ACCURATE
A CENTER FOR CORRECT, USABLE, RELIABLE,
AUDITABLE, AND TRANSPARENT ELECTIONS

PUBLIC COMMENT ON THE VOLUNTARY VOTING SYSTEM GUIDELINES, VERSION 1.1*

Submitted to
The United States Election Assistance Commission

September 28, 2009

*This material is based upon work supported by the National Science Foundation under A Center for Correct, Usable, Reliable, Auditable and Transparent Elections (ACCURATE), Grant Number CNS-0524745. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This public comment narrative was prepared by Aaron Burstein and Joseph Lorenzo Hall in consultation with the ACCURATE Principal Investigators and Advisory Board Members Lillie Coney, David Jefferson, and Whitney Quesenbery. These comments benefited from contributions by Andrew Appel, Matt Bishop, Paco Hope, Sean Peisert, Eric Rescorla, Gregg Vanderheiden, and Ka-Ping Yee.

ACCURATE Principal Investigators

Aviel D. Rubin
ACCURATE Director
Department of Computer Science
Johns Hopkins University
rubin@cs.jhu.edu
http://www.cs.jhu.edu/~rubin/

Dan S. Wallach
ACCURATE Associate Director
Department of Computer Science
Rice University
dwallach@cs.rice.edu
http://www.cs.rice.edu/~dwallach/

Dan Boneh
Department of Computer Science
Stanford University
dabo@cs.stanford.edu
http://crypto.stanford.edu/~dabo/

Michael D. Byrne
Department of Psychology
Rice University
byrne@rice.edu
http://chil.rice.edu/byrne/

Drew Dean
Computer Science Laboratory
SRI International
ddean@csl.sri.com
http://www.csl.sri.com/users/ddean/

David L. Dill
Department of Computer Science
Stanford University
dill@cs.stanford.edu
http://verify.stanford.edu/dill/

Jeremy Epstein
Computer Science Laboratory
SRI International
jepstein@csl.sri.com
http://www.csl.sri.com/people/epstein/

Douglas W. Jones
Department of Computer Science
University of Iowa
jones@cs.uiowa.edu
http://www.cs.uiowa.edu/~jones/

Deirdre K. Mulligan
School of Information
University of California, Berkeley
dkm@ischool.berkeley.edu
http://www.ischool.berkeley.edu/
people/faculty/deirdremulligan

Peter G. Neumann
Computer Science Laboratory
SRI International
neumann@csl.sri.com
http://www.csl.sri.com/users/neumann/

David A. Wagner
Department of Computer Science
University of California, Berkeley
daw@cs.berkeley.edu
http://www.cs.berkeley.edu/~daw/

Preface

A Center for Correct, Usable, Reliable, Auditable and Transparent Elections (ACCURATE),[1] a multi-institution, interdisciplinary, academic research center funded by the National Science Foundation's (NSF) "CyberTrust Program,"[2] is pleased to provide these comments on the Voluntary Voting System Guidelines Version 1.1 (VVSG v1.1) to the Election Assistance Commission (EAC).

ACCURATE was established in 2005 to conduct fundamental research into methods for improving voting technology. ACCURATE's Principal Investigators direct research investigating software architecture, tamper-resistant hardware, cryptographic protocols and verification systems as applied to electronic voting systems. Additionally, ACCURATE evaluates voting system usability and how public policy, in combination with technology, can better support elections.

Since receiving NSF funding in 2005, ACCURATE has made many important contributions to the science and policy of electronic voting.[3] The ACCURATE Center has published groundbreaking results in security, cryptography, usability, and verification of voting systems. ACCURATE has also actively contributed to the policy discussion through regulatory filings, through testimony and advising decisionmakers as well as conducting policy research.[4] ACCURATE researchers have participated in running elections and assisting election officials in activities such as unprecedented technical evaluation of voting systems and redesigning election procedures.[5] Finally, the education and outreach mission of ACCURATE has flourished through the development of numerous undergraduate and graduate classes and the creation of the premier venue for voting technology research.[6]

With experts in computer science, systems, security, usability, and technology policy, and knowledge of election technology, procedure, law and practice, ACCURATE is uniquely positioned to provide helpful guidance to the EAC as it attempts to strengthen the specifications and requirements that ensure the functionality, accessibility, security, privacy and trustworthiness of our voting technology.

We welcome this opportunity to further assist the EAC and hope this process continues the collaboration between the EAC and independent, academic experts in order to sustain improvements in election systems and procedures.

---

[1] *See:* http://www.accurate-voting.org/.

[2] National Science Foundation Directorate for Computer & Information Science & Engineering, CyberTrust, *see:* http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=13451&org=CISE.

[3] A Center for Correct, Usable, Reliable, Auditable and Transparent Elections. *2006 Annual Report*. Jan. 2007. URL: http://accurate-voting.org/wp-content/uploads/2007/02/AR.2007.pdf; A Center for Correct, Usable, Reliable, Auditable and Transparent Elections. *2007 Annual Report*. Jan. 2008. URL: http://accurate-voting.org/wp-content/uploads/2008/01/2007.annual.report.pdf; A Center for Correct, Usable, Reliable, Auditable and Transparent Elections. *2008 Annual Report*. Jan. 2009. URL: http://accurate-voting.org/wp-content/uploads/2008/12/2008annualreport.pdf

[4] *List of ACCURATE Testimony*. ACCURATE Website. URL: http://accurate-voting.org/pubs/testimony/; A Center for Correct, Usable, Reliable, Auditable and Transparent Elections. *Public Comment on the 2005 Voluntary Voting System Guidelines*. Sept. 2005. URL: http://accurate-voting.org/accurate/docs/2005_vvsg_comment.pdf; A Center for Correct, Usable, Reliable, Auditable and Transparent Elections. *Public Comment on the Voluntary Voting System Guidelines, Version II (First Round)*. May 2008. URL: http://accurate-voting.org/wp-content/uploads/2008/05/accurate_vvsg2_comment_final.pdf.

