May 10, 2017

The Honorable Brian Kemp
214 State Capitol
Atlanta, Georgia 30334
(Via email tfleming@sos.ga.gov)

Dear Secretary Kemp:

We write to request your prompt review of Georgia's voting system under the provisions of Georgia Code §21-2-379.2 to assess whether the current voting system "can be safely and accurately used" in the June 20 Congressional District 6 election. Georgia has a long history of voter concerns related to the unverifiable touchscreen voting system with no paper trail. Concerns have escalated because of recent unresolved security issues, as well as the heightened risk of cyber attacks in the current environment. We respectfully request that your office undertake, at a minimum, a partial review of the system to determine whether specific hardware, software, and procedures can be safely and accurately used as required by §21-2-379.2, and separately whether the system is in compliance with applicable federal and state election standards.

Given the indisputable escalation of cyber-security threats in the 15 years since the Diebold system was deployed, this examination is essential for public confidence and security of the upcoming June 20 election. Given the unprecedented national interest in the runoff election, we urge your office to undertake this work immediately. In the likely event that system security deficiences are detected, officials should implement a paper ballot election.

To define the highest-priority areas for our requested review, we have conferred with computer scientists experts in voting system security. We are not seeking a complete "top-to-bottom" certification and laboratory system testing review prior to the conduct of the June 20 election. We are not requesting an immediate recertification of the system. Instead, we request that you initially respond to our concerns by reference to system records. Responses to our listed concerns should be readily available in your office's existing records, staff knowledge and resources, and through conferral with the Center for Election Systems at Kennesaw State University.

We are Georgia electors who believe that the national attention focused on the June 20 special election calls for increased scrutiny concerning the transparency, security, and verifiability of our voting system.

We believe that responses to our listed concerns can be answered within a few hours by knowledgeable staff of your office and the Center for Election Systems. The cost of this reexamination should be modest, and should be borne by the state, not by private citizens. We respectfuly request that you charge only de minimis amounts or waive the requirement that requesting citizens bear the cost of this essential examination conducted for the benefit of all Georgia voters.

The priority areas related to ***safety and accuracy*** of the system listed below must be satisfactorialy addressed in a publicly available report prior to the June 20 election.

Priority areas related to ***safety and accuracy*** of the system include:

1. March 15 Computer Scientist Inquiry
   Leading voting system computer scientists expressed their concerns about Georgia's system and urged you to move the state forward to a system of paper ballots in their March 15 letter attached as Exhibit A. It is our understanding that no response was received from your office. Please address the concerns raised in the letter by disclosing the conclusions made by your office and any mitigating actions taken or planned.

2. Database design
   The attached research, *GEMS Tabulation Database Design Issues in Relation to Voting Systems Certification Standards* by Thomas P. Ryan and Candice Hoke (Exhibit B), presents architecture flaws in the GEMS database design that create unacceptable risks of inaccurate tabulation and reporting. What mitigation has been employed to address these vulnerabilities, and how have any mitigation efforts been tested for adequacy?

3. Malicious attack code threat
   The attached research, *Security Analysis of the Diebold AccuVote-TS Voting Machine* by Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten (Exhibit C), presents detailed and important security weaknesses in the Diebold system. Section 2.2 explains various ways that attack code could be installed. The referenced work is also summarized in these two video recordings—one from a Congressional hearing (https://www.youtube.com/watch?v=HBqGzgxcfAk) and one in a laboratory setting (https://www.youtube.com/watch?v=aZws98jw67g). Have the security weaknesses presented in these videos been mitigated to ensure that the machines can be used "safely and accurately" without realistic security attacks? If so, please provide a description and date of the mitigation efforts.

4. Security of electronic transmission of votes and results
   Votes and results are transmitted via modem from PCMCIA TS memory cards to GEMS servers. Are these election night transmissions of memory card data secured through cryptographic means to prevent interception and possible alteration on their way to the GEMS servers? If so, please provide a description and effective date.

5. Memory card security
   What system software protections prevent the introduction of substituted or modified TS PCMCIA cards prior to TSx uploading to the GEMS server? What system controls are in place to ensure that all precinct cards have been collected and successfully uploaded? What measures prevent forged or maliciously programmed voter access or supervisor cards from transmitting malware to the voting machines?

6. Accessibility of audit logs
   In compliance with VVSG2002 2.2.4.2 and 2.2.5, are cast vote records, TS and TSx audit logs, OS audit logs, and GEMS audit logs readily exportable in human-readable format reports to permit officials, observers, and members of the public to timely review and verify against reported totals? Do current procedures require review of such audit logs for signs of irregularities or system errors?

7. Internet exposure
   What specific guidelines and required processes prevent connecting TS or TSx voting machines or the GEMS servers to the Internet either directly or through the use of removable media (such as flash memory cards) that have been exposed or connected to the Internet?

8. Uploading protocols
   What specific guidelines, required processes, and/or software mechanisms are in place to prevent improper election data, including cast votes, from being uploaded to the GEMS servers either because of human error or software or hardware failures?

9. Encryption key disclosure
   Was the system upgraded and secured against malicious use of the encryption key after it was erroneously published on the Internet? If so, please disclose the date and version numbers of software upgrades or repairs that address the system security issues presented by the widespread knowledge of the system encryption key.

10. <u>ExpressPollbook software flaws</u>
    At the April 22 meeting, the Fulton County Election Board discussed pollbook software errors that caused voters to be sent to improper precincts during the April 18 election. Please explain the source of the software problems and what mitigation steps have been taken to protect the June 20 election from this software issue's harmful impacts and potential voter disenfranchisement.

11. <u>Physical security of voting machines (DREs)</u>
    How are DREs protected from intrusion when not in use, including storage before and after they are delivered to the polling place and during warehouse storage after the election? Given the large number of machines, polling locations, and ease of concealing physical intrusion into the machines, we are concerned that is it impractical to ensure that machines are protected from intrusions that can implant malware. Please reference Exhibit C, section 2.2.1.

12. <u>Compliance with certification standards</u>
    Is the system as currently configured and used certified under federal standards? What standards of certification are required for this specific system configuration under current Georgia law? Is the state certification documentation current?

We are concerned that the system cannot be used safely and accurately, particularly if deficiencies are identified in any of the above controls. As noted in the *Security Analysis of the Diebold AccuVote-TS Voting Machine*, Section 5.4 in Exhibit C, even if the current system configuration is certified to VVSG2002 standards, such certification ***does not imply that the system can be "safely and accurately used"*** as §21-2-379.2(c) requires.

The above list of priority areas is not comprehensive. As we continue to confer with voting system computer scientists, we may amend this letter to add other urgent concerns or remove any less urgent concerns.

We do not seek any proprietary information or security details that would compromise the security of the voting system. Instead, we request a description of the type of review undertaken and a general description of any mitigation adopted that would assure the public that the system is free from previously disclosed security risks.

The software versions we understand to be in current use are listed on Exhibit D. Please inform us if our understanding is inaccurate, and please supply a list of currently installed software.

We also request a copy of the most recent certification documentation for the current voting system and its compliance with applicable Georgia law and election rules.

Dr. Duncan_Buell is our technical adviser and contact point for purposes of discussions with your office. You may contact him through buell@acm.org and 803-479-7128. Dr. Buell is the NCR Chair in Computer Science and Engineering at the University of South Carolina and a voting systems expert.

Thank you for your prompt consideration of our request.

Sincerely,

Mustaque Ahamad
Atlanta, GA 30306

David Bader
Atlanta, GA 30306

Ricardo Davis
Woodstock, Georgia 30188

Richard DeMillo
Atlanta GA 30305

Virginia Forney
Atlanta, GA 30309

Merrick Furst
Atlanta 30306

Adam Ghetti
Atlanta, GA 30324

Jeff Levy
Atlanta, GA 30306

Rhonda J. Martin
Atlanta, GA 30305

Paul Nally
Rydal, GA 30171

Michael S Optiz
Marietta, GA

cc: DeKalb County Elections, H. Maxine Daniels, Director voterreg@dekalbcountyga.gov
Fulton County Elections, Director Richard Barron Richard.Barron@fultoncountyga.gov
Cobb County Election Director Janine Eveler, info@cobbelections.org

March 15, 2017

The Honorable Brian Kemp
214 State Capitol
Atlanta, Georgia 30334

Dear Secretary Kemp,

On March 3rd it was reported that the Federal Bureau of Investigations is conducting a criminal investigation into an alleged cyber attack of the Kennesaw State University Center for Election Systems. According to the KSU Center for Election Systems' website, "the Secretary of State authorized KSU to create a Center for Election Systems, dedicated to assisting with the deployment of the Direct Record Electronic (DRE) voting technology and providing ongoing support."[1] The Center is responsible for ensuring the integrity of the voting systems and developing and implementing security procedures for the election management software installed in all county election offices and voting systems.

The Center has access to most if not all voting systems and software used in Georgia. It also is responsible for programming these systems and accessing and validating the software on these systems. It is our understanding that the Center also programs and populates with voter records the electronic poll books used in polling places statewide. A security breach at the Center could have dire security consequences for the integrity of the technology and all elections carried out in Georgia.

In order for citizens to have faith and confidence in their elections, transparency is crucial, including about events such as the KSU breach, and its extent and severity. While we understand that this investigation is ongoing and that it will take time for the full picture to emerge, we request that you be as forthcoming and transparent as possible regarding critical information about the breach and the investigation, as such leadership not only will be respected in Georgia but also emulated in other states where such a breach could occur. We expect that you are already pursuing questions such as the following, regarding the breach, and trust that you will make public the results of such inquiry:

1. Can you estimate when the attacker breached KSU's system?
2. How did the attacker breach KSU's system?
3. How was the breach discovered?
4. Which files were accessed?
5. Were any files accessed that related to software or "hashes" for the voting machines?
6. Is there any evidence that files were modified? If so, which files?
7. Had KSU begun ballot builds for the upcoming special election?
8. To whom are these attacks being attributed? Could this be an insider attack? Has the FBI identified any suspects or persons of interest?

[1] http://elections.kennesaw.edu/about/history.php

9. Has the FBI examined removable media for the possibility of implanted malware?
10. Has the FBI examined the hash or verification program for tampering?
11. What mitigations are planned for the near- and long-term?

In any state an attack on a vendor providing software and system support with such far-reaching responsibilities would be devastating. This situation is especially fragile, because of the reliance on DRE voting machines that do not provide an independent paper record of verified voter intent. KSU has instead sought to verify the validity of the software on the voting machines by running a hash program on all machines before and after elections in an effort to confirm that the software has not been altered. However, if KSU's election programming were compromised, it is also possible that the verification program could have been modified to affirm that the software is correct, even if it were not. This is a risk of using software to check the correctness of software.

Of course all Georgia elections are important. This month and next include special elections as well. If these upcoming elections are to be run on DREs and e-pollbooks that are maintained and programmed by KSU while the KSU Center for Election Systems is itself the subject of an ongoing criminal investigation, it can raise deep concerns. And today's cyber risk climate is not likely to improve any time soon.

We urge you to provide Georgia's citizens with information they need to confirm before going to vote that their name will appear correctly on the voter rolls, as well as back-up printed voter lists in case anomalies appear. Most importantly, we urge you to act with all haste to move Georgia to a system of voter-verified paper ballots and to conduct post-election manual audits of election results going forward to provide integrity and transparency to all of Georgia's elections. We would be strongly supportive of such efforts and would be willing to help in any way we can.

Sincerely,

Dr. Richard DeMillo
Charlotte B, and Roger C. Warren Professor of Computing
Georgia Tech

Dr. Andrew W. Appel
Eugene Higgins Professor of Computer Science,
Princeton University

Dr. Duncan Buell
Professor, Department of Computer Science & Engineering, NCR Chair of Computer Science & Engineering,
University of South Carolina

Dr. Larry Diamond
Senior Fellow, Hoover Institute and
Freeman Spogli Institute, Stanford University

Dr. David L. Dill
Professor of Computer Science,
Stanford University

Dr. Michael Fischer

Dr. J. Alex Halderman

Professor of Computer Science,
Yale University

Professor, Computer Science and Engineering
Director, Center for Computer Security and Society
University of Michigan

Dr. Joseph Lorenzo Hall
Chief Technologist,
Center for Democracy & Technology

Candice Hoke
Co-Director, Center for Cybersecurity & Privacy Protection and Professor of Law,
Cleveland State University

Harri Hursti
Chief Technology Officer and co-founder, Zyptonite, and founding partner, Nordic Innovation Labs.

Dr. David Jefferson
Lawrence Livermore National Laboratory

Dr. Douglas W. Jones
Department of Computer Science
University of Iowa

Dr. Joseph Kiniry
Principal Investigator, Galois
Principled CEO and Chief Scientist,
Free & Fair

Dr. Justin Moore
Software Engineer, Google

Dr. Peter G. Neumann
Senior Principal Scientist, SRI International Computer Science Lab, and moderator of the ACM Risks Forum

Dr. Ronald L. Rivest
MIT Institute Professor

Dr. John E. Savage
An Wang Professor of Computer Science,
Brown University

Bruce Schneier
Fellow and lecturer
Harvard Kennedy School of Government

Dr. Barbara Simons
IBM Research (retired),
former President Association for Computing Machinery (ACM)

Dr. Philip Stark
Associate Dean, Division of Mathematics and Physical Sciences,
University of California, Berkeley

Dr. Vanessa Teague
Department of Computing & Information systems, University of Melbourne

Affiliations are for identification purposes only, they do not imply institutional endorsements.

