But if the information were encoded as *number* of touches for a set of candidates—such as three touches for James and five for Andrew—then rearranging the FAT entries does not obscure the "signal" in the channel.

Even if the countermeasure were appropriate, we have to consider its effects on the use of the FAT. Reordering events within the log changes the analysts' view of the sequence in which those events occurred. This inhibits accurate analysis. As an example, suppose a bug in the code will delete one of James's votes and add one to Thomas's votes whenever someone votes for Andrew. If the sequence of the changes are jumbled so that all votes subtracted from James's votes come first, then the votes for Thomas, and then the votes for Andrew, the connection between the vote changes will be lost—and the FAT will be of minimal use to the auditors.

An alternate approach preserves the existing log entries; it simply obscures them. Instead of rearranging the entries, the system adds spurious log entries to the FAT as the FAT is generated. This follows the second category of approaches for handling covert channels: adding noise. It obscures the "signal" because now the auditors cannot determine which entries are real and which entries are not. Returning to our example, this mechanism might cause two more entries to be added. So rather than the sequence James, Andrew, James, Thomas, the log may show James, Thomas, Andrew, James, Thomas. This complicates the analysis because the spurious entries introduce potential dependencies that the analysts must eliminate.

Indeed, obfuscation has been used to defeat security analysis. Garfinkel discussed its use in "anti-forensics" [12], and Conti specifically addressed attacking visualization systems for security [9]. Other areas include code obfuscation [3], steganography [14], methods for defeating attempts to mine statistical databases [1, 10]. But the goal of the obfuscation proposed is not to defeat the derivation of accurate forensic information; it is designed to close a covert channel, and so the focus differs from these approaches.

A key question is how the addition of FAT entries will affect the analysis of the log. If the new entries reflect problems, then they may cause the analysts to spend time looking at problems not relevant to the goal of the audit. In practice, this is undesirable, so the entries should be added only when they can be inserted without obscuring the sequence of *actions* within the system or the final contents of the ballot. The question is how to guarantee this.

One obvious approach is to generate log entries from sections of code that have a very high assurance of being correct. For example, if two successive screen touches overwrite the same location register, as many screen touches as desired may be added *before* the final one without affecting the results of the vote (as the last candidate touched is the one being voted for). A second approach is to insert bogus log entries that advance the voter to the next screen, enter some random data corresponding to log entries, and then return to the previous screen. With care, log entries for selections that the voter does not make on the future screen can be eliminated simply by adding new FAT entries that undo those selections.

Analysts who have access to the FAT only will be unable to distinguish between randomly generated entries and entries corresponding to actions. But if analysts have access to both the FAT and the source code, then the analysts may be able to compare the FAT to the code for creating log entries and distinguish the two types of entries. To prevent this, the random entries can be generated using an asynchronous timer, causing an interrupt that invokes a routine to add a random FAT entry.

As an example, suppose the screen of candidates defines "hot spots" to be areas $A_1, ..., A_n$ and variables $v_1, ..., v_n$ capture the state corresponding to those areas. So, if the voter has touched area A_i once, then v_i contains the value of "1." Then when the asynchronous routine is invoked, it can either generate a completely random press (to indicate the voter has touched the screen outside the hot spots) or generate a press for any of the given areas. In the latter case, the routine would need to retain the state of the variable corresponding to that area. The system must ensure that on exit (a ballot is cast), the state of that variable is what it would be without the asynchronous routine's interference.

This approach appears to offer significant benefits over rearranging log entries. It does not obscure the sequence of events; that sequence is a (proper) subsequence of the FAT entries. It does obscure both the number of FAT entries and the distance (number of entries) between any pair of FAT entries. Thus, problems existing in the unaugmented FAT would also exist in the augmented FAT.

Although deleting log entries would enhance this technique, in the context of auditing, deleting real entries will hinder the analysis and so are not appropriated.

5 Future Work

The exact algorithms to be used will vary based upon the construction of the software used in the e-voting system. In some cases, the asynchronous approach discussed above would work. In others, for example should the e-voting system not support that approach, perhaps adding code to the FAT logging routines to generate between 0 and n entries before generating the correct entry would provide the appropriate measure of noise to make the bandwidth of the covert channel acceptable. This leads us to a number of interesting theoretical questions.

The first is how to analyze the covert channel to determine the various method that can be used to pass information. We have discussed using sequence information and counts, both of which the random FAT entry technique hinders. Are there others, and if so in what ways can we hinder their use? In answering this question, we must take into account the inability to close covert channels generally, and be content with sufficient "noise" to render the analysts being able to discern the communication as unlikely as desired.