[5] ACCURATE researchers have participated in voting system evaluations sponsored by the States of California, Florida, Kentucky, Ohio and the District of Columbia.

[6] For more on our educational output, please see those sections of our Annual Reports (*see* note 3). The Electronic Voting Technology workshop (EVT), collocated with the USENIX Security Symposium, was started in 2006 and continues to attract the highest caliber voting technology research. *See:* http://www.usenix.org/event/evtwote09/.

Contents

1 Introduction and Background

The EAC's proposed revision of VVSG 2005 will require extensive changes to current voting systems, yet yield modest benefits on key issues of accessibility, usability, reliability, accuracy, and security. The commission claims that its proposal aims to revise technical requirements that do not require changes in current voting system hardware or "complex software changes,"[1] and to include VVSG II provisions that clarify existing requirements or improve testing; the Commission notes that VVSG II commenters were "near[ly] unanimous" in praising these provisions.[2] The EAC positions the proposed revision as a set of relatively minor technical changes and non-controversial changes in testing requirements.[3]

While the inclination to proceed incrementally is understandable (although we believe ill-advised as a policy and market matter), the changes proposed in the draft VVSG v1.1 in some instances will require complex software changes and omit some crucial improvements in testing recommended in the draft VVSG II. Most importantly, we believe the proposed revisions are not well targeted. In our last public comment on the draft VVSG II we highlighted four essential improvements—software independence, adversarial vulnerability testing (OEVT), usability benchmark testing and volume testing—necessary to make substantial progress.[4] However, the EAC'S approach is not currently moored in a substantive prioritization of risks that need be addressed. The impact of this gap is most profound in proposed security revisions that lack the core requirement for software independence recommended by NIST in the draft VVSG II. There are also practical issues surrounding the transition from VVSG 2005 to VVSG v1.1 that amplify our concern with the substantive proposals. Specifically, it is unclear how the revision will affect voting systems that are currently being tested under VVSG 2005 or that will have obtained certification by the time the EAC adopts VVSG v1.1. Additionally, under the current draft, VVSG v1.1 will go into effect immediately upon adoption by the EAC, without the transition period of past VVSG, therefore further complicating the certification environment in an already troubled market.

Viewed in isolation, many of the proposed technical requirements would improve upon VVSG 2005. However the proposed changes must be assessed for their effects in promoting trustworthiness in entire voting systems. While the proposed changes will yield improvements, they neither target the most substantial problems nor address second and third tier problems in an economical and efficient fashion. For this reason we find them misguided and inefficient because in the long run they are likely to result in greater costs for less benefit than a more comprehensive or risk-oriented approach. It is important to note that the VVSG II recommendations that provide the basis for the new software requirements in VVSG v1.1 were formulated against the backdrop of the principle of *software independence*, which holds that "an undetected error or fault in the voting system's software is not capable of causing an undetectable change in election results."[5] In the absence of this core principled change to the structure

[1] U.S. Election Assistance Commission. *Proposed Guidance on Voluntary Voting System Guidelines*. 74 Federal Register 26665. June 2009, 26666.

[2] Ibid., 26666.

[3] The EAC states two broad goals in issuing the draft VVSG v1.1: improving testing under the VVSG and revising certain requirements to "reflect advancements in voting technology." (Ibid., 26666) The EAC did not issue a final version of the "next iteration" of the VVSG, commonly known as "VVSG II" or "VVSG 2007," to achieve these goals. Instead, it decided to replace portions of VVSG 2005 with portions of VVSG II. *See:* (U.S. Election Assistance Commission, Technical Guidelines Development Committee. *Voluntary Voting System Guidelines Recommendations to the Election Assistance Commission*. Aug. 2007. URL: `http://www.eac.gov/files/vvsg/Final-TGDC-VVSG-08312007.pdf`; U.S. Election Assistance Commission. *Draft Voluntary Voting System Guidelines Version 1.1*. May 2009. URL: `http://www.eac.gov/program-areas/voting-systems/voting-system-certification/2005-vvsg/draft-revisions-to-the-2005-voluntary-voting-system-guidelines-vvsg-v-1-1`, Introduction).

[4] A Center for Correct, Usable, Reliable, Auditable and Transparent Elections. *Public Comment on the Voluntary Voting System Guidelines, Version II (First Round)*. May 2008. URL: `http://accurate-voting.org/wp-content/uploads/2008/05/accurate_vvsg2_comment_final.pdf`.

[5] U.S. Election Assistance Commission, Technical Guidelines Development Committee, *VVSG II*, see n. 3, Vol. 1, § 2.4.

of voting systems the proposed revisions lose their importance. Picking and choosing elements from VVSG II guided by a desire to avoid complex and/or controversial revisions will produce costly yet substantively unsatisfactory results.