GEMS Tabulation Database Design Issues in Relation to Voting Systems Certification Standards

Thomas P. Ryan[1] and Candice Hoke[2]

Abstract

This paper analyzes the Diebold Election Systems, Inc. election management software (GEMS) using publicly accessible postings of GEMS election databases. It finds that the GEMS architecture fails to conform to fundamental database design principles and software industry standards for ensuring accurate data. Thus, in election tabulations, aspects of the GEMS design can lead to, or fail to protect against, erroneous reporting of election results. Further, GEMS's dependence on Microsoft's JET technology introduces additional risks to data accuracy and security.

Despite these technical and systemic deficiencies, GEMS received approval as complying with Federal Voting System 2002 standards. Questions then arise concerning the adequacy of the 2002 and 2005 regulatory standards. The paper concludes that the standards structurally encourage and reward election system vendors for using less exacting database design standards.

With unprecedented Federal funding available to States under the Help America Vote Act of 2002 (HAVA),[3] election administration has become highly reliant on computer technologies. While some continue to praise the new voting and tabulation technologies as a significant advance, the augmented computerization has introduced new possibilities for wide-impact election operational errors and may have opened new avenues for tampering with election results. Previous vulnerability analyses have focused on a direct-recording electronic (DRE) voting machine,[4] a paper ballot optical scanning device,[5] computerized vote-tallying,[6] and a pilot test of internet voting.[7] But the systemic design features of currently utilized election tabulation databases have yet to be closely examined.

This paper analyzes the Diebold Election Systems, Inc. (DESI) election management software named Global Election Management System ("GEMS") using publicly accessible postings of GEMS election databases.[8] It finds that the GEMS architecture violates fundamental design principles and software industry standards for ensuring accurate data. When utilized for election tabulations, the GEMS design can lead to data errors, which in turn create a serious risk for generating erroneous election results. GEMS architectural design plus its use of Microsoft's JET technology,[9] introduces significant risk of data errors in elections administered using GEMS.

Either of these design aspects would be worrisome. For the GEMS database (DB) to have been structured with fundamental flaws at the levels of both system architecture and system technology, and yet still obtain Federal and State certification, raises questions concerning the adequacy of the existing regulatory standards. Thus the paper turns to ask what the relationship is between the regulatory standards and the technical database flaws. It argues the regulatory standards structurally encourage low DB design standards rather than promoting the use of tabulation system architecture that meet widely recognized industry standards for data accuracy and reliability.

This paper proceeds by briefly reviewing the DB design principles of the First and Second Normal Forms. In part II, the paper examines the GEMS DB in light of these fundamental design principles, concluding that GEMS does not satisfy even the most

[1] J.D., 2007; Technical Staff, Center for Election Integrity, Cleveland State University.

[2] Director, Center for Election Integrity and Associate Professor of Law, Cleveland State University. This paper was submitted to EVT/USENIX on April 23, 2007, accepted for publication on June 1, 2007, and will be presented at the EVT '07 Conference on August 6, 2007. A longer version will be available by August 1, 2007 (posted in the Working Papers section, Center for Election Integrity website, www.urban.csuohio.edu/cei/) that is styled for the nontechnical audience. The Center initiated the Collaborative Public Audit of the November 2006 election in Cuyahoga County cited here, and its staff provided technical analysis for the audit.

[3] 42 U.S.C. §§ 15301 – 15545 (2006).

[4] Ariel J. Feldman et al., *Security Analysis of the Diebold AccuVote-TS Voting Machine*, (Sept. 13, 2006), *at* http://itpolicy.princeton.edu/voting/ts-paper.pdf.

[5] Hursti, Hari, *Critical Security Issues with Diebold Optical Scan Design*, (July 4, 2005), *at* http://www.blackboxvoting.org/BBVreport.pdf.

[6] Saltman, Roy G., *Accuracy, Integrity, and Security in Computerized Vote-Tallying*, NBS Special Publication 500-158, (August 1988), http://www.itl.nist.gov/lab/specpubs/500-158.htm.

[7] David Jefferson, Ari D. Rubin, Barbara Simons, David A. Wagner, *A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE)*, at http://www.cs.berkeley.edu/~daw/papers/servereport.pdf

[8] *See* http://www.equalccw.com/dieboldtestnotes.html *and* http://www.bbvforums.org/forums/messages/2197/44189.html This paper's GEMS assessment is perforce limited to examples of the end product but the design flaws are discernible at this level.

basic, essential precepts of the First Normal Form. Further, its use of JET technology renders GEMS susceptible to additional difficulties. Part III critically evaluates the federal regulatory structure and standards for certifying election management software. It concludes that the federal standards produce the unintended and injurious consequence of rewarding poor database designs with lower vendor research and development costs, and faster movement through less intensive certification reviews than if the DB design were more sophisticated.[10]

I. Database Design Fundamentals

Any successful database (DB) must accurately and precisely store data without mixing values or losing information--an obvious essential in managing election results data. To diminish the incidence of anomalies which reduce the accuracy of DB contents, computer science and engineering have established fundamental DB design precepts, including "normalization."[11]

Normalization is a methodology of DB design that creates proper relations, removes redundant data, promotes efficient use of disk space,[12] and reduces the likelihood that accessing and manipulating data will result in anomalies. Normal form classification uses consecutive, progressive numerical titles (e.g., 1NF) to describe in shorthand whether a particular DB has satisfied the fundamental design precepts. If a DB design has not been normalized, the DB has been designed in a manner that fails to prevent avoidable errors and data corruption.[13] For example, when the DB design causes storage of specific data in multiple locations or tables, updates to that data can cause anomalies to occur. Failure to update the specific data in every location virtually simultaneously causes inconsistencies in the data between the two locations (an update anomaly). Normalized DBs also create correct dependencies[14] among data sets. Incorrect dependencies can create errors when data is added or deleted from the DB.

A. Normal Forms

1. *First Normal Form (1NF)*

As Edgar Codd has outlined,[15] satisfaction of the first Normal Form requires a DB design to (a) eliminate repeating groups in individual tables (atomicity);[16] (b) identify each set of related data with a primary key; [17] and (c) create a separate table for each set of related data.[18]

Violations of the first Normal Form (1NF) include the flaws of repeating groups, the absence of unique identifiers, the inclusion of multiple meaningful values in a single field, and the inclusion of multiple columns representing the same type of atomic data. Data corruption is highly probable if any of these violatons are found within the DB design.

2. *Second Normal Form (2NF)*

The overarching purpose of the Second Normal Form (2NF) is to reduce the amount of redundant and duplicate entries within a DB. A DB table satisfies 2NF if (a) it conforms to 1NF and (b) each non-primary key element is dependent upon the primary key.[19] DB satisfaction of 2NF means tables with repeating information separate the repeating data and reference those records through the use of "integrity constraints." Integrity constraints provide a method to ensure data entry changes or updates do not result in a loss of data consistency.[20] The most common tool deployed is known as a foreign key

The first and second Normal Forms contain the most fundamental design principles for efficient and accurate DBs. Any DB that fails to satisfy the first two Normal Forms will suffer various failures upon deployment.

[10]The term "database" within this paper is limited to a modern relational database. Owing to limitations where proprietary software is protected from certain types of evaluative reviews the examination of GEMS DB design and implementation issues is not comprehensive. Instead, the paper seeks to serve as a starting point for future computer science, industry, and regulatory public policy analyses.

[11] Edgar F. Codd, *Normalized Data Base Structure: A Brief Tutorial*, Proceedings of 1971 ACM-SIGFIDET Workshop on Data Description, Access and Control, 1-21 (November 11-12, 1971).

[12] Ponniah, Paulraj, *Database Design and Development*, at 308-9 (3d ed. 2003).

[13]"Data corruption" as used in this paper and in computer science indicates a departure from the original or from what is pure or correct; the term need not import malevolent intent or an operator's deliberate intrusion to modify stored values.

[14] Dependencies are relationships between data, where one value depends on another. To credit the correct number of votes to candidate Joe Smith, one table may specify an identifier as '1234', and then the identifier 1234 is defined in a separate table to be Joe Smith

[15] Codd, Edgar F., *A Relational Model of Data for Large Shared Data Banks*, 13 (6) Comm. of the ACM 13, 377-87 (June 1970) *at* http://www.acm.org/classics/nov95/toc.html.

[16] A repeating group is one that is not atomic, that is, holds more than one meaningful entry per data block.

[17] A primary key is a unique identifier; most commonly the table will start with the integer 1 and increase entry by a factor of one. e.g. 1,2,3,4, etc.

[18] Related data is data with such a strong relationship that it should not be separated. A common example is the elements of an individuals address. (i.e. Street number, street, city, state, zip).

[19] *See* Codd, note 12.

[20] Abraham Silberschatz et al., *Database System Concepts*, 193 (3d ed. 1999).

II. GEMS Database Design Flaws

This paper's analysis of the GEMS DB design is based on review of publicly available GEMS election DBs that are publicly accessible via the internet.[21]

A. System Architecture Design Flaws

Analysis of the GEMS DB architecture[22] demonstrates that it violates both 1NF and 2NF.

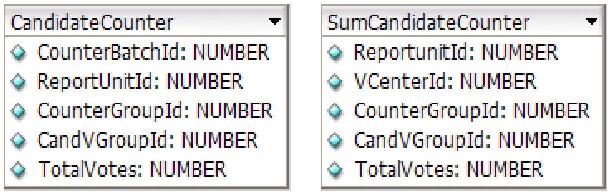
1. *Violations of 1NF*

The GEMS DB design violates fundamental principles of DB architecture for it fails to conform to several 1NF principles. Consequently, GEMS is susceptible to the common errors and anomalies that 1NF seeks to eliminate. Most troubling, nonconformity with 1NF can cause erroneous data to be entered into the DB through normal operation of the system. System failures can then occur without an operator knowing or having any indication that the system is failing because the DB lacks essential design constraints in place to prevent invalid data.

First 1NF Violation: GEMS's Race table violates 1NF because it has multiple columns representing the same type of atomic data. The Race table is structured for two columns to contain the same type of atomic data, VGroup1Id and VGroup2Id, but the purpose of having two columns cannot be distinguished from examining the table alone. Through normal operation, the GEMS design creates unnecessary processing and uses DB storage inefficiently. GEMS thus violates one of the main purposes of the 1NF: eliminating duplicative columns from the DB.[23]

Second 1NF Violation: GEMS includes multiple meaningful values within a single field as demonstrated by the VCenter table of GEMS. The VCenter table holds information regarding polling locations but because of the column "Label," the VCenter table violates 1NF: multiple meaningful values are held within the same field. Combining data in this manner makes it difficult to query voting locations and allows for numerous entries for the same polling location.

Third 1NF Violation: Several GEMS tables lack a unique identifier, a failure demonstrated by review of Figure 1. Within the two Counter tables, the third and forth entries of both Counter tables are indistinguishable from one another other than by their position in the table.


CandidateCounter
- CounterBatchId: NUMBER
- ReportUnitId: NUMBER
- CounterGroupId: NUMBER
- CandVGroupId: NUMBER
- TotalVotes: NUMBER

SumCandidateCounter
- ReportunitId: NUMBER
- VCenterId: NUMBER
- CounterGroupId: NUMBER
- CandVGroupId: NUMBER
- TotalVotes: NUMBER

Figure 1: GEMS tables CandidateCounter and SumCandidateCounter

These tables additionally attempt to avoid data duplication via deployment of MS Access-specific functions, a strategy which has proved to be unreliable.[24]

Other 1NF Violations The design of a number of other GEMS tables crucial to producing accurate election results reports violate 1NF principles, with flaws similar to those inventoried above. [25]

2. *Violations of 2NF*

The GEMS DB design reveals little if no attempt to conform to 2NF principles. GEMS lacks data constraints that ensure data integrity,[26] and omits referential constraints that ensure data consistency.[27] GEMS also replicates the same data throughout numerous locations and tables.

The most troubling violation of the 2NF involves the duplication of data between two tables holding vote tallies. In *Figure 1*, for instance, both CandidateCounter and SumCandidateCounter tables hold total vote data for a candidate -- which is identified by the CandVGroupID column. This dual depositing scheme for election results data can easily generate update anomalies — otherwise known as inconsistencies in election results data for a given candidate and race. After an update anomaly has occurred, it is impossible to rectify the inconsistency without identifying when the anomaly occurred, or starting again from the beginning point when data processing began. If an election ended and the tables held different numbers in each table, the question

[21]*See* note 8 above.

[22] DESI's GEMS software includes components for electronic ballot creation and other tasks but this paper restricts its scope to the GEMS database design for interacting with JET to manage election tabulations and the reporting of results.

[23] Kent, W., *A Simple Guide to Five Normal Forms in Relational Database Theory*, 26 Comm. of the ACM 120-25 (1983).

[25] The longer version of the paper contains this discussion. *See* note 2, above.

[26] Data integrity constraints ensure the data type allowed is the correct one for the field, such as only allowing positive integers for a vote total field.

[27] Foreign keys are not utilized correctly, if at all.

becomes which vote total should be accepted as the correct value.[28]

Eliminating the opportunity for these types of anomalies is precisely the objective of 2NF. GEMS's design thus violates 2NF in tables essential to producing accurate and consistent election results reports.

Additional exemplars of GEMS' departures from 2NF can be identified. DB architects use data integrity constraints to ensure the type of data is correct for a given field. These limits ensure a program cannot input letters where numbers are expected or a negative number where only positive numbers should be allowed. The SumCandidateCounter table (see *Figure 1*), holding vote tally information, violates 2NF in allowing negative values to be entered into the table.

VCenterId is the polling location where votes were registered, and thus should always have a positive value associated with it.[29] A table that permits incorrect negative values, such as *Figure 2*, vitiates one identifier of DB corruption.