The second is how to integrate the defensive mechanism into the e-voting system's code base in such a way that it will not introduce new problems. Many studies have demonstrated the poor quality of e-voting system code (see for example [6, 5, 16, 29]). The integration of additional routines, particularly asynchronous ones, must be done carefully. Perhaps this type of mechanism will encourage the development of more robust e-voting systems with greater assurance than the current generation of e-voting systems.

6 Conclusions

Much effort and study has gone into the auditing of elections. Virtually no effort and study has gone into the auditing of e-voting systems. The two are fundamentally different.

Auditing an election requires that the votes be verified to have been tallied correctly. The concept of "software independence," introduced by Rivest [24], requires that the election results be demonstrably correct even in the face of software errors. The use of a VVPAT is one mechanism that helps satisfy software independence; even if the e-voting system misrecords votes electronically, the VVPAT can be used as the authoritative record of votes[4] regardless of the electronic count of votes.

Auditing an e-voting system requires that the system be shown to have worked correctly. If the system did not, the audit must show how the system failed. The VVPAT will help minimally here, and only if the VVPAT records can be matched against the electronically recorded ballots. If there is no discrepancy, then either the vote was correctly recorded or *both* the VVPAT and the electronic record misrecorded it. If the two differ, one is wrong. Beyond that, the analysts must guess at where the problem is likely to be, or try to analyze the software directly—a complex task [29].

But detailed audit records, of the sort that this paper calls a FAT, can be used to leak information. They must be obscured if the audit is to be public. Thus, they must be protected in some way, yet in such a way that does not change the results of any audit. Hence the proposed scheme.

One obvious question is, why not simply use trusted auditors and disallow anyone else from seeing the FAT? The answer lies in the realm of the non-technical. If elections are to be transparent, so must the post-election processes. And if a problem arises, the resolution of that problem must also be transparent. This means that the auditors must be able to demonstrate what happened, and why, and how, to others. This would require untrusted people having access to the FAT.

[4] Assuming the VVPAT is implemented correctly, works during Election Day, and voters check that the VVPAT correctly records their vote.

Acknowledgements

Sean Peisert was supported by grant 2006-CS-001-000001 from the U. S. Department of Homeland Security, under the auspices of the Institute for Information Infrastructure Protection & I3P research program. The I3P is managed by Dartmouth College. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security, the I3P, or Dartmouth College.

Matt Bishop gratefully acknowledges the support of award number CNS-0716827 from the National Science Foundation to the University of California at Davis.

Alec Yasinsac was supported in part through DoD Grant H98230-07-1-0221.

References

[1] Nabil R. Adam and John C. Worthman. Security-Control Methods for Statistical Databases: a Comparative Study. *ACM Computing Surveys*, 21(4):515–556, December 1989.

[2] James P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, James P. Anderson Co., Fort Washington, PA, April 1980.

[3] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, and Salil Vadhan Ke Yang. On the (Im)possibility of Obfuscating Programs. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology (Crypto)*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, August 2001.

[4] Matt Bishop, Sophie Engle, Carrie Gates, Sean Peisert, and Sean Whalen. We Have Met the Enemy and He is Us. In *Proceedings New Security Paradigms Workshop (NSPW)*

(to appear), Lake Tahoe, CA, September 22–25, 2008.

[5] Matt Bishop and David Wagner. Risks of E-Voting. *Communications of the ACM*, 50(11):120, November 2008.

[6] Matt Bishop *et al. UC Red Team Report of California Secretary of State Top-to-Bottom Voting Systems Review*, July 2007.

[7] Matt Blaze, Patrick McDaniel, and Giovanni Vigna *et al. EVEREST: Evaluation and Validation of Election-Related Equipment, Standards and Testing*. Secretary of State of Ohio, December 7, 2007.

[8] David Bonyun. The Role of a Well-Defined Auditing Process in the Enforcement of Privacy and Data Security. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, 1980.

[9] Gregory Conti, Mustaque Ahamad, and John Stasko. Attacking Information Visualization System Usability Overloading and Deceiving the Human. In *2005 Symposium on Usable Privacy and Security (SOUPS)*, 2005.

[10] Dorothy E. Denning. Secure Statistical Databases with Random Sample Queries. *ACM Transactions on Database Systems*, 5(3):291–315, September 1980.