The EAC has yet to decide what to do with the TGDC's software independence recommendation, and the proposed VVSG v1.1 must be understood in light of this uncertainty.[6] The software changes proposed in VVSG v1.1 cannot make up for its omission, and these changes do little to lay the groundwork for software independence in the future.[7] Some changes will not be fully realized without requiring software independence; for example, requiring voting systems to generate electronic data in auditable, structured formats is an extremely valuable improvement (as discussed in Section 3) but not nearly as powerful as it would be when embedded in a framework for software independent voting systems. Similarly, though VVSG v1.1's cryptography requirements would probably remedy some of the past misuses in voting systems' use of cryptography, it is unrealistic to expect better cryptography to produce a leap forward in the integrity of election results without software independence. As discussed in Section 4, other components of voting systems will remain vulnerable to attacks that do not require subverting their cryptographic modules. Finally, some of the new software security requirements push (or exceed) the state of the art but bring a questionable benefit to the security of the voting system as a whole. In Section 5 we point out that these requirements either focus on workmanship improvements that are likely to yield marginal improvements in testability and security or require software validation methods that lack foolproof real-world implementations.

The changes in voting system testing and documentation hold greater promise for improving voting system testing and increasing voting system trustworthiness in the near term. However, the testing and documentation changes that are included in VVSG v1.1 would benefit from further refinement. Yet, here too we are disappointed to see that what we believe were core recommendations in VVSG II with respect to testing—adversarial vulnerability testing (OEVT) and volume testing—were not included.[8] While the new more stringent and precise requirements for accuracy and reliability testing are welcome, the absence of adversarial testing and volume testing misses an opportunity to centralize at the federal level a expensive types of security and reliability testing that has proved valuable. The requirements for performance-based usability benchmarks and testing mark major advances in the assessment of voting systems. Yet, as we note in Section 7, test protocols for usability benchmarking being developed by NIST—which are not included in VVSG v1.1—are critically important. We recommend that they be added to VVSG v1.1 if they are ready; if not, the EAC should encourage their completion and add them to the standards as soon as possible. The documentation requirements in VVSG v1.1 hold considerable promise for encouraging voting system vendors to adopt sound security engineering practices and for setting a high standard for security testing and evaluation during the certification process. If these were coupled with core substantive revisions, including software independence, adversarial vulnerability test-

[6]EAC had planned to revise the VVSG II and submit the revision for another round of public comment before adoption. However, at the beginning of this year, this plan was changed to include an interim VVSG (VVSG v1.1) before taking substantially more time to consider the draft VVSG II and the associated public comments. *See:* U.S. Election Assistance Commission. *Implementation Plan for 2005 VVSG Revision / Next Iteration*. Jan. 2009. URL: `http://www.eac.gov/program-areas/voting-systems/docs/vvsg-timeline-update-2005-31march09/attachment_download/file`.

[7]Nor would they push voting technology in a direction that would serve an alternative to software independence. For example, in response to direction from the EAC to "[d]evelop possible alternatives to the requirement of Software Independence," NIST has offered "auditability"—ensuring that "any error in [a voting system's] recording of votes or vote totals, whether randomly occurring or maliciously induced, is detectable." National Institute of Standards and Technology. *EAC Reasearch Areas for the TGDC VVSG Recommendations*. Jan. 2009. URL: `http://www.eac.gov/program-areas/voting-systems/docs/nist-response.pdf/attachment_download/file`, 1, 3

[8]As explained in U.S. Election Assistance Commission, Technical Guidelines Development Committee, *VVSG II*, see n. 3, 3:3.4, open-ended vulnerability testing involves attempts by skilled testers to falsify the assertion that a system is secure by demonstrating that vulnerabilities in a system can be exploited. Volume testing, discussed in Section 6, involves testing a large amount of exemplar voting systems under conditions meant to simulate the election environment.

ing, usability benchmarking and volume testing, progress would be certain. We discuss these changes, and suggest some further revisions, in Section 8.

2 Issues Arising from the Transition from VVSG 2005 to VVSG v1.1

This section discusses some of the issues involved in the transition from VVSG 2005 to VVSG v1.1.

2.1 Effects of the Transition on the Market for Voting Systems

Given that the Commission expects the VVSG 1.1 to be in effect for two years before issuing the next iteration of the VVSG,[9] how the proposed revision will affect manufacturers and jurisdictions is an issue worth considering. First, it is unclear how the revision will affect systems that are currently under testing.[10] Our understanding of the EAC's certification program is that any system certified under VVSG 2005 will be able to maintain its certification, provided that none of the conditions that might trigger decertification[11] arise. The revision's effects on systems that are undergoing testing is less clear. Will manufacturers be required to make changes to their systems to conform to new requirements governing cryptography, software development and workmanship, and other substantive technical revisions? Or will manufacturers and VSTLs have the option of continuing testing under the VVSG 2005? Current EAC guidance on this subject—primarily the *Testing and Certification Manual*—does not answer these questions.

Second, the VVSG 1.1 could create some uncertainty for jurisdictions that are considering purchasing new voting systems. A division between systems certified to VVSG 2005 and VVSG 1.1 might encourage jurisdictions to wait until the latter are available.[12] This would prolong states' dependence on voting systems that were certified to the 2002 VSS (or earlier standards).[13]

2.2 Effective Date of VVSG v1.1

Further complicating matters is the fact that the Commission proposes to make VVSG 1.1 effect immediately upon adoption.[14] This is in contrast to the 24-month waiting period between the adoption of VVSG 2005 and their going into effect.[15] This choice is understandable from the standpoint that the EAC intends to make immediate improvements in testing and technical requirements until it adopts the VVSG II in final form. The Commission is also assuming that the changes in requirements are minor

[9] U.S. Election Assistance Commission, *Implementation Plan for 2005 VVSG Revision / Next Iteration*, see n. 6.

[10] U.S. Election Assistance Commission. *Voting System Testing and Certification Program Manual*. Dec. 2006. URL: `http://www.eac.gov/voting%20systems/docs/testingandcertmanual.pdf/attachment_download/file`, Chs. 3 & 4 does not discuss how to handle changes in standards during the course of testing a voting system.

[11] Ibid., § 7.

[12] It seems that few, if any, state statutes that require use of NASED-certified or EAC-certified voting systems require that those systems be certified to the *latest* national standards or guidelines. Therefore, most election jurisdictions that face such requirements would seem to have the option of purchasing older systems. *See:* U.S. Election Assistance Commission. *State Requirements and the Federal Voting System Testing and Certification Program*. 2009

[13] Jurisdictions might be able to manage this uncertainty by purchasing systems certified to the VVSG 2005 and specifying in the purchase contract that the manufacturer is obligated to update the system to conform to the VVSG 1.1. Our experience, based on a review of hundreds of manufacturer-jurisdiction contracts, is that such contract terms are relatively rare; it is far more common for contracts to specify that jurisdictions must pay to have their systems brought into conformance with new standards.

[14] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, vi.

[15] On a related historical note, the EAC proposes to strike much of the description of the history of the voting system testing and certification process. *See* ibid., §§ 1.1-1.2. This historical content is quite helpful for understanding the current federal certification system, and we suggest leaving this material in VVSG 1.1.

and thus make the immediate effect of VVSG 1.1 realistic. As later sections of our comment suggest, this assumption might not be correct. Changes in the requirements for voting system cryptography and software workmanship, for example, could require significant changes to current voting system design and architecture.[16]

3 The Importance of Auditability and Structured Data

Most voting systems fielded today have not been designed for auditability and evidentiary needs—i.e., robust support for maintaining the integrity of event logs, vote data and ballot configuration data. In this section, we argue that it is important for voting systems to keep data that support of various notions of procedural and forensic auditing, and to do so in standardized formats and methods.

3.1 Progress in Supporting Auditability and Structured Data Formats

The VVSG v1.1 makes a number of significant changes to support more robust auditing. The VVSG v1.1 now recognizes[17] that electronic records must support the 22-month archival storage requirements from federal law,[18] recently applied to electronic records in litigation in California.[19] Changes to the standards have also been made to increase the fidelity and integrity of audit logs and event logs.[20]

In terms of electronic reports of ballot records or aggregated ballot data, the new § 2.4.4 requires voting systems to have the capability to export these records in a:

> non-restrictive, publicly-available format. Manufacturers shall provide a specification describing how they have implemented the format with respect to the manufacturer's specific voting devices and data, including such items as descriptions of elements, attributes, constraints, extensions, definitions for data fields and schemas.[21]

Subsequent sections go on to detail what these reports must contain, the quality of randomization of ballot records and other polling place-related metadata.[22] For systems that include VVPAT, the guidelines now include a requirement that ballot image records be exportable in a "a publicly documented format, such as XML".[23]

3.2 Why Standardized Structured Data Is Important

Standardized, documented, structured, open data formats could significantly increase voting system competition and support for auditability, forensics and results reporting. Each of these four elements are important:

- *Structured:* Data formats need to be structured through the use of technologies like XML, in which the data is accompanied by a machine-readable document definition that can be used to validate data. This allows enforcement of complex type systems in an unambiguous manner. A data user with XML data and the document definition can "transform" the data into any format they wish.

[16] *See* §§ 4 and 5, below.

[17] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, §§ 2.1.10, 4.1.3.2, 4.1.6.1.b, 4.1.6.2.c, 4.1.7.1 and 5.3. (Volume II requires documentation of support for these requirements. (Ibid., Volume II § 2.8.5.i.))

[18] 42 U.S.C. § 1974.

[19] Americans for Safe Access, *et al.* vs. County of Alameda, *et al.*, 174 Cal. App. 4th 1287 (May 22, 2009).

[20] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 2.4.4.

[21] Ibid., § 2.4.4.1.

[22] Ibid., §§ 2.4.4.2, 2.4.4.3.

[23] Ibid., § 7.9.3.b.

- *Documented:* Public documentation of all elements of the data format are crucial so that data creators and users can share a common understanding of the semantic and technical meaning of each data element. Data formats like PDF and DOC do not meet this element of this definition because they are either not formally documented or contain proprietary elements that are undocumented and unavailable to most users.

- *Open:* Data formats must be open; they must publicly available for royalty-free uses unencumbered from copyright or patent claims by contributors to the creation of the format.

- *Standardized:* Finally, all this can be achieved without a deliberative, peer review process. However, standards created in openly deliberative forums allow all potential stakeholders input into the direction of the standard, and short circuit efforts by special interests to control their evolution.

The only standard for election data that meets these elements is OASIS' Election Markup Language (EML).[24] Other election-related data interchange standards are either proprietary—*i.e.*, Hart InterCivic's EDX[25]—or have long since stalled and never reached approved standard status—*i.e.*, IEEE 1622.[26]

The VVSG v1.1 could go further to support competition between voting system vendors, increased auditability of voting systems, evidentiary requirements and results reporting to the public and the media. In terms of competition, requiring that ballot definition files be standardized into a particular format would allow, for example, makers of voter registration products to export ballot definitions and other election-specific configuration data such that a jurisdictional customer could use an improved voter registration product, or simply the voter registration product that their state-level election entity has chosen or developed to implement HAVA's requirements for statewide voter registration databases.[27] Recently patents have been awarded to manufacturers on crucial election-related technologies, such as ballot marking devices (BMDs), that, when combined with obscure data interchange formats, ensures that customers that wish to use only the BMD product have to purchase the vendor's entire election suite. A well-documented data interchange format could allow a voting services company to design an EMS product that serviced a variety of vendor's voting system components.

Recently, interest has increased remarkably in the topic of voting system auditability. Post-election machine and manual tally auditing, particularly, has received a great deal of attention. In terms of auditability, requiring that manufacturers support standardized, structured, documented, and open data formats shifts the focus of these records from proprietary uses contemplated by the manufacturer, to a customer-centered focus. Such data formats will enable election officials, not just the vendor, to perform audits. For example, in a recent study implementing state-of-the-art "risk-limiting" post-election audit models in California, the most substantial hurdle in performing the work was not complicated mathematics and statistics, but simply getting election results in a machine-readable format to perform the necessary calculations.[28] When these researchers attempted to use electronic results reports from the voting system EMSs from three major vendors, the default format was PDF, a format designed for presentation of data, not calculation. EMSs that did allow export in formats suitable for computation,

[24] Election and Voter Services Technical Committee. *Election Markup Language (EML)*. Organization for the Advancement of Structured Information Standards. URL: `http://xml.coverpages.org/eml.html`.

[25] Hart InterCivic, Inc. *EDX Specification 2.27.1*. 2005. URL: `http://www.hartintercivic.com/files/edx/edxdoc/index.htm`.

[26] IEEE. *Voting Systems Electronic Data Interchange—Project 1622*. 2005. URL: `http://grouper.ieee.org/groups/scc38/1622/index.htm`.

[27] 42 U.S.C. § 15483 *et seq.*

[28] Joseph Lorenzo Hall, Luke W. Miratrix, Philip B. Stark, Melvin Briones, Elaine Ginnold, Freddie Oakley, Martin Peaden, Gail Pellerin, Tom Stanionis, and Tricia Webber. "Implementing Risk-Limiting Post-Election Audits in California". *Electronic Voting Techology Workshop/Workshop on Trustworthy Elections 2009 (EVT/WOTE 2009)* (Aug. 2009). URL: `http://www.usenix.org/events/evtwote09/tech/full_papers/hall.pdf`.

such as comma-separated value (CSV) and Microsoft Excel (XLS) formats, did not describe the formats and often contained significant problems.[29] While the systems used in this study were likely certified under to the 2002 VSS, we would expect that the requirements in the VVSG v1.1 should make substantial progress towards fixing problems like these. There a number of enhancements to the VVSG v1.1 electronic results report requirements that could assist contest-specific auditing efforts.[30]

Increasingly, types of forensic election data are being consulted as evidence in investigations and election contests. Researchers have pointed out that voting systems do not record sufficient forensic data.[31] They have determined what additional data should be recorded as well as how these data can be better captured and stored to assist investigators in determining which theoretical causes and effects are the actual causes and effects of anomalies of interest. A standard model of what data needs to be collected for forensics uses, as well as an open standardized grammar for recording it would mean that auditors and investigators could concentrate on the problems at hand, instead of idiosyncrasies and deficiencies of the voting system's infrastructure.

We have seen recent developments in the use of structured results reporting to better serve the information consumption needs of the media and the public. California uses the OASIS standard Election Markup Language (EML) to report results in real-time on election day and immediately afterward.[32] This has a number of benefits, in that media outlets can design their own tools to transform or adapt this data into formats and analyses suitable for engaging the public in election news. Unfortunately, California spends a good deal of time hand-coding results from counties that they receive as PDF reports and even faxes from counties to get this data into the EML format. Requiring voting systems to export results in publicly-documented structured data formats, as the VVSG v1.1 now does, will allow states like California to simplify this process, at most having to write a transform for each vendor or county that will cast the results reported from the EMS into the format that they need for aggregation at the state level and reporting.

3.3 The VVSG v1.1 Should Require Voting Systems Support EML

All this being said, it seems appropriate that the VVSG v1.1 should require voting systems to support specific standardized data interchange elements. The VVSG v1.1 should mandate support of election-specific *input*, *forensic records* and *output* from voting systems in a specific standardized, documented, structured open format.

What is needed here is not simply a publicly-available and specified format that the VVSG v1.1 calls for, but a specific standardized grammar that voting systems must "speak".[33] At this point in time, this would likely mean that voting systems must support the following in EML v5 or later:

- *election-specific voting system input* in the form of ballot definitions;

[29] For example, where columns or rows appear to have been swapped and aggregate numbers did not match values summed from individual values.

[30] While an extensive list is beyond the scope of this document, we will list a few: results need to be reported at the contest level for each unit subject to audit (which can be machine, precinct, polling place, etc. depending on state law). Also, for absentee and vote-by-mail balloting, "batches" of ballots are scanned in and this "batch" is the natural unit for auditing; results reports should report all contests present in a batch with their associated aggregate results. Finally, provisional ballots need to be reported separately in these reports so they can be disaggregated from the other results.

[31] Matt Bishop, Sean Peisert, Candice Hoke, Mark Graff, and David Jefferson. "E-Voting and Forensics: Prying Open the Black Box". *USENIX/ACCURATE/IAVoSS Electronic Voting Technology Workshop/Workshop on Trustworthy Elections 2009 (EVT/WOTE' 09)* (Aug. 2009). URL: `http://www.usenix.org/events/evtwote09/tech/full_papers/bishop.pdf`, 5.

[32] California Secretary of State. *Election Night Data-Feed Information*. URL: `http://www.sos.ca.gov/media/`.

[33] Bishop makes this case for audit logs, although his argument translates easily to election-specific input and output. *See:* Matt Bishop. "A Standard Audit Trail Format". *Proceedings of the Eighteenth National Information Systems Security Conference*, 136–145 (Oct. 1995). URL: `http://nob.cs.ucdavis.edu/~bishop/papers/1995-nissc/stdaudfmt.pdf`.

- *forensic records* in the form of event logs; and,
- *election-specific voting system output* in the form of tabulator and EMS results reporting.

To be sure, the draft VVSG II contained explicit recommendations that "encouraged [manufacturers] to use consensus-based, publicly available formats such as the OASIS Election Markup Language (EML) standard [OASIS07] or those emanating from the IEEE Voting System Electronic Data Interchange Project 1622 [P1622]."[34] Unfortunately, both the IEEE 1622 and IEEE 1583 voting system standards efforts have been defunct since 2005, having never adopted a final version of their respective standards, making their use or recommendation in the VVSG v1.1 inappropriate.[35]

4 Changes in Cryptography Requirements Are Significant Improvements but Do Not Address System-Level Issues

The draft Guidelines make important changes to how voting systems must use cryptography to support the security of several critical voting system components and functions, including:

1. signing electronic ballot images,[36]
2. signing voting system electronic reports[37] and voting system software (including updates) to be deposited in the National Software Reference Library,[38] and
3. signing transmissions of election information.[39]

These changes are generally positive from the standpoint of remedying some of the problems seen in how existing voting systems have implemented cryptographic modules. They take advantage of NIST's excellent work in defining requirements for cryptographic modules.[40]

From a system-level security perspective, however, the effects of the changes in the VVSG are likely to be limited. Cryptographic modules are but one voting system component that may be used to protect the integrity of election results; but they are embedded in and rely on software (such as commercially available operating systems) that may not be trustworthy and may permit attacks that entirely bypass the protections that cryptography offers.

4.1 FIPS 140-2 Provides a Solid Foundation for Implementing Cryptography in Voting Systems

Requiring FIPS 140-2-compliant cryptographic modules will likely remedy some of the worst abuses of cryptography seen in past systems. For example, one manufacturer's DRE used a cyclic redundancy

[34] U.S. Election Assistance Commission, Technical Guidelines Development Committee, *VVSG II*, see n. 3, § 1.1.5.

[35] *See* discussion of the use of IEEE 1583 draft standards elements in VVSG v1.1 in 5.1.

[36] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 7.9.3.

[37] Ibid., §§ 2.4.4.1-2.4.4.3.

[38] Ibid., § 7.4.6.

[39] Ibid., § 7.7.3.

[40] National Institute of Standards and Technology. *Federal Information Processing Standards (FIPS) Publication 140-2: Security Requirements for Cryptographic Modules*, 5 defines "cryptographic module" to mean "the set of hardware, software, and/or firmware that implements Approved security functions (including cryptographic algorithms and key generation) and is contained within the cryptographic boundary."

check (CRC) to authenticate data, an algorithm that is ineffective against malicious attacks.[41] Modules that comply with FIPS 140-2 would not have allowed such a choice.

Of course, if a cryptographic module that uses insecure algorithms is certified as FIPS 140-2-compliant, it is possible that a voting system manufacturer would use it in a voting system. VVSG v1.1 contains one important requirement to reduce the chance of this happening: Volume 2, § 2.6.7 states that manufacturers must list in the Technical Data Package (TDP) the algorithms the system supports, as well as provide identifying information and proof of NIST certification for each cryptographic module use in the voting system. This information would give VSTLs an opportunity to easily identify weaknesses in voting systems' uses of cryptography at an early stage.[42] VVSG v1.1 should clarify that VSTLs should consider the use of a FIPS 140-2-compliant cryptographic module necessary but not sufficient for conformance with the revised guidelines.

Another advantage of requiring manufacturers to use FIPS 140-2-compliant cryptographic modules is that compliant implementations are evaluated against a wide variety of criteria designed to ensure sound design and resistance to attacks.[43] As the draft Guidelines take into account throughout the document,[44] embracing the FIPS 140-2 framework requires a shift from specifying key lengths and ciphers to specifying "security strength"—the amount of work it takes to break the algorithm. As the NIST has noted in its recommendations for cryptographic key management, the overall strength of a cryptographic module depends on factors aside from key length, such as the algorithm and the environment in which the device using the module is used.[45] By requiring cryptographic modules that have demonstrated a minimum security strength, the draft Guidelines will likely require manufacturers to use technologies that have demonstrated their fitness along a number of dimensions, not just key length.

4.2 Changes in Cryptography Specifications do not Address Systemic Issues

Changes in the ways the VVSG would require voting systems to implement cryptography will only go so far in improving voting system security. This subsection focuses on broader, system-level issues that VVSG v1.1 would not address.

First, the draft Guidelines do not set requirements for implementing certain key elements of the cryptosystem. The algorithms that VVSG v1.1 focuses on are unlikely to be the weakest link in a voting system's security; these algorithms were developed through a rigorous, public process of peer review and have been the objects of continued study. FIPS 140-2 leaves other steps in implementing a cryptosystem, such as including key generation and key management, open to considerably greater discretion by the implementer.[46] For example, the requirement for key storage in FIPS 140-2 permits storing keys in plain text or encrypted form, leaving the specifics of storage to the implementation.[47]

In addition, hardware and software components lying outside the cryptographic module are potential sources of voting system insecurity cryptography cannot address, not matter how well it is implemented.

[41] Matt Blaze, Arel Cordero, Sophie Engle, Chris Karlof, Naveen Sastry, Micah Sherr, Till Stegers, and Ka-Ping Yee. *Source Code Review of the Sequoia Voting System*. July 2007. URL: `http://www.sos.ca.gov/elections/voting_systems/ttbr/sequoia-source-public-jul26.pdf`, 3.2.2.

[42] Details about the cryptographic modules used in the voting system would also give other reviewers, such as those who conducted reviews in California, Florida, and Ohio, access to this vital information. Since the EAC does not make TDPs public as a matter of course, however, only VSTLs are likely to have access to TDPs before the EAC decides whether to certify a voting system.

[43] National Institute of Standards and Technology, *FIPS 140-2*, see n. 40, § 3.

[44] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, §§ 2.4.4.1, 2.4.4.2, 7.4.5.1, 7.4.6, 7.5.1, 7.6.1, 7.7.3, 7.8.2, 7.9.3.

[45] National Institute of Standards and Technology. *Special Publication 800-57: Recommendation for Key Management—Part 1: General*. Mar. 2007, §§ 5.3.1, 5.6.1.

[46] National Institute of Standards and Technology, *FIPS 140-2*, see n. 40.

[47] Ibid., § 4.7.5.

Operating systems and other software programs are potential sources of vulnerabilities that would allow attackers to alter ballot definitions or cast ballots, outside of the cryptographic module. In addition, as recent research has demonstrated, hardware-based attacks are practical and potentially undetectable.[48]

The second issue relating to the limitations of the draft Guidelines' treatment of cryptography is that the Guidelines do not provide a framework within which manufacturers can govern decisions about how to implement cryptography. This is a for good reason; the judgments that are necessary may be specific to a manufacturer's technology and organization. They may be subtle, involving, for example, trade-offs between security strength, performance, and cost. Generally applicable standards are unlikely to be of much use here. It is important to recognize, however, that if voting system manufacturers lack internal processes to review cryptosystem implementation, then the prospects for cryptographic modules—no matter how thoroughly tested and certified—to improve voting system trustworthiness are bleak.

5 Changes to Software Security do not Obviate Software Independence

In addition to the changes to cryptography requirements discussed above, requirements related to software security have been modified in two major ways in VVSG v1.1: there are new software workmanship standards in § 5.2 and software validation requirements in § 7.4.6. We are glad to see an increased focus by NIST and EAC on elements of software design and development requirements. ACCURATE Principal Investigators, researchers and students have participated in a number of voting system evaluations in California, Ohio, and Florida and each of these reviews found serious, exploitable vulnerabilities due to simple programming mistakes that elements of the new workmanship requirements would alleviate.[49]

Software installation is an especially vulnerable process, and viral propagation of malicious code on voting systems has been recognized as a genuine threat since the last time the VVSG was updated.[50] Currently fielded systems were not designed with "software substitution" attacks—where an attacker replaces the certified software intended for a system—within the scope of the threats against which they claim to protect. The result is that current systems have features intended to make software installation trivially easy for anyone who enjoys physical access to the machines. Unfortunately, with only modest physical access to voting systems, attacks that exploit software installation and data input vulnerabilities can be subtle, elusive and leave no forensic trace of ever having been resident on the system.[51] As we explain below, we are not confident that the new software validation requirements of the VVSG v1.1 will significantly mitigate this dangerous class of threats.

[48] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten. "Security Analysis of the Diebold AccuVote-TS Voting Machine". In: *Proceedings of USENIX/ACCURATE Electronic Voting Technology Workshop*. 2007. URL: `http://www.usenix.org/events/evt07/tech/full_papers/feldman/feldman.pdf`.

[49] California Secretary of State. *Top-To-Bottom Review of California's Voting Systems*. Mar. 2007. URL: `http://www.sos.ca.gov/elections/elections_vsr.htm`; Patrick McDaniel, Matt Blaze, Giovanni Vigna, et al. *EVEREST: Evaluation and Validation of Election-Related Equipment, Standards and Testing (Academic Final Report)*. Ohio Secretary of State. Dec. 2007. URL: `http://www.sos.state.oh.us/sos/info/EVEREST/14-AcademicFinalEVERESTReport.pdf`; Ryan Gardner, Alec Yasinsac, Matt Bishop, Tadayoshi Kohno, Zachary Hartley, John Kerski, David Gainey, Ryan Walega, Evan Hollander, and Michael Gerke. *Software Review and Security Analysis of the Diebold Voting Machine Software*. July 2007. URL: `http://election.dos.state.fl.us/pdf/SAITreport.pdf`.

[50] J. Alex Halderman, Eric Rescorla, Hovav Shacham, and David Wagner. "You Go to Elections with the Voting System You Have: Stop-Gap Mitigations for Deployed Voting Systems". *USENIX/ACCURATE Electronic Voting Technology Workshop* (Aug. 2008). URL: `http://www.usenix.org/event/evt08/tech/full_papers/halderman/halderman.pdf`.

[51] Stephen Checkoway, Ariel J. Feldman, Brian Kantor, J. Alex Halderman, Edward W. Felten, and Hovav Shacham. "Can DREs Provide Long-Lasting Security? The Case of Return-Oriented Programming and the AVC Advantage". *USENIX/ACCURATE/IAVoSS Electronic Voting Technology Workshop/Workshop on Trustworthy Elections 2009 (EVT/WOTE' 09)* (Aug. 2009). URL: `http://www.usenix.org/events/evtwote09/tech/full_papers/checkoway.pdf`.

The reality is that these two new sections, despite their intent, are not up to the task of eliminating software flaws and vulnerabilities which could affect the outcome of an election. Software independence,[52] a prominent feature of the draft VVSG II, is the only requirement we know of that would *ensure* these problems are unlikely to result in an uncertain election. In this section of our comments, we describe how the complexity and incompleteness of these two new sections of the VVSG v1.1 are potentially problematic, and cannot obviate the need for software independence.

As we mentioned earlier in this document, there is an open question as to whether these changes, as they will be extensive for some manufacturers who use older languages and support large legacy code bases, are so extensive that they will draw resistance from voting system manufacturers and testing laboratories. These changes should probably be implemented with a greater lead time[53] rather than becoming effectively immediately upon adoption by the EAC.

5.1 Software Development and Workmanship Requirements

5.1.1 Software Coding Requirements

In addition to easily exploitable software installation mechanisms (*see* § 5.2) and architectural elements based on faulty trust models, perhaps the most significant source of vulnerability in currently fielded voting systems is the prevalence of software defects. These vulnerabilities include buffer overflows, SQL (database query language) injection, and string format vulnerabilities, each of which in their worst form can permit an attacker to run malicious code. Increasingly sophisticated attacks, such as return-oriented programming[54]—where an attacker uses small pieces of the legitimate software to construct their own malicious code—can only be avoided by system developers working diligently to minimize the number and extent of these kinds of flaws coupled with architectures and tools—like type-safe languages—that make these errors less serious. Voting system reviews such as the California Secretary of State's Top-To-Bottom Review, the Ohio Secretary of State's EVEREST Review and the various reviews conducted on behalf of the Florida Secretary of State found multiple instances of these types of flaws in each system examined.[55]

We are encouraged by the VVSG v1.1's new Section 5.2, "Software Design and Coding Standards" which begins to address fundamental problems in software workmanship in voting systems, with respect to software defects but also in terms of other measures of software quality. This section restricts the selection of programming languages used in voting systems to those that support a specific list of features which together enhance workmanship, security, integrity, testability and maintainability.[56] Further, software source code development must adhere to a credible, published set of coding guidelines that have gained some acceptance from other non-voting manufacturers.[57] Similarly, code must be compiled with a COTS compiler and/or interpreter, which prohibits manufacturers from inventing new programming

[52] Ronald L. Rivest and John Wack. *On the Notion of "Software Independence" in Voting Systems*. National Institute of Standards and Technology HAVA Technical Guidelines Development Committee. July 2006. URL: `http://vote.nist.gov/SI-in-voting.pdf`.

[53] *E.g.*, the 24 month waiting period for previous VVSG iterations, see Section 2.2.

[54] Checkoway, Feldman, Kantor, Halderman, Felten, and Shacham, see n. 51.

[55] California Secretary of State, *California TTBR*, see n. 49; McDaniel, Blaze, Vigna, et al., see n. 49; Gardner, Yasinsac, Bishop, Kohno, Hartley, Kerski, Gainey, Walega, Hollander, and Gerke, see n. 49; Florida State University's Security and Assurance in Information Technology Laboratory. *Software Reviews and Security Analyses of Florida Voting Systems*. Feb. 2008. URL: `http://www.sait.fsu.edu/research/evoting/index.shtml`.

[56] Programming languages, such as C, which do not support a specific feature, such as exception handling for C, must be used with COTS extensions that add this functionality. U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 5.2.2

[57] Ibid., § 5.2.3.

languages.[58]

As is a theme throughout this comment, a requirement for software independence would reduce the need for such detailed attention to these issues in the VVSG v1.1.

5.1.2 Mandate the Use of Type-safe Languages or Require Static Analysis

VVSG v1.1 § 5.2.8 establishes detailed requirements for error checking. These requirements will tend to minimize the occurrence of the type of software errors we discuss at the beginning of this section, but they also go further to specify that programmers need