Reportunit Id	VCenter Id	Counter GroupId	CandV GroupId	Total Votes
829	-1	0	1	246
829	-1	0	2	45
829	-1	0	3	231
829	-1	0	4	51
829	-1	0	5	252
829	-1	0	6	56
829	-1	0	7	230
829	-1	0	8	49
829	-1	0	9	231

Figure 2, Sample Data from GEMS SumCandidateCounter

Allowing negative values into the VCenter column means tracking the origin of votes cannot occur[30] and the value '-1' might, but does not necessarily, indicate the table and/or DB is corrupted.[31] Placing data constraints upon the VCenter table would force the values to remain positive integers.

Integrity constraints provide the foundation for managing data that resides in several interrelated tables. The election data management code within the GEMS software lacks this essential foundation for data accuracy. Although the GEMS software holds some form of data management, without integrity constraints it is only a matter of time before major problems surface. While modifying the GEMS architecture to include integrity constraints would not completely solve the problem, it would assist in ensuring data accuracy.

B. System Technology Flaws: Use of JET

Microsoft's Joint Engine Technology (JET) is a basic DB engine[32] technology that is appropriate for personal computing and very small scale applications requiring DB technology. Commercially known as Microsoft Access®, JET is a file-sharing DB that can support DBs with sizes up to 2 gigabytes.[33] JET is often considered ideal for small DB deployments with very few concurrent user/processes,[34] and can also be used by custom programs to access the data through the Microsoft Data Access Components Application Programming Interface (MDAC API).

But JET's limitations have led Microsoft (MS) to state that JET is inappropriate for systems that require data integrity, security, and transaction logs and rollbacks.[35]

> Microsoft JET … was not intended (or architected) for the high-stress performance required by 24x7 scenarios, ACID transactions, or unlimited users, that is, scenarios where there has to be absolute data integrity or very high concurrency.[36]

An election management system obviously requires both "absolute data integrity" and in many urban jurisdictions if not all, a "very high concurrency" of users. Thus, the GEMS' architects' choice of inexpensive JET as the DB engine places the entire election tabulation process at very high risk.

[28] The Cuyahoga County (Ohio) Collaborative Public Audit (of the November 8, 2006 General Election) Final Report, which was issued after this paper had been drafted, mentions the inconsistent tables containing election results data as a troubling feature for the accuracy and reliability of its election data. *See* http://urban.csuohio.edu/cei/public_monitor/cuyahoga_2006_audit_rpt.pdf at 34-36 (April 19, 2007; hereafter Cuyahoga 2006 Election Audit Report).

[29] The VCenter table has ids ranging from 1 to 302. There is no entry for -1.

[30] Votes may not be able to be tracked at all if there are duplicate entries for all values in the table.

[31] Repeating values, such as the value "-1," can indicate corruption for some database engines such as Microsoft JET.

[32] A database engine is the underlying software that creates, retrieves, updates, and deletes information from the database.

[33] Luke Chung & Dan Haught, *When to Migrate from Microsoft Access to Microsoft SQL Server*, (2005) *at* http://download.microsoft.com/download/5/d/0/5d026b60-e4be-42fc-a250-2d75c49172bc/when_to_Migrate_from_Access.doc.

[34] Concurrent users and processes are those attempting to use the database at the same time. In the election tabulation context, concurrent usage could include the uploading of election results from scores of DRE units operating simultaneously, or from dozens of optical scanners.

[35] *Microsoft Access or SQL Server: What's Right in your Organization?*, (2005) *at* http://www.microsoft.com/sql/solutions/migration/access/compare-access.mspx

[36] *Using Microsoft JET with IIS*, (Rev. 6.1 2007) *at* http://support.microsoft.com/default.aspx/kb/222135.

1. *Capacity Limit of 2 Gigabytes*

GEMS exacerbates JET's limitations and can lead to DB failure. Its 2 gigabyte limit can easily be exceeded in large turnout urban elections, especially where central count scanning is utilized.[37] If the capacity limit is exceeded, database corruption is highly probable.

2. *Data Corruption During Normal Operation*

As Microsoft documentation has stated:

> When Microsoft JET is used in a multi-user environment, multiple client processes are using file read, write, and locking operations on a shared database. Because multiple client processes are reading and writing to the same database and because Jet does not use a transaction log (as do the more advanced database systems, such as SQL Server), ***it is not possible to reliably prevent any and all database corruption***.[38]

Because this is a file-locking DB system,[39] the operating system (Windows) could function as a "user" that locks the DB file. Corruption of JET DBs can occur from hardware conflicts from peripherals,[40] software conflicts,[41] multi-user access,[42] and an overall poor DB design.[43]

3. *Multi-User Access Limitation*

Software systems that utilize a DB typically have multiple clients or users that attempt to access the data at the same time but the JET DB engine is not designed to manage such simultaneous requests. Microsoft has recommended that fewer than ten clients concurrently access the DB,[44] but single users have also created concurrency errors.[45]

In a GEMS election tabulation, Windows can be one of the processes accessing the DB. In Ohio, during election tabulations the DB is monitored by GEMS as well as a State-mandated security program, DigitalGuardian (DG). Thus, at a minimum, in Ohio GEMS is faced with mediating three potential concurrent clients of the DB. In addition to these three programs, during uploading and processing of election data, GEMS is accessing the DB at a very high rate.

Further, GEMS must mediate a large number of concurrent data requests of the DB during election tabulations. At some points, data is simultaneously being uploaded to the DB from multiple sources (for instance, 30 memory cards); snapshot election results reports are requested (generating data analysis requests from the DB); and software audit logging is occurring (both Windows events logging and GEMS audit logging).

The context of election tabulations ineluctably presents GEMS with a high rate of data concurrency and throughput -- exactly the situation Microsoft has warned can cause DB corruption in its Microsoft JET technology. GEMS cannot be an exception to JET's core deficiencies. This constellation of issues raises very serious questions on whether GEMS is capable of managing and producing accurate election tabulations and other data reports.

4. *Microsoft Deprecation of JET Components*

"Deprecation" is a term used by software companies to notify end users and software developers that a portion of a product line or Application Programming Interface (API)[46] will not be supported in future releases. Microsoft has decided to deprecate MS Data Access Components (MDAC); future releases of JET will not include the MDAC

[37] A Microsoft spokesperson confirmed the Cuyahoga Audit Committee's finding that Microsoft recommended a different system for operations as large as Cuyahoga County's. *See* Bob Driehaus, *Audit Finds Many Faults in Cleveland's '06 Voting, N.Y. Times* Section A, Page 16 (April 20, 2007). The GEMS-JET database can be compressed and backed up but each operation introduces additional risks of database corruption.

[38] *How to Troubleshoot and to Repair a Damaged Access 2002 or Later Database*, (Rev. 6.1 2006) *at* http://support.microsoft.com/default.aspx?scid=kb;en-us;283849 *(emphasis added).; see also NY Times*, note 37 above ("Scott Massey, a Microsoft spokesman, said any file-based database was subject to corruption if a connection was lost while a transfer was in progress"); and Cuyahoga 2006 Audit Report at page 67, cited in note 28 above.

[39] When a process is accessing the database, it prevents all other concurrent access. This "locks out" all other processes until the first process has completed its tasks.

[40] Hardware conflicts occur between hardware devices such as two network cards in use on one machine.

[41] Software conflicts occur between software programs both accessing the database.

[42] Multi-user access conflicts occur when multiple users are using the same program, each accessing the database concurrently.

[43] A poor design includes the lack of normalization described previously within this paper.

[44] "Jet can support up to 255 concurrent users, but performance of the file-based architecture can prevent its use for many concurrent users. In general, it is best to use Jet for 10 or fewer concurrent users." Fitzgerald, James, *Microsoft Data Engine (MSDE) for Microsoft Visual Studio 6.0: An Alternative to Jet for Building Desktop and Shared Solutions*, (2002) *at* http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmsde/html/msdeforvs.asp

[45] *Single User Concurrency Issues with ADO and JET*, (2001) *at* http://support.microsoft.com/kb/216925/EN-US.

[46] Application Programming Interfaces allow programmers to reuse code, such as the code used to communicate to the JET database engine. APIs allow programmers to use the functionality of established code, such as the JET engine, without "reinventing the wheel."

components.[47] The GEMS software utilizes MDAC to communicate with the JET DB engine.[48]

It remains unclear whether DESI was marketing GEMS after MS published the deprecation notice yet omitted disclosure of the point and its consequences to prospective purchasers, election administrative jurisdictions. Software that utilizes the JET MDAC components, such as GEMS, will likely need to be re-written to utilize a different DB technology, and potentially at a high cost for any jurisdictions transitioning to a new GEMS product.

II. Compliance with Federal Voting System Standards

Section 222(e) of HAVA declared the 2002 Voting System Standards (VSS) to be HAVA's first set of voluntary voting system technical standards. Via section 221, Congress authorized a Technical Guidelines Development Committee (TGDC)[49] to develop recommendations for improvements to the Voting System Standards of 2002,[50] in the expansive time frame of nine months. The TGDC's 2005 prolix recommendations address voting system performance standards (Volume I) and testing standards (Volume II).[51] As a matter of federal law, the 2005 VSS remain voluntary rather than compulsory on VS manufacturers although some States have mandated VS equipment be certified by federally approved independent testing laboratories.

Volume I of VSS 2005 incorporates much of the early FEC standards from 2002.[52] The new Security section is written in highly technical language and adds some substantial overdue protections for voting systems technical security. But whatever its improvements for security and other issues, Volume I of the 2005 VSS omits a requirement that qualifying election tabulation databases must satisfy 1NF and 2NF.[53]

But DB design issues fall easily within the scope of voting systems (VS) technical standards and within the TGDC expertise. If the TGDC is able to create and develop detailed standards regarding highly technical security concerns, it would appear also to possess the regulatory scope and technical resources to develop election tabulation DB design and implementation standards.

Volume II of the 2005 VSS is primarily concerned with testing standards for the "qualification" or certification process. It focuses upon the specific details for Independent Testing Authorities (ITAs, now renamed), vendors, and election officials in the qualification process. Like Volume I, Volume II also adopts and reaffirms a vast amount of the 2002 VSS testing standards. The TGDC significantly updated the standards, however, to include changes to reflect the U.S. Election Assistance Commission's process for certification of voting systems and HAVA's usability and accessibility requirements.

In its description of VS testing requirements, Volume II offers no new references regarding election tabulation DB design specifications or testing procedures. Thus, the DB testing standards remain as they were in 2002 without specific requirements or constraints on the designs for an effective and reliable DB.

Volume II lists the documentation that must be provided to the Independent Testing Authority ("ITA") before the election management software (including tabulation functions) can be qualified. In detailing the required vendor DB documentation, the VSS provides the most specific standards for DB design requirements.[54] This Volume II section, however, only requires substantial DB documentation to be provided to the ITA if the specifically listed DB design paradigms were utilized. Those vendors whose DB designs are not reflected in specified paradigms–and thus less likely to be soundly designed--are not required to supply the additional DB documentation. If a vendor chooses to design a DB using paradigms such as entity relationships, or security and privacy constraints, it then must submit substantial documentation to the ITA.

The upshot of this regulatory approach is that the VS vendor who offers a poorly designed tabulation DB that can still meet the minimum requirements set forth in Volume I, Section 2.2.6, can likely reach the testing/certification phase faster than the vendor seeking to market a better designed DB. Moreover,

[47]Microsoft has stated "Starting with version 2.6, MDAC no longer contains Jet components. In other words, MDAC 2.6, 2.7, 2.8, and all future MDAC releases do not contain Microsoft Jet, Microsoft Jet OLE DB Provider, or the ODBC Desktop Database Drivers." Shirolkar, Prash, *Data Access Technologies Roadmap,* (2004) *at* http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmdac/html/data_mdacroadmap.asp

[48] *GEMS User's Guide – Version 1.17.15*, (Rev. 3, 2001) *available at* http://freespeech.metacolo.com/pimaupgrade.zip.

[49] 42 U.S.C § 15361 (2006); the TGDC reports its recommendations to the U.S. Election Assistance Commission.

[50] 42 U.S.C § 15361(b)(2) (2006).

[51] Voluntary Voting System Guidelines Version 1: Initial Report, (2005) *at* http://vote.nist.gov/VVSGVol1&2.pdf. (hereinafter VSS 2005).

[52] The major new sections of this volume include the Human Factors section and the Security section.

[53] Volume I includes basic functions that an election DB must manage, including "identify contests, candidates and issues"; "define ballot formats and appropriate voting options"; "accumulate vote totals at multiple reporting levels as indicated in the system documentation"; "generate the post-voting reports required by Section 2.5" but omits any reference to satisfaction of fundamental DB design requirements. See Vol. I of VSS 2005, section 2.2.6 (cited in note 46). Software standards found in Volume I, section 4, similarly sidestep DB design precepts.

[54] VSS 2005, Volume II, section 2.5.8 (see note 46).

this low horizon vendor will experience substantially lower costs for DB design and VSS required documentation. More documentation requires a larger financial investment for the vendor.

The VSS 2005, and especially Volume II, section 2.5.8, therefore creates a disincentive for election system vendors to design DBs that adhere to sound well, established design paradigms. This incentive runs exactly counter to the overwhelming public interest in accurate and reliable election tabulations. This regulatory inversion regarding DB design standards suggests that the entire VSS 2005 should be analyzed to identify other sections that may inadvertently create incentives that undermine the public interest in accurate, secure elections, and to provide pointers for the next VSS revision.

Conclusion

The GEMS DB has not been designed to adhere to fundamental normalization principles that can permit a DB to reach high standards of accuracy and reliability. But the federal regulatory apparatus has also failed, for it imposes differential documentation requirements and financial burdens on vendors seeking certification for their election DB software. Those vendors who attempt to achieve higher design standards face far greater burdens and costs, including possibly more delays, than vendors who settle for DB designs with lower horizons. None of the 2005 VSS standards constitute a mandatory federal floor for voting systems to be deployed in federal elections. The certification of the GEMS software notwithstanding the significant demonstrable design flaws, offer a clear demonstration of the inadequacy of the current certification regime.

Acknowledgements

The authors appreciate the significant assistance of Andrew W. Appel, David Kettyle, George H. Taylor, and David A. Wagner, and the advice of Aaron Burstein and Joe Hall in preparing this paper. Errors are the responsibility of the authors alone.

Security Analysis of the Diebold AccuVote-TS Voting Machine

Ariel J. Feldman[*], J. Alex Halderman[*], and Edward W. Felten[*,†]

[*]Center for Information Technology Policy and Dept. of Computer Science, Princeton University
[†]Woodrow Wilson School of Public and International Affairs, Princeton University
{ajfeldma,jhalderm,felten}@cs.princeton.edu

Abstract

This paper presents a fully independent security study of a Diebold AccuVote-TS voting machine, including its hardware and software. We obtained the machine from a private party. Analysis of the machine, in light of real election procedures, shows that it is vulnerable to extremely serious attacks. For example, an attacker who gets physical access to a machine or its removable memory card for as little as one minute could install malicious code; malicious code on a machine could steal votes undetectably, modifying all records, logs, and counters to be consistent with the fraudulent vote count it creates. An attacker could also create malicious code that spreads automatically and silently from machine to machine during normal election activities—a voting-machine virus. We have constructed working demonstrations of these attacks in our lab. Mitigating these threats will require changes to the voting machine's hardware and software and the adoption of more rigorous election procedures.

1 Introduction

The Diebold AccuVote-TS and its newer relative the AccuVote-TSx are together the most widely deployed electronic voting platform in the United States. In the November 2006 general election, these machines were used in 385 counties representing over 10% of registered voters [12]. The majority of these counties—including all of Maryland and Georgia—employed the AccuVote-TS model. More than 33,000 of the TS machines are in service nationwide [11].

This paper reports on our study of an AccuVote-TS, which we obtained from a private party. We analyzed the machine's hardware and software, performed experiments on it, and considered whether real election practices would leave it suitably secure. We found that the machine is vulnerable to a number of extremely serious attacks that undermine the accuracy and credibility of the vote counts it produces.

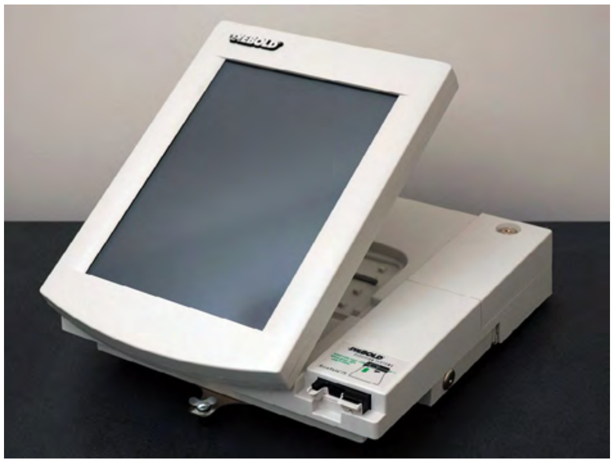


Figure 1: The Diebold AccuVote-TS voting machine

Computer scientists have been skeptical of voting systems of this type, Direct Recording Electronic (DRE), which are essentially general-purpose computers running specialized election software. Experience with computer systems of all kinds shows that it is exceedingly difficult to ensure the reliability and security of complex software or to detect and diagnose problems when they do occur. Yet DREs rely fundamentally on the correct and secure operation of complex software programs. Simply put, many computer scientists doubt that paperless DREs can be made reliable and secure, and they expect that any failures of such systems would likely go undetected.

Previous security studies of DREs affirm this skepticism (e.g., [7, 18, 22, 30, 39]). Kohno, Stubblefield, Rubin, and Wallach studied a leaked version of the source code for parts of the Diebold AccuVote-TS software and found many design errors and vulnerabilities [22]. Hursti later examined the hardware and compiled firmware of AccuVote-TS and TSx systems and discovered problems with the software update mechanism that could allow malicious parties to replace the programs that operate the machines [18]. Our study confirms these results by building working demonstrations of several previously reported attacks, and it extends them by describing a variety of serious new vulnerabilities.

Main Findings The main findings of our study are:

1. Malicious software running on a single voting machine can steal votes with little risk of detection. The malicious software can modify all of the records, audit logs, and counters kept by the voting machine, so that even careful forensic examination of these records will find nothing amiss. We have constructed demonstration software that carries out this vote-stealing attack.

2. Anyone who has physical access to a voting machine, or to a memory card that will later be inserted into a machine, can install said malicious software using a simple method that takes as little as one minute. In practice, poll workers and others often have unsupervised access to the machines.

3. AccuVote-TS machines are susceptible to voting-machine viruses—computer viruses that can spread malicious software automatically and invisibly from machine to machine during normal pre- and post-election activity. We have constructed a demonstration virus that spreads in this way, installing our demonstration vote-stealing program on every machine it infects. Our demonstration virus spreads via the memory cards that poll workers use to transfer ballots and election results, so it propagates even if the machines are not networked.

4. While some of these problems can be eliminated by improving Diebold's software, others cannot be remedied without replacing the machines' hardware. Changes to election procedures would also be required to ensure security.

The details of our analysis appear below, in the main body of this paper.

Given our findings, we believe urgent action is needed to address these problems. We discuss potential mitigation strategies below in Section 5.

The machine we obtained came loaded with version 4.3.15 of the Diebold BallotStation software that runs the machine during an election.[1] This version was deployed in 2002 and certified by the National Association of State Election Directors (NASED) [15]. While some of the problems we identify in this report may have been remedied in subsequent software releases (current versions are in the 4.6 series), others are architectural in nature and cannot easily be repaired by software changes. In any case, subsequent versions of the software should be assumed insecure until fully independent examination proves otherwise.

Though we studied a specific voting technology, we expect that a similar study of another DRE system, whether from Diebold or another vendor, would raise similar concerns about malicious code injection attacks and other problems. We studied the Diebold system because we had access to it, not because it is necessarily less secure than competing DREs. All DREs face fundamental security challenges that are not easily overcome.

Despite these problems, we believe that it is possible, at reasonable cost, to build a DRE-based voting *system*—including hardware, software, and election procedures—that is suitably secure and reliable. Such a system would require not only a voting machine designed with more care and attention to security, but also an array of safeguards, including a well-designed voter-verifiable paper audit trail system, random audits and forensic analyses, and truly independent security review.[2]

Outline The remainder of this paper is structured as follows. Section 2 describes several classes of attacks against the AccuVote-TS machine as well as routes for injecting malicious code. Section 3 discusses the machine's design and its operation in a typical election, focusing on design mistakes that make attacks possible. Section 4 details our implementation of demonstration attacks that illustrate the security problems. Section 5 examines the feasibility of several strategies for mitigating all of these problems. Section 6 outlines prior research on the AccuVote system and DREs more generally. Finally, Section 7 offers concluding remarks.

2 Attack Scenarios

Elections that rely on Diebold DREs like the one we studied are vulnerable to several serious attacks. Many of these vulnerabilities arise because the machine does not even attempt to verify the authenticity of the code it executes. In this section we describe two classes of attacks—vote stealing and denial-of-service [20]—that involve injecting malicious code into the voting machine. We then outline several methods by which code can be injected and discuss the difficulty of removing malicious code after a suspected attack.

[1]The behavior of our machine conformed almost exactly to the behavior specified by the source code to BallotStation version 4.3.1, which leaked to the public in 2003.

[2]Current testing agencies are often referred to as "independent testing agencies" (ITAs), but "independent" is a misnomer, as they are paid by and report to the voting machine vendor.

2.1 Classes of Attacks

2.1.1 Vote-Stealing Attacks

The AccuVote-TS machine we studied is vulnerable to attacks that steal votes from one candidate and give them to another. Such attacks can be carried out without leaving any evidence of fraud in the system's logs. We have implemented a demonstration attack to prove that this is possible; it is described in Section 4.2.

To avoid detection, a vote-stealing attack must transfer votes from one candidate to another, leaving the total number of votes unchanged so that poll workers do not notice any discrepancy in the number of votes reported. Attacks that only add votes or only subtract votes would be detected when poll workers compared the total vote count to the number of voters who signed in at the desk.[3]

The machine we studied maintains two records of each vote—one in its internal flash memory and one on a removable memory card. These records are encrypted, but the encryption is not an effective barrier to a vote-stealing attack because the encryption key is stored in the voting machine's memory where malicious software can easily access it. Malicious software running on the machine would modify both redundant copies of the record for each vote it altered. Although the voting machine also keeps various logs and counters that record a history of the machine's use, a successful vote-stealing attack would modify these records so they were consistent with the fraudulent history that the attacker was constructing. In the Diebold DRE we studied, these records are stored in ordinary flash memory, so they are modifiable by malicious software.

Such malicious software can be grafted into the BallotStation election software (by modifying and recompiling BallotStation if the attacker has the BallotStation source code, or by modifying the BallotStation binary), it can be delivered as a separate program that runs at the same time as BallotStation, it can be grafted into the operating system or bootloader, or it can occupy a virtualized layer below the bootloader and operating system [21]. The machine contains no security mechanisms that would detect a well designed attack using any of these methods. However it is packaged, the attack software can modify each vote as it is cast, or it can wait and rewrite the machine's records later, as long as the modifications are made before the election is completed.

The attack code might be constructed to modify the machine's state only when the machine is in election mode and avoid modifying the state when the machine is performing other functions such as pre-election logic and accuracy testing. The code could also be programmed to operate only on election days. (Elections are often held according to a well-known schedule—for example, U.S. presidential and congressional elections are held on the Tuesday following the first Monday of November, in even-numbered years.) Alternatively, it could be programmed to operate only on *certain* election days, or only at certain times of day.

By these methods, malicious code installed by an adversary could steal votes with little chance of being detected by election officials.[4] Vote counts would add up correctly, the total number of votes recorded on the machine would be correct, and the machine's logs and counters would be consistent with the results reported—but the results would be fraudulent.

2.1.2 Denial-of-Service Attacks

Denial-of-service (DoS) attacks aim to make voting machines unavailable on election day or to deny officials access to the vote tallies when the election ends [20, 28, 3]. It is often known in advance that voters at certain precincts, or at certain times, will vote disproportionately for one party or candidate. A targeted DoS attack can be designed to distort election results or to spoil an election that appears to be favoring one party or candidate. Several kinds of DoS attacks are practical on the AccuVote-TS system because of the ease with which malicious code may be executed.

One style of DoS attack would make voting machines unavailable on election day. For example, malicious code could be programmed to make the machine crash or malfunction at a pre-programmed time, perhaps only in certain polling places. In an extreme example, an attack could strike on election day, perhaps late in the day, and completely wipe out the state of the machine by erasing its flash memory. This would destroy all records of the election in progress, as well as the bootloader, operating system, and election software. The machine would refuse to boot or otherwise function. The machine would need to be serviced by a technician to return it to a working state. If many machines failed at once, available technicians would be overwhelmed. Even if the machines were repaired, all records of the current election would be lost. (We have created a demonstration version of this attack, which is described below in Section 4.4.) A similar style of DoS attack would try to spoil an election by modifying the machine's vote counts or logs in a manner that would be easy to detect but impossible to correct, such as adding or removing so many votes that the resulting totals would

[3]It might be possible to subtract a few votes without detection (if poll workers interpret the missing votes as voters who did not vote in that race) or to add a few votes to compensate for real voters who did not cast ballots; but in any case transferring votes from one candidate to another is a more effective attack.

[4]Officials might try to detect such an attack by parallel testing. As we describe in Section 5.3, an attacker has various countermeasures to limit the effectiveness of such testing.

be obviously wrong. A widespread DoS attack of either style could require the election to be redone.

2.2 Injecting Attack Code

To carry out these attacks, the attacker must somehow install his malicious software on one or more voting machines. If he can get physical access to a machine for as little as one minute, he can use attacks discovered by Hursti [18] to install the software manually. The attacker can also install a voting machine virus that spreads to other machines, allowing him to commit widespread fraud even if he only has physical access to one machine or memory card.

2.2.1 Direct Installation

An attacker with physical access to a machine would have at least three methods of installing malicious software. The first is to create an EPROM chip containing a program that will install the attack code into the machine's flash memory, and then to open the machine, install the chip on its motherboard, and reboot from the EPROM.[5]

The second method is to exploit a back door feature in Diebold's code, first discovered by Hursti. This method allows the attacker to manually install attack software from a memory card. When the machine boots, it checks whether a file named `explorer.glb` exists on the removable memory card. If such a file is present, the machine boots into Windows Explorer rather than Diebold's BallotStation election software. An attacker could insert a memory card containing this file, reboot the machine, and then use Explorer to copy the attack files onto the machine or run them directly from the card. [18]

The third method exploits a service feature of the machine's bootloader, also discovered by Hursti. On startup, the machine checks the removable memory card for a file named `fboot.nb0`. If this file exists, the machine replaces the bootloader code in its on-board flash memory with the file's contents. An attacker could program a malicious bootloader, store it on a memory card as `fboot.nb0`, and reboot the machine with this card inserted, causing the Diebold bootloader to install the malicious software [18]. (A similar method would create a malicious operating system image.)

The first method requires the attacker to remove several screws and lift off the top of the machine to get access to the motherboard and EPROM. The other methods only require access to the memory card slot and power button, which are both behind a locked door on the side of the machine.[6] The lock is easily picked—one member of our group, who has modest locksmithing skills, can pick the lock consistently in less than 10 seconds. Moreover, in their default configuration, all AccuVote-TS machines can be opened with the same key [4], and copies of this key are not difficult to obtain. The particular model of key that the AccuVote-TS uses is identified by an alphanumeric code printed on the key. A Web search for this code reveals that this exact key is used widely in office furniture, jukeboxes, and hotel mini bars, and is for sale at many online retailers. We purchased copies of the key from several sources and confirmed that they all can open the machine.

A poll worker, election official, technician, or other person who had private access to a machine for as little as one minute could use these methods with little risk of detection. Poll workers often do have such access; for instance, in a widespread practice called "sleepovers," machines are sent home with poll workers the night before the election [35].

2.2.2 Voting Machine Viruses

Rather than injecting code into each machine directly, an attacker could create a computer virus that would spread from one voting machine to another. Once installed on a single "seed" machine, the virus would spread to other machines by methods described below, allowing an attacker with physical access to one machine (or card) to infect a potentially large population of machines. The virus could be programmed to install malicious software, such as a vote-stealing program or denial-of-service attack, on every machine it infected.

To prove that this is possible, we constructed a demonstration virus that spreads itself automatically from machine to machine, installing our demonstration vote-stealing software on each infected system. Our demonstration virus, described in Section 4.3, can infect machines and memory cards. An infected machine will infect any memory card that is inserted into it. An infected memory card will infect any machine that is powered up or rebooted with the memory card inserted. Because cards are transferred between machines during vote counting and administrative activities, the infected population will grow over time.

Diebold delivers software upgrades to the machines via memory cards: a technician inserts a memory card containing the updated code and then reboots the machine, causing the machine's bootloader to install the new code from the memory card. This upgrade method relies on the correct functioning of the bootloader, which is supposed to copy the upgraded code from the memory card into the machine's flash memory. But if the bootloader is

[5]When the machine is rebooted, it normally emits a musical chime that might be noticed during a stealth attack; but this sound can be suppressed by plugging headphones (or just a headphone connector) into the machine's headphone jack.

[6]The locked door must be opened in order to remove one of the screws holding the machine's top on.

already infected by a virus, then the virus can make the bootloader behave differently. For example, the bootloader could pretend to install the updates as expected but instead secretly propagate the virus onto the memory card. If the technician later used the same memory card to "upgrade" other machines, he would in fact be installing the virus on them. Our demonstration virus illustrates these spreading techniques.

Memory cards are also transferred between machines in the process of transmitting election definition files to voting machines before an election. According to Diebold, "Data is downloaded onto the [memory] cards using a few [AccuVote] units, and then the stacks of [memory] cards are inserted into the thousands of [AccuVote] terminals to be sent to the polling places." ([10], p. 13) If one of the few units that download the data is infected, it will transfer the infection via the "stacks of [memory] cards" into many voting machines.

2.3 Difficulty of Recovery

If a voting machine has been infected with malicious code, or even if infection is suspected, it is necessary to disinfect the machine. The only safe way to do this is to put the machine back into a known-safe state, by, for example, overwriting all of its stable storage with a known configuration.

This is difficult to do reliably. We cannot depend on the normal method for installing firmware upgrades from memory cards, because this method relies on the correct functioning of the bootloader, which might have been tampered with by an attacker. There is no foolproof way to tell whether an update presented in this way really has been installed safely.

The only assured way to revert the machine to a safe state is to boot from EPROM using the procedure described in Section 3. This involves making an EPROM chip containing an update tool, inserting the EPROM chip into the motherboard, setting the machine to boot from the chip, and powering it on. On boot, the EPROM-based updater would overwrite the on-board flash memory, restoring the machine to a known state. Since this process involves the insertion (and later removal) of a chip, it would probably require a service technician to visit each machine.

If the disinfection process only reinstalled the software that was currently supposed to be running on the machines, then the possibility of infection by malicious code would persist. Instead, the voting machine software software should be modified to defend against installation and viral spreading of unauthorized code. We discuss in Section 5 what software changes are possible and which attacks can be prevented.

3 Design and Operation of the Machine

Before presenting the demonstration attacks we implemented, we will first describe the design and operation of the AccuVote-TS machine and point out design choices that have led to vulnerabilities.

3.1 Hardware

The machine (shown in Figure 1) interacts with the user via an integrated touchscreen LCD display. It authenticates voters and election officials using a motorized smart card reader, which pulls in cards after they are inserted and ejects them when commanded by software. On the right side of the machine is a headphone jack and keypad port for use by voters with disabilities, and a small metal door with a lightweight lock of a variety commonly used in desk drawers and file cabinets. Behind this door is the machine's power switch, a keyboard port, and two PC Card slots, one containing a removable flash memory card and the other optionally containing a modem card used to transfer ballot definitions and election results. The machine is also equipped with a small thermal roll printer for printing records of initial and final vote tallies.

Internally, the machine's hardware closely resembles that of a laptop PC or a Windows CE hand-held device. The motherboard, shown in Figure 2, includes a 133 MHz SH-3 RISC processor, 32 MB of RAM, and 16 MB of flash storage. The machine's power supply can switch to a built-in rechargeable battery in case power is interrupted.

In normal operation, when the machine is switched on, it loads a small bootloader program from its on-board flash memory. The bootloader loads the operating system—Windows CE 3.0—from flash, and then Windows starts the Diebold BallotStation application, which runs the election. Unfortunately, the design allows an attacker with physical access to the inside of the machine's case to force it to run code of her choice [29].

A set of two switches and two jumpers on the motherboard controls the source of the bootloader code that the machine runs when it starts. On reset, the processor begins executing at address 0xA0000000. The switches and jumpers control which of three storage devices—the on-board flash memory, an EPROM chip in a socket on the board, or a proprietary flash memory module in the "ext flash" slot—is mapped into that address range. A table printed on the board lists the switch and jumper configurations for selecting these devices. The capability to boot from a removable EPROM or flash module is useful for initializing the on-board flash when the machine is new or for restoring the on-board flash's state if it gets corrupted, but, as we discussed in Section 2, it could also be used by an attacker to install malicious code.

When we received the machine, the EPROM socket was

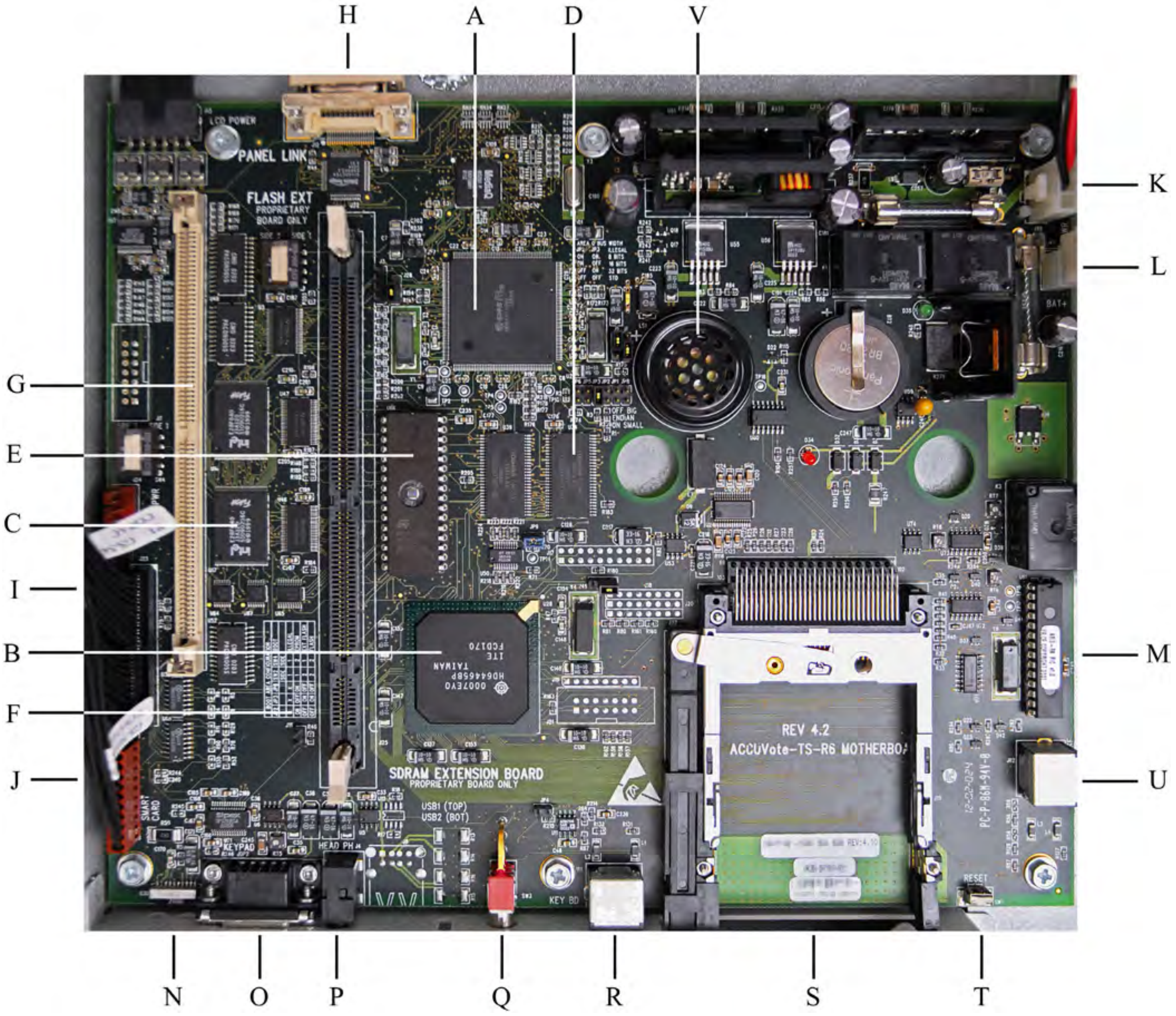


Figure 2: The AccuVote-TS motherboard incorporates a (A) HITACHI SUPERH SH7709A 133 MHZ RISC MICROPROCESSOR, (B) HITACHI HD64465 WINDOWS CE INTELLIGENT PERIPHERAL CONTROLLER, two (C) INTEL STRATA-FLASH 28F640 8 MB FLASH MEMORY CHIPS, two (D) TOSHIBA TC59SM716FT 16 MB SDRAM CHIPS, and a socketed (E) M27C1001 128 KB ERASABLE PROGRAMMABLE READ-ONLY MEMORY (EPROM). A (F) PRINTED TABLE lists jumper settings for selecting the boot device from among the EPROM, on-board flash, or “ext flash,” presumably an external memory inserted in the (G) “FLASH EXT” SLOT.

Connectors on the motherboard attach to the (H) TOUCH SENSITIVE LCD PANEL, (I) THERMAL ROLL PRINTER, and (J) SECURETECH ST-20F SMART CARD READER/WRITER, and receive power from the (K) POWER SUPPLY and (L) BATTERY, which are managed by a (M) PIC MICROCONTROLLER. An (N) IRDA TRANSMITTER AND RECEIVER, (O) SERIAL KEYPAD CONNECTOR, and (P) HEADPHONE JACK are accessible through holes in the machine’s case. A (Q) POWER SWITCH, (R) PS/2 KEYBOARD PORT, and two (S) PC CARD SLOTS can be reached by opening a locked metal door, while a (T) RESET SWITCH and (U) PS/2 MOUSE PORT are not exposed at all. An (V) INTERNAL SPEAKER is audible through the case.

occupied by a 128 KB EPROM containing a bootloader that was older than, but similar to, the bootloader located in the on-board flash. The bootloader contained in the EPROM displays a build date of June 22, 2001 whereas the bootloader contained in the on-board flash displays June 7, 2002. The machine came configured to boot using the on-board flash memory. On our machine, the on-board flash memory is divided into three areas: a 128 KB bootloader, a 3.3 MB GZIP-ed operating system image, and a 10 MB file system partition.

3.2 Boot Process

When the machine is booted, the bootloader copies itself to RAM and initializes the hardware. Then it looks for a memory card in the first PC Card slot, and if one is present, it searches for files on the card with special names. If it finds a file called `fboot.nb0`, it assumes that this file contains a replacement bootloader, and it copies the contents of this file to the bootloader area of the on-board flash memory, overwriting the current bootloader. If it finds a file called `nk.bin`, it assumes that this file contains a replacement operating system image in Windows CE Binary Image Data Format [27], and it copies it to the OS area of the on-board flash, overwriting the current OS image. Finally, if it finds a file called `EraseFFX.bsq`, it erases the entire file system area of the flash. The bootloader does not verify the authenticity of any of these files in any way, nor does it ask the user to confirm any of the changes. As Hursti [18] suggests, these mechanisms can be used to install malicious code.

If none of these files are present, the bootloader proceeds to uncompress the operating system image stored in on-board flash and copy it to RAM, then it jumps to the entry point of the operating system kernel. The operating system image is a kind of archive file that contains an entire Windows CE 3.0 installation, including the kernel's code, the contents of the `Windows` directory, the initial contents of the Windows registry, and information about how to configure the machine's file system.

When Windows starts, the kernel runs the process `Filesys.exe`, which in turn unpacks the registry and runs the programs listed in the `HKEY_LOCAL_MACHINE\Init` registry key [26]. On our machine, these programs are the Debug Shell `shell.exe`, the Device Manager `device.exe`, the Graphics, Windowing, and Events Subsystem `gwes.exe`, and the Task Manager `taskman.exe`. This appears to be a standard registry configuration [25].

The Device Manager is responsible for mounting the file systems. The 10MB file system partition on the on-board flash is mounted at `\FFX`. This partition appears to use the FlashFX file system, a proprietary file system from Datalight, Inc [8]. The memory card, if it is present, is mounted at `\Storage Card`, and may use the FAT or FAT32 file system. The root file system, mounted at \, is stored in RAM rather than nonvolatile memory, which causes any files written to it to disappear when the machine is rebooted or otherwise loses power. This design could be leveraged by an attacker who wished to use the file system for temporarily storing data or malicious code without leaving evidence of these activities.

Diebold has customized `taskman.exe` so that it automatically launches the BallotStation application, `\FFX\Bin\BallotStation.exe`. Another customization causes `taskman.exe` to behave differently depending on the contents of any memory cards in the PC Card slots. If a memory card containing a file called `explorer.glb` is present at start-up, `taskman.exe` will invoke Windows Explorer instead of BallotStation. Windows Explorer would give an attacker access to the Windows Start menu, control panels, and file system, as on an ordinary Windows CE machine. The, `taskman.exe` process also searches the memory card for files with names ending in `.ins` [18]. These files are simple scripts in a Diebold-proprietary binary format that automate the process of updating and copying files. Like the special files that the bootloader recognizes, `taskman.exe` accepts `explorer.glb` without authentication of any kind. While `taskman.exe` requests confirmation from the user before running each `.ins` script, we found multiple stack-based buffer overflows in its handling of these files. This suggests that a malformed `.ins` file might be able to bypass the confirmation and cause the machine to execute malicious code.

3.3 Software and Election Procedures

All of the machine's voting-related functions are implemented by BallotStation, a user-space Windows CE application. BallotStation operates in one of four modes: Pre-Download, Pre-Election Testing, Election, and Post-Election. Each corresponds to a different phase of the election process. Here we describe the software's operation under typical election procedures. Our understanding of election procedures is drawn from a number of sources [34, 13, 36, 40] and discussions with election workers from several states. Actual procedures vary somewhat from place to place, and many polling places add additional steps to deal with multiple voter populations (e.g., different parties or electoral districts) and other complicating factors. We omit these details in our description, but we have considered them in our analysis and, except where noted below, they do not affect the results.

At any given time, the machine's mode is determined by the contents of the currently-inserted memory card. Specifically, the current election mode is stored in the header of the election results file, `\Storage Card\`

`CurrentElection\election.brs`. When one memory card is removed and another is inserted, the machine immediately transitions to the mode specified by the card. In addition, if the machine is rebooted, when BallotStation restarts it will return to the mode specified by the current card. As a result, if a machine is powered off while an election is taking place, it will return to Election mode when it is turned back on.

3.3.1 Election Setup

Typically, the voting machines are stored by the local government or the voting machine vendor in a facility with some degree of access control. Before the election (sometimes the night before, or in other cases the same morning) the machines are delivered to polling places where they are set up and prepared by poll workers. Prior to the election, poll workers may configure BallotStation by inserting a memory card containing a ballot description—essentially, a list of races and candidates for the current election. If, instead, a card containing no recognizable election data is inserted into the machine, BallotStation enters Pre-Download mode. In this mode, the machine can download a ballot definition by connecting to a Windows PC running Diebold's GEMS server software.

After election definitions have been installed, BallotStation enters Pre-Election Testing mode. Among other functions, Pre-Election Testing mode allows poll workers to perform so-called "logic and accuracy" (L&A) testing. During L&A testing, poll workers put the machine into a simulation mode where they can cast several test votes and then tally them, checking that the tally is correct. These votes are not counted in the actual election.

After any L&A testing is complete, the poll workers put the machine into Election mode. The software prints a "zero tape" which tallies the votes cast so far. Since no votes have been cast, all tallies should be zero. Poll workers check that this is the case and then sign the zero tape and save it.

3.3.2 Voting

When a voter arrives at the polling place, she checks in at the front desk, where poll workers give her a "voter card," a special smart card that signifies that she is entitled to cast a vote.[7] The voter inserts her voter card into a voting machine, which validates the card. The machine then presents a user interface that allows the voter to express her vote by selecting candidates and answering questions. After making and confirming her selections, the voter pushes a button on the user interface to cast her vote. The machine modifies the voter card, marking it as invalid, and then ejects it. After leaving the machine, the voter returns the now-invalid voter card to the poll workers, who may re-enable it for use by another voter.

[7]Kohno *et al.* found numerous vulnerabilities and design flaws in BallotStation's smart card authentication scheme [22], which remain uncorrected in the machine we studied.

3.3.3 Post-Election Activities

At the end of the election, poll workers insert an "Ender Card" to tell the voting software to stop the election and enter Post-Election Mode.[8] Poll workers can then use the machine to print a "result tape" showing the final vote tallies. The poll workers check that the total number of votes cast is consistent with the number of voters who checked in at the front desk. Assuming no discrepancy, the poll workers sign the result tape and save it.

After the result tape is printed, the election results are transferred to the central tabulator, a PC running the GEMS software. Like the ballot definitions, the election results may be transferred over a local area network, a phone line, or a serial cable. Once results from all machines have reached the central tabulator, the tabulator can add up the votes and report a result for the election.

For convenience, it is also possible to "accumulate" the results from several machines into a single AccuVote-TS voting machine, which can then transmit the accumulated results to the central tabulator in a single step. To accumulate results, one machine is put into accumulator mode, and then the memory cards from other machines are inserted (in sequence) into the accumulator machine, which reads the election results and combines them into a single file that will be transferred to the central tabulator or used as an input to further accumulation steps.

If a recount is ordered, the result tapes are rechecked for consistency with voter check-in data, the result tapes are checked for consistency with the results stored on the memory cards, and the tabulator is used again to sum up the results on the memory cards. Further investigation may examine the state stored on memory cards and a machine's on-board file system, such as the machine's logs, to look for problems or inconsistencies.

4 Implementing Demonstration Attacks

To confirm our understanding of the vulnerabilities in the Diebold AccuVote-TS system, and to demonstrate the severity of the attacks that they allow, we constructed demonstration implementations of several of the attacks described above and tested them on the machine. We are not releasing the software code for our demonstration attacks to the public at present; however, a video showing

[8]They can also use a "Supervisor Card" for this purpose. Supervisor cards enable access to extra setup and administrative operations in pre- and post-election modes.

some of our demonstration attacks in operation is available online at http://itpolicy.princeton.edu/voting.

4.1 Backup and Restore

As a prerequisite to further testing, we developed a method for backing up and restoring the complete contents of the machine's on-board flash memory. This allowed us to perform experiments and develop other demonstration attacks without worrying about rendering the machine inoperable, and it ensured that we could later restore the machine to its initial state for further testing and demonstrations.

We began by extracting the EPROM chip from its socket on the motherboard and reading its 128 KB contents with a universal EPROM programmer. We then disassembled the bootloader contained on the chip using IDA Pro Advanced [9], which supports the SH-3 instruction set. Next, we created a patched version of the EPROM bootloader that searches any memory card[9] in the first PC Card slot for files named `backup.cmd` and `flash.img`. If it finds a file named `backup.cmd`, it writes the contents of the on-board flash to the first 16 MB of the memory card, and if it finds a file named `flash.img`, it replaces the contents of the on-board flash with the contents of that file. We programmed our modified bootloader into a new, standard, 128 KB EPROM chip and inserted it into the motherboard in place of the original chip. We configured the machine to boot using the code in the chip instead of the normal bootloader in its on-board flash memory, as described in Section 3.

4.2 Stealing Votes

Several of the demonstration attacks that we have implemented involve installing code onto AccuVote-TS machines that changes votes so that, for a given race, a favored candidate receives a specified percentage of the votes cast on each affected machine. Since any attacks that significantly alter the total number of votes cast can be detected by election officials, our demonstration software steals votes at random from other candidates in the same race and gives them to the favored candidate. The software switches enough votes to ensure that the favored candidate receives at least the desired percentage of the votes cast on each compromised voting machine.

Election results (i.e., the record of votes cast) are stored in files that can be modified by any program running on the voting machine. The primary copy of the election results is stored on the memory card at `\Storage Card\CurrentElection\election.brs` and a backup copy is stored in the machine's on-board flash memory at `\FFX\AccuVote-TS \BallotStation\CurrentElection\ election.brs`. Our software modifies both of these files.

Our demonstration vote-stealing software is implemented as a user-space Windows CE application written in C++ that runs alongside Diebold's BallotStation application. Since our software runs invisibly in the background, ordinary users of BallotStation would not notice its presence. It is pre-programmed with three parameters hard-coded into the binary: the name of the race to rig, the name of the candidate who is supposed to win, and the minimum percentage of the vote that that candidate is to receive.

Alternatively, an attacker could create a graphical user interface that allows more immediate, interactive control over how votes would be stolen. We have also created a demonstration of this kind of attack. In practice, a real attacker would more likely design a vote-stealing program that functioned invisibly, without a user interface.

Our demonstration vote-stealing applications can be generalized to steal votes on behalf of a particular party rather than a fixed candidate, to steal votes only in certain elections or only at certain dates or times, to steal votes only or preferentially from certain parties or candidates, to steal a fixed fraction of votes rather than trying to ensure a fixed percentage result, to randomize the percentage of votes stolen, and so on. Even if the attacker knows nothing about the candidates or parties, he may know that he wants to reduce the influence of voters in certain places. He can do this by creating malicious code that randomly switches a percentage of the votes, and installing that code only in those places. Any desired algorithm can be used to determine which votes to steal and to which candidate or candidates to transfer the stolen votes.

Every time a new memory card is inserted into the machine, our demonstration vote-stealing software looks for an election definition file on the card located at `\Storage Card\CurrentElection\election.edb` and, if one is present, determines whether the current election contains a race it is supposed to rig. If no such race is found, the software continues to wait. If a target race is found, it searches that race for the name of the favored candidate. Upon finding that the preferred candidate is on the ballot, the software proceeds to poll the election result files every 15 seconds to see if they have been changed.

If the demonstration vote-stealing software successfully opens the result files during one of its polling attempts, it first checks the result files' headers to see whether the machine is in Election mode. If not, the attack software does not change any votes. This feature ensures that the software would not be detected during Logic and Accuracy testing, which occurs when the machine is in Pre-Election

[9]While Diebold sells special-purpose memory cards for use in the machine, we were able to substitute a CompactFlash card (typically used in digital cameras) and a CompactFlash-to-PC Card adapter.

Testing mode. The software could be further enhanced so that it would only change votes during a specified period on election day, or so that it would only change votes in the presence or absence of a "secret knock." A secret knock is a distinctive sequence of actions, such as touching certain places on the screen, that an attacker executes in order to signal malicious software to activate or deactivate itself.

If the machine is in election mode and the demonstration vote-stealing software successfully opens the result files, then the software checks whether any new ballots have been cast since the last time it polled the files. For each new ballot cast, the software determines whether the race being rigged is on that ballot, and if so, determines whether the corresponding result record contains a vote for the favored candidate or for an opponent. The software maintains a data structure that keeps track of the location of every result record that contains a vote for an opponent of the favored candidate so that it can come back later and change some of those records if necessary. Since each result record is only labeled with the ID number of the ballot to which it corresponds, the software must look up each record's ballot ID in the election definition file in order to determine which candidates the votes in the record are for.

Once it has parsed any newly cast ballots, the software switches the minimum number of votes necessary to ensure that the favored candidate gets at least the desired percentage of the vote. The vote-stealing software chooses which votes to switch by selecting entries at random from its data structure that tracks votes for the opponents of the favored candidate. After the necessary changes have been made to the result files, the software closes the files, resumes the BallotStation process, and continues to wait in the background.

The steps described above are all that is necessary to alter every electronic record of the voters' intent that an AccuVote-TS machine produces. Several of the machine's supposed security features do not impede this attack. The so-called "protective counter," supposedly an unalterable count of the total number of ballots ever cast on the machine, is irrelevant to this attack because the vote-stealing software does not change the vote count.[10] The machine's audit logs are equally irrelevant to this attack because the only record they contain of each ballot cast is the log message "Ballot cast." Furthermore, the fact that election results are stored redundantly in two locations is not an impediment because the vote-stealing software can modify both copies. Finally, as discussed in Section 2, the fact that the election results are encrypted does not foil this attack.

[10]In any event, the "protective counter" is simply an integer stored in an ordinary file, so an attack that needed to modify it could do so easily [22].

4.3 Demonstration Voting Machine Virus

In addition to our demonstration vote-stealing attacks, we have developed a voting machine virus that spreads the vote-stealing code automatically and silently from machine to machine. The virus propagates via the removable memory cards that are used to store the election definition files and election results, and for delivering firmware updates to the machines. It exploits the fact, discovered by Hursti [18], that when the machine boots, the Diebold bootloader will install any code found on the removable memory card in a file with the special name `fboot.nb0`. As a result, an attacker could infect a large population of machines while only having temporary physical access to a single machine or memory card.

Our demonstration virus takes the form of a malicious bootloader that infects a host voting machine by replacing the existing bootloader in the machine's on-board flash memory. Once installed, the virus deploys our demonstration vote-stealing software and copies itself to every memory card that is inserted into the infected machine. If those cards are inserted into other machines, those machines can become infected as well.

The cycle of infection proceeds as follows. When the virus is carried on a memory card, it resides in a 128 KB bootloader image file named `fboot.nb0`. This file contains both the malicious replacement bootloader code and a Windows CE executable application that implements the demonstration vote-stealing application. The vote-stealing executable is stored in a 50 KB region of the bootloader file that would normally be unused and filled with zeroes.

When a card carrying the virus is inserted into a voting machine and the machine is switched on or rebooted, the machine's existing bootloader interprets the `fboot.nb0` file as a bootloader update and copies the contents of the file into its on-board flash memory, replacing the existing bootloader with the malicious one. The original bootloader does not ask for confirmation before replacing itself. It does display a brief status message, but this is interspersed with other normal messages displayed during boot. These messages are visible for less than 20 seconds and are displayed in small print at a 90 degree angle to the viewer. After the boot messages disappear, nothing out of the ordinary ever appears on the screen.

Once a newly infected host is rebooted, the virus bootloader is in control. Since the bootloader is the first code that runs on the machine, a virus bootloader is in a position to affect all aspects of system operation. While booting, the virus bootloader, like the ordinary bootloader, checks for the presence of a memory card in the first PC Card slot. However, if it finds a bootloader software update on the card, it pretends to perform a bootloader update by printing out the appropriate messages, but actually does nothing.[11] Thus, once a machine has been infected, the only way to remove the virus bootloader is to restore the machine's state using an EPROM-resident bootloader.

If a memory card is present, the virus bootloader copies itself to the card as a file named `fboot.nb0` so that it can spread to other machines. If the card already contains a file with that name, the bootloader replaces it. Consequently, if a service technician performing bootloader updates tries to update an infected machine using a card containing an `fboot.nb0` file, the infected machine will not be updated (although it will pretend to be), and all subsequent machines that the technician tries to update using the same card will receive the virus bootloader instead of the updated one. Similarly, updates to the BallotStation software or operating system can also propagate the virus.

The malicious bootloader also copies the vote-stealing executable to the memory card as a file named `AV.EXE`. Then, immediately before starting Windows, the virus bootloader scans the region of RAM occupied by the operating system image (0x8C080000–0x8C67FFFF) for the hard-coded string in the `taskman.exe` binary that points to the BallotStation executable `\FFX\Bin\BallotStation.exe` and replaces it with `\Storage Card\AV.EXE`. Consequently, when Windows starts, `taskman.exe` will launch the demonstration vote-stealing application instead of BallotStation.

When the demonstration vote-stealing application on the memory card starts, it first renames the legitimate BallotStation executable to `\FFX\Bin\AccuVote.exe`, and then it copies itself to the machine's on-board flash memory with the name `\FFX\Bin\BallotStation.exe`. It adopts the name of the BallotStation executable so it will still run at start-up even if the machine is booted without a memory card in the first PC Card slot. Next, it copies the malicious bootloader image from the card to the on-board flash . Thereafter, the software periodically checks whether an uninfected memory card is present in the machine, and, if so, it copies the virus files onto the card so that other machines where the card is used will become infected. Finally, the vote-stealing application runs in the background, changing votes in the manner described in Section 4.2.

[11] In order to avoid printing out fake update messages when the copy of `fboot.nb0` on the card was put there by the virus bootloader itself, whenever the virus bootloader copies itself to a card, it sets the hidden, system, and read-only FAT attributes of the resulting `fboot.nb0` file. Then, when the virus bootloader checks for the presence of the `fboot.nb0` file on the card, it only prints out fake update messages if the file does not have those attributes. Alternatively, the virus bootloader could identify copies of itself by examining the contents of the `fboot.nb0` file for some characteristic bit string.

4.4 Demo Denial-of-Service Attack

To illustrate how malicious software running on an AccuVote-TS could launch a denial-of-service (DoS) attack, we developed a demonstration attack program that, on command, erases the contents of both the currently-inserted memory card and the machine's on-board flash memory. This attack not only destroys all records of the election currently in progress (both the primary and backup copies), but also renders the machine inoperable until a service technician has the opportunity to dismantle it and restore its configuration.

The demonstration DoS program is comprised of a user-space Windows CE executable that triggers the attack and a malicious bootloader that functions like an ordinary bootloader, except that upon receiving the appropriate signal, it completely erases the currently-inserted memory card and the machine's on-board flash memory. The user-space trigger program works by first writing a special value to a part of the machine's on-board flash memory that is accessible from user-space programs and then crashing the machine by invoking the `PowerOffSystem()` Windows CE API call. The `PowerOffSystem()` API is supposed to put the system in a low-power "sleep" mode from which it can later "wake-up," but when this API is invoked on an AccuVote-TS, the machine simply crashes. When the machine is rebooted (which must be done manually), the malicious bootloader notices that the special value has been written to the machine's on-board flash memory. On this signal, it completely erases the contents of both the currently-inserted memory card and the machine's on-board flash memory. In so doing, the malicious bootloader destroys all of the data, software, and file system formatting on both the memory card and the on-board flash memory.

In order to account for the possibility that the malicious bootloader never gets a chance to completely erase both storage media or that the memory card is removed before the machine is rebooted, the user-space trigger program deletes as much as it can before crashing the machine. It deletes all of the files on the memory card and on the machine's on-board `\FFX` file system including both the primary and backup copies of the election results (`election.brs`), the audit logs (`election.adt`), and the BallotStation executable. When it deletes these files, it overwrites each of them with garbage data to make it less likely that the files will ever be recovered.

While our demonstration DoS attack is triggered by a user's command, a real attacker could create malicious software that only triggers the above attack on a specific date and time, such as on election day. An attacker could also design the attack to launch in response to a specific trend in voting during an election, such as an apparent victory for a particular candidate. Like a vote-stealing attack, a DoS attack could be spread by a virus.

5 Mitigation

The vulnerabilities that we have described can be mitigated, to some extent, by changing voting machine designs and election procedures. In this section we discuss several mitigation strategies and their limitations.

5.1 Software and Hardware Modifications

The AccuVote-TS machine is vulnerable to computer viruses because it automatically loads and runs code found on memory cards without authenticating it. Its software could be redesigned to inhibit the spread of viruses, however. One approach would be to digitally sign all software updates and have the machine's software verify the signature of each update before installing it. Such a change would ensure that only updates signed by the manufacturer or another trusted certifying authority could be loaded.[12] It would also be helpful to require the person using the machine to confirm any software updates. Confirmation of updates would not prevent a malicious person with physical access to the machine from loading an update, but at least it would make the accidental spread of a virus less likely while the machine was being used by honest election officials.

While redesigning the voting machine's software can help mitigate some of the security problems that we identify, there are other problems inherent in the AccuVote-TS hardware architecture that cannot be addressed by software changes. For example, there is nothing to stop an adversary who has physical access to the machine from booting and installing his own malicious software by replacing the socketed EPROM chip on the motherboard. Furthermore, because all of the machine's state is kept in rewritable storage (RAM, flash memory, or a memory card), it is impossible to create tamper-proof logs, records, and counters. In addition, as is the case with ordinary PCs, it is difficult to determine with certainty that the machine is actually running the software that it is supposed to run. Rootkit techniques [16] and virtualization technologies [21], which are often used to conceal malware in the PC setting, could be adapted for use on the voting machines.

Researchers have proposed various strategies for building specialized hardware capable of maintaining tamper-proof and tamper-evident logs, records, and counters (e.g., [37]), as well as software strategies that provide more limited protection (e.g., [33]). Although such methods could prevent attacks that aim to alter votes after they have been recorded, they could not prevent malicious code from changing future votes by altering data before it is sent to the storage device.

Assuring a computer's software configuration is also a notoriously difficult problem, and research has focused on mechanisms to ensure that only approved code can boot [1] or that a machine can prove to a remote observer that it is running certain code [37]. For example, commercial systems such as Microsoft's Xbox game console have incorporated mechanisms to try to resist modification of the boot code or operating system, but they have not been entirely successful [17]. Although mechanisms of this type are imperfect and remain subjects of active research, they seem appropriate for voting machines because they offer some level of assurance against malicious code injection. It is somewhat discouraging to see voting machine designers spend much less effort on this issue than game console designers.

While changes to the hardware and software of voting machines can reduce the threats of malicious code injection and log tampering, no purely technical solution can eliminate these problems.

5.2 Physical Access Controls

Despite the best efforts of hardware and software designers, any physical access to a computer still raises the possibility of malicious code installation, so election officials should limit access to voting machines' internals, their memory cards, and their memory card slots to the extent possible.

There is some benefit in sealing the machine's case, memory card, and card bay door with individually numbered tamper-evident seals, in the hope of detecting illicit accesses to these areas. While these measures may expose some classes of attacks, they make denial-of-service attacks easier. Suppose, for example, that a malicious voter cuts a seal while an election is in progress. If machines with broken seals are treated as completely untrustworthy, then cutting the seal is itself an effective denial-of-service attack. If broken seals are usually ignored when everything else seems to be in order, then an attacker has a good chance of successfully inserting malicious code that cleans up after itself. There seems to be no fully satisfactory compromise point between these two extremes.

Even leaving aside the possibility that voters will deliberately break seals, broken seals are an unfortunately common occurrence. The most comprehensive study of AccuVote DRE election processes in practice examined the May 2006 primary election in Cuyahoga County, Ohio, which used AccuVote-TSx machines. The study found that more than 15% of polling places reported at least one

[12] Adding signatures would not be effective if a machine has already been infected with malicious code; machines would need to be booted from EPROM and completely restored to a known state before their software were updated to a version that checked signatures.

problem with seals [13].

The available evidence is that machines and memory cards are not handled with anything approaching the necessary level of care. For example, the Cuyahoga County study [13] reported many procedural weaknesses: "A lack of inventory control and gaps in the chain of custody of mission critical assets (i.e. DRE memory cards, [DREs], . . .)" (p. 103); "the systems of seals, signatures and other security features of the. . . machine memory cards were not implemented on a consistent basis" (p. 109); "It appears that memory cards are regularly removed and re-inserted when a DRE becomes out-of-service. Security tabs are broken and no log of this remove and replace activity is maintained. . . There is no indication that a record comparing memory card to DRE serial number is kept" (p. 138); "Security seals are not checked for integrity at the end of Election Day, nor are they matched with a deployment list of Security seal serial numbers. There is no attempt to reconcile memory cards intended for the precinct with memory cards removed from the DREs at the end of the day. . . Therefore, it is unknown whether these memory cards were tampered with during Election Day" (p. 139); "There is no established chain of custody during the transfer of the memory cards. . . from the vote center to the BOE [Board of Elections]" (p. 140); "Security seals are collected upon return to the BOE, but these serial numbers are neither logged nor checked against the original security seal serial numbers deployed with the memory cards. Therefore, it is unknown whether these memory cards were tampered with during transport to the BOE from the polling location" (p. 140). These problems require immediate attention from election officials.

Security seals do some good, but it is not a solution simply to assume that seals will always be used, always be checked, and never be broken. Inevitably, some seals will be missing or broken without an explanation, providing potential cover for the insertion of malicious code or a voting machine virus.

5.3 Effective Parallel Testing

In parallel testing, election officials choose some voting machines at random and set them aside, casting simulated votes on them throughout election day and verifying at the end of the election that the machines counted the simulated votes correctly. The goal of parallel testing is to trigger and detect any vote-stealing software that may be installed on the machines.

A challenge in parallel testing is how to make the simulated voting pattern realistic. If the pattern is unrealistic in some respect—if, say, the distribution of votes throughout the day doesn't match what a real voting machine would see—then vote-stealing software may be able to tell the difference between a real election and parallel testing, allowing the software to steal votes in the real election while leaving results unchanged in parallel testing.

Parallel testing is also vulnerable to a "secret knock" attack by a testing insider. Generally, parallel tests are carried out by representatives from all political parties to ensure impartiality. However, if one representative has placed vote altering code on the machines, she could disable the code on the machine being tested by issuing a surreptitious command. For example, the code might watch for a specific sequence of touches in a normally unused area of the screen and deactivate its vote altering function in response. Preventing this kind of attack requires carefully scripting the testing procedure.

Alternatively, a secret knock might be used to activate malicious code. In this scheme, malicious voters would perform the secret knock on the machines being used to collect real votes, or a malicious election worker would perform it surreptitiously when setting up the machines, and vote-stealing software would wait for this secret knock before operating. Machines chosen for parallel testing would not see the secret knock and so would count votes honestly. This approach has the drawback (for the attacker) of requiring a significant number of malicious voters or a malicious poll worker to participate, though these participants would not have to know all the details of the attack.

These possibilities reduce the usefulness of parallel testing in practice, but we think it can still be a worthwhile precaution when conducted according to rigorously controlled procedures.

5.4 Effective Whole-System Certification

Despite their very serious security flaws, the Diebold DREs were certified according to federal and state standards. This demonstrates that the certification processes are deficient. The Federal Election Commission's 2002 Voting System Standards [14] say relatively little about security, seeming to focus instead on the machine's reliability if used non-maliciously.

The U.S. Election Assistance Commission issued voluntary voting system guidelines [38] in 2005. These are considerably more detailed, especially in the area of security, than the FEC's 2002 standards. Though it would not be entirely fair to apply the 2005 guidelines to the pre-2005 version of the AccuVote software we studied, we do note that the AccuVote-TS hardware architecture may make it impossible to comply with the 2005 guidelines, in particular with the requirement to detect unauthorized modifications to the system software (see [38], Volume I, Section 7.4.6). In practice, a technology can be deployed despite noncompliance with certification requirements if the testing agencies fail to notice the problem.

In general, the certification process seems to rely more on testing than analysis. Testing is appropriate for some properties of interest, such as reliability in the face of heat, cold, and vibration, but testing is ill-suited for finding security problems. As discussed frequently in the literature, testing can only show that a system works under specific, predefined conditions; it generally cannot ensure that there is no way for an attacker to achieve some goal by violating these conditions. Only a competent and thorough security analysis can provide any confidence that the system can resist the full range of realistic attacks.

Weak certification would be less of a problem if information about the system's design were more widely available to the public. Researchers and other experts would be able to provide valuable feedback on voting machine designs if they had the information to do so. Ideally, strong certification procedures would be coupled with public scrutiny to provide the highest assurance.

5.5 Software-Independent Design

Although the strategies described above can contribute to the integrity of election results, none are sufficient to mitigate the vote-stealing attacks that we have demonstrated. The only known method of achieving an acceptable level of security against the attacks we describe is software-independent design. "A voting system is *software-independent* if an undetected change or error in its software cannot cause an undetectable change or error in an election outcome [31]." In the near term, the only practical way to make DREs software-independent is through the use of a voter-verifiable paper audit trail (VVPAT) coupled with random audits. The VVPAT creates a paper record, verified visually by the voter, of how each vote was cast. This record can be either a paper ballot that is deposited by the voter in a traditional ballot box, or a ballot-under-glass system that keeps the paper record within the voting machine but lets the voter see it [24]. A VVPAT makes our vote-stealing attack detectable. In an all-electronic system like the Diebold DREs, malicious code can modify all of the logs and records in the machine, thereby covering up its vote stealing, but the machine cannot modify already created paper records, and the accuracy of the paper records is verified by voters.

Paper trails have their own failure modes, of course. If they are poorly implemented, or if voters do not know how or do not bother to check them, they may have little value [3, 13]. The real advantage of a paper trail is that its failure modes differ significantly from those of electronic systems, making the combination of paper and electronic record keeping harder to defraud than either would be alone. Requiring a would-be vote stealer to carry out both a code-injection attack on the voting machines and a physical ballot box stuffing attack would significantly raise the difficulty of attacking the system.

Paper ballots are only an effective safeguard if they are actually used to check the accuracy of the machines. This need not be done everywhere. It is enough to choose a small fraction of the polling places at random and verify that the paper ballots match the electronic records there. If the polling places to recount are chosen by a suitable random procedure, election officials can establish with high probability that a full comparison of paper and electronic records would not change the election's result. Methods for conducting these random audits are discussed by Rivest [2] and Calandrino, *et al.* [6], among others.

Another limitation of VVPATs is that they cannot stop a denial-of-service attack from spoiling an election by disabling a large number of voting machines on election day. Given this possibility, if DREs are used, it is worthwhile to have an alternative voting technology available, such as paper ballots.

In the future, cryptographic voting may provide an alternative means of achieving software-independence that offers greater security than VVPATs. Cryptographic voting systems (e.g., [32, 5]) aim not only to allow voters to verify that their votes were recorded as cast, but also to allow them to confirm that their ballots were actually included in the final vote totals. Currently, however, achieving acceptable usability and maintaining ballot secrecy remain challenges for such schemes (see [19]).

6 Related Work

Several previous studies have criticized the security of the Diebold AccuVote DRE systems. The first major study of these machines was published in 2003 by Kohno *et al.* [22], who did not have access to a machine but did have a leaked version of the source code for BallotStation. They found numerous security flaws in the software and concluded that its design did not show evidence of any sophisticated security thinking. They did not study the AccuVote-TS's kernel or bootloader, however.

Public concern in light of Kohno's study led the state of Maryland to authorize two security studies. The first study, by SAIC, reported that the system was "at high risk of compromise" [34]. RABA, a security consulting firm, was hired to do another independent study of the Diebold machines. RABA had access to a number of machines and some technical documentation. Their study [30] was generally consistent with Kohno's findings, and found some new vulnerabilities. It suggested design changes to the Diebold system, and outlined some steps that Maryland might take to reduce the risk of security problems. The state responded by adopting many of RABA's suggestions [23].

A further security assessment was commissioned by the Ohio Secretary of State and carried out by the Compuware Corporation [7]. This study examined several

DRE systems, including the AccuVote-TS running the same version of BallotStation as our machine, and identified several high risk security problems.

In 2006, in response to reports that Harri Hursti had found flaws in Diebold's AccuBasic subsystem, the state of California asked Wagner, Jefferson, and Bishop to perform a study of AccuBasic security issues. Their report [39] identified several vulnerabilities that differ from those that we describe because the machine that we studied lacks the AccuBasic subsystem.

Later in 2006, Hursti released a report [18] describing several security weaknesses in the AccuVote-TS and -TSx systems that could allow an attacker to install malicious software by subverting the systems' software update mechanisms. These weaknesses form the basis for many of the attacks that we describe in the current study. With limited access to the voting machines, Hursti could only confirm that one of these weaknesses could be exploited; we show that many of the others can be as well.

Our work builds on these previous reports. Our findings generally confirm the behaviors and vulnerabilities described by Kohno *et al.*, RABA, and Hursti, and demonstrate through proof-of-concept implementations that the vulnerabilities can be exploited to implement viral attacks and to change election results. To our knowledge, our work is the first comprehensive, public description of these threats to Diebold's DREs.

Several studies discuss general issues in the construction of software-based attacks on DRE voting machines. Kelsey [20] catalogs the attacker's design choices; our analysis confirms that all or nearly all of the attack options Kelsey discusses can be carried out against the Diebold machine we studied. The Brennan Center report [3] offers a broader but less technical discussion; its discussion of malicious software injection attacks is based partially on Kelsey's analysis.

Additionally, there is an extensive literature on electronic voting in general, which we will not attempt to survey here.

7 Conclusion

From a computer security standpoint, DREs have much in common with desktop PCs. Both suffer from many of the same security and reliability problems, including bugs, crashes, malicious software, and data tampering. Despite years of research and enormous investment, PCs remain vulnerable to these problems, so it is doubtful, unfortunately, that DRE vendors will be able to overcome them.

Nevertheless, the practical question facing public officials is whether DREs provide better security than other election technologies, which have their own history of failure and fraud. DREs may resist small-scale fraud as well as, or better than, older voting technologies; but DREs are much more vulnerable to large-scale fraud. Attacks on DREs can spread virally, they can be injected far in advance and lurk passively until election day, and they can alter logs to cover their tracks. Procedures designed to control small-scale fraud are no longer sufficient—DREs demand new safeguards.

Electronic voting machines have their advantages, but experience with the AccuVote-TS and other paperless DREs shows that they are prone to very serious vulnerabilities. Making them safe, given the limitations of today's technology, will require safeguards beginning with software-independent design and truly independent security evaluation.

Acknowledgments

We thank Andrew Appel, Jeff Dwoskin, Laura Felten, Shirley Gaw, Brie Ilenda, Scott Karlin, Yoshi Kohno, David Robinson, Avi Rubin, Adam Stubblefield, Dan Wallach, and Harlan Yu for technical help, information, and feedback. We are especially grateful to the party who provided us the machine to study.

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] ARBAUGH, W., FARBER, D., AND SMITH, J. A secure and reliable bootstrap architecture. In *Proc. 1997 IEEE Symposium on Security and Privacy*, pp. 65–71.

[2] ASLAM, J., POPA, R., AND RIVEST, R. On estimating the size of a statistical audit. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07).*

[3] BRENNAN CENTER TASK FORCE ON VOTING SYSTEM SECURITY. *The Machinery of Democracy: Protecting Elections in an Electronic World.* 2006.

[4] BROACHE, A. Diebold reveals 'key' to e-voting? *CNet News.com* (Jan. 2007). Available at http://news.com.com/2061-10796_3-6153328.html.

[5] C. A. NEFF. Practical high certainty intent verification for encrypted votes. Available at http://www.votehere.com/vhti/documentation/vsv-2.0.3638.pdf, 2004.

[6] CALANDRINO, J., HALDERMAN, J. A., AND FELTEN, E. Machine-assisted election auditing. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07).*

[7] COMPUWARE. Direct recording electronic (DRE) technical security assessment report. Available at http://www.sos.state.oh.us/sos/hava/compuware112103.pdf, 2003.

[8] DATALIGHT. FlashFX product details. Available at http://datalight.com/products/flashfx/productdetails.php.

[9] DATARESCUE. IDA Pro Disassembler. Available at http://www.datarescue.com/idabase.

[10] DIEBOLD ELECTION SYSTEMS. Checks and balances in elections equipment and procedures prevent alleged fraud scenarios. Available at http://www.votetrustusa.org/pdfs/DieboldFolder/checksandbalances.pdf, 2003.

[11] DIEBOLD ELECTION SYSTEMS. Press release: State of Maryland awards Diebold electronic voting equipment order valued at up to $55.6 million. Available at http://www.diebold.com/news/newsdisp.asp?id=2979, 2003.

[12] ELECTION DATA SERVICES. 2006 voting equipment study. Available at http://www.edssurvey.com/images/File/ve2006_nrpt.pdf.

[13] ELECTION SCIENCE INSTITUTE. DRE analysis for May 2006 primary, Cuyahoga County, Ohio. Available at http://bocc.cuyahogacounty.us/GSC/pdf/esi_cuyahoga_final.pdf, Aug. 2006.

[14] FEDERAL ELECTION COMMISSION. Voting system standards. Available at http://www.eac.gov/election_resources/vss.html, 2002.

[15] HAUG, N. Vendor proposal evaluation findings report and addendum. Available at http://www.sos.state.oh.us/sos/hava/findings091003.pdf, 2003.

[16] HOGLUND, G., AND BUTLER, J. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley, 2005.

[17] HUANG, A. *Hacking the Xbox: An Introduction to Reverse Engineering*. No Starch Press, 2003.

[18] HURSTI, H. Critical security issues with Diebold TSx (unredacted). Available at http://www.bbvdocs.org/reports/BBVreportIIunredacted.pdf, May 2006.

[19] KARLOF, C., SASTRY, N., AND WAGNER, D. Cryptographic voting protocols: A systems perspective. In *Proc. 14th USENIX Security Symposium* (2005).

[20] KELSEY, J. Strategies for software attacks on voting machines. In *NIST Workshop on Threats to Voting Systems* (2005).

[21] KING, S., CHEN, P., WANG, Y., VERBOWSKI, C., WANG, H., AND LORCH, J. SubVirt: Implementing malware with virtual machines. In *Proc. 2006 IEEE Symposium on Security and Privacy*, pp. 314–327.

[22] KOHNO, T., STUBBLEFIELD, A., RUBIN, A., AND WALLACH, D. Analysis of an electronic voting system. In *Proc. 2004 IEEE Symposium on Security and Privacy*, pp. 27–42.

[23] MARYLAND STATE BOARD OF ELECTIONS. Response to: Department of legislative services trusted agent report on Diebold AccuVote-TS voting system. Available at http://mlis.state.md.us/Other/voting_system/sbe_response.pdf, 2004.

[24] MERCURI, R. *Electronic Vote Tabulation: Checks and Balances*. PhD thesis, University of Pennsylvania, 2001.

[25] MICROSOFT. Configuring the process boot phase. Available at http://msdn2.microsoft.com/en-us/library/ms901773.aspx.

[26] MICROSOFT. Filesys.exe boot process. Available at http://msdn2.microsoft.com/en-us/library/ms885423.aspx.

[27] MICROSOFT. Windows CE binary image data format specification. Available at http://msdn2.microsoft.com/en-us/library/ms924510.aspx.

[28] NEUMANN, P. Security criteria for electronic voting. In *16th National Computer Security Conference* (1993).

[29] OPEN VOTING CONSORTIUM. Worst ever security flaw found in Diebold TS voting machine. Available at http://www.openvotingconsortium.org/blog/2006-jul-31/worst_flaw_ever_in_diebold_touch_screen_voting_machine_revealed, 2006.

[30] RABA TECHNOLOGIES. Trusted agent report: Diebold AccuVote-TS voting system. Available at http://www.raba.com/press/TA_Report_AccuVote.pdf, 2004.

[31] RIVEST, R., AND WACK, J. On the notion of "software independence" in voting systems. Available at http://vote.nist.gov/SI-in-voting.pdf, July 2006.

[32] S. POPOVENIUC AND B. HOSP. An introduction to punchscan. Available at http://www.punchscan.org/papers/popoveniuc_hosp_punchscan_introduction.pdf, 2006.

[33] SCHNEIER, B., AND KELSEY, J. Cryptographic support for secure logs on untrusted machines. In *Proc. 7th USENIX Security Symposium* (1998).

[34] SCIENCE APPLICATIONS INTERNATIONAL CORPORATION. Risk assessment report: Diebold AccuVote-TS voting system and processes (unredacted). Available at http://www.bradblog.com/?p=3731, 2003.

[35] SONGINI, M. E-voting security under fire in San Diego lawsuit. *Computerworld* (Aug. 2006).

[36] STATE OF MARYLAND. Code of Maryland regulations, Title 33, State Board of Elections. Available at http://www.dsd.state.md.us/comar/subtitle_chapters/33_Chapters.htm.

[37] TRUSTED COMPUTING GROUP. TCG TPM specification. Available at https://www.trustedcomputinggroup.org/specs/TPM.

[38] UNITED STATES ELECTION ASSISTANCE COMMISSION. Voluntary voting systems guidelines. Available at http://www.eac.gov/vvsg_intro.htm, 2005.

[39] WAGNER, D., JEFFERSON, D., AND BISHOP, M. Security analysis of the Diebold AccuBasic interpreter. Available at http://www.ss.ca.gov/elections/voting_systems/security_analysis_of_the_diebold_accubasic_interpreter.pdf, Feb. 2006.

[40] WILLIAMS, B. Security in the Georgia voting system. Available at http://www.votescount.com/georgia.pdf, 2003.

Exhibit D

Assumed software versions in use in Georgia for 2017 elections

Optical Scan
AccuVote OS 1.94W

Touch Screen
R6 – Ballot Station 4.5.2! *
TSx – Ballot Station 4.5.2! *

ExpressPoll
Express Poll 2.1.2
Security Key 4.5

Election Management System
GEMS 1.18.22 G

Source-- Georgia's Logic and Accuracy Testing Manual v1.4
(https://www.eac.gov/assets/1/28/Logic_and_Accuracy_Testing_Manual_Final_v1.4.pdf)

*Note: We assume that the version in use is 4.5.2 as issued by Diebold/Premier and installed without modifications. We assume that the "!" in the version number is a stray typo with no significance.