[11] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten. Security Analysis of the Diebold AccuVote-TS Voting Machine. `http://itpolicy.princeton.edu/voting`, September 2006.

[12] Simson Garfinkel. Anti-Forensics: Techniques, Detection, and Countermeasures. In *Proceedings of the 2nd International Conference on i-Warfare and Security (ICIW)*, Naval Postgraduate School, Monterey, CA, March 2007.

[13] J. Thomas Haigh, Richard Kemmerer, John McHugh, and William D. Young. An Experience Using Two Covert Channel Analysis Techniques on a Real System Design. *IEEE Transactions on Software Engineering*, 13(2):157–168, Feb 1987.

[14] Neil F. Johnson, Zoran Duric, and Sushil Jajodia. Information Hiding: Steganography and Watermarking—Attacks and Countermeasures. *Journal of Electronic Imaging*, 10(3):825–826, July 2001.

[15] Richard A. Kemmerer. Shared resource matrix methodology: An approach to identifying storage and timing channels. *ACM Transactions on Computer Systems*, 1(3):256–277, August 1983.

[16] Tadayoshi Kohno, A. Stubblefield, Aviel D. Rubin, and Daniel S. Wallach. Analysis of an Electronic Voting System. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, pages 27–40, 2004.

[17] Arvind Narayanan and Vitaly Shmatikov. Robust De-anonymization of Large Sparse Datasets. In *Proceedings of the 29th IEEE Symposium on Security and Privacy*, pages 111–125, Oakland, CA, May 2008.

[18] National Institute of Standards and Technology (NIST). Voluntary voting system guidelines (vvsg). `http://vote.nist.gov/vvsg-report.htm`.

[19] Sean Peisert, Matt Bishop, Sidney Karin, and Keith Marzullo. Principles-Driven Forensic Analysis. In *Proceedings of the 2005 New Security Paradigms Workshop (NSPW)*, pages 85–93, Lake Arrowhead, CA, October 2005.

[20] Sean Peisert, Matt Bishop, Sidney Karin, and Keith Marzullo. Analysis of Computer Intrusions Using Sequences of Function Calls. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 4(2):137–150, April–June 2007.

[21] Sean Peisert, Matt Bishop, and Keith Marzullo. Computer Forensics *In Forensis*. In *Proceedings of the Third International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering (IEEE-SADFE)*, pages 102–122, Oakland, CA, May 22, 2008.

[22] Sean Philip Peisert. *A Model of Forensic Analysis Using Goal-Oriented Logging*. PhD thesis, Department of Computer Science and Engineering, University of California, San Diego, March 2007.

[23] Phillip A. Porras and Richard A. Kemmerer. Covert flow trees: A technique for identifying and analyzing covert storage channels. In *Proceedings of the 1991 Symposium on Security and Privacy*, 1991.

[24] Ronald L. Rivest and John P. Wack. On the Notion of "Software Independece" in Voting Systems. `http://vote.nist.gov/SI-in-voting.pdf`, July 2006.

[25] Marv Schaefer, Barry Gold, Richard Linde, and John Scheid. Program Confinement in KVM/370. In *Proceedings of the 1977 ACM Annual Conference*, pages 404–410, Oct. 1977.

[26] Bruce Schneier and John Kelsey. Secure Audit Logs to Support Computer Forensics. *ACM Transactions on Information and System Security (TISSEC)*, 2(2):159–176, May 1999.

[27] Alec Yasinsac and Matt Bishop. Of Paper Trails and Voter Receipts. In *Proceedings of the 2008 Hawaii International Conference on System Sciences*, January 2008.

[28] Alec Yasinsac and Matt Bishop. The Dynamics of Counting and Recounting Votes. *IEEE Security and Privacy Magazine*, 6(3):22–29, May/June 2008.

[29] Alec Yasinsac, David Wagner, Matt Bishop, Ted Baker, Breno de Medeiros, Gary Tyson, Michael Shamos, and Mike Burmester. *Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware: Final Report For the Florida Department of State*. Security and Assurance in Information Technology Laboratory, Florida State University, Tallahassee, Florida, February 23, 2007.

[30] Dmitrii Zagorodnov, Keith Marzullo, Lorenzo Alvisi, and Thomas C. Bressoud. Engineering Fault-Tolerant TCP/IP Servers Using FT-TCP. In *Proceedings of the 2003 International Conference on Dependable Systems and Networks (DSN)*, 2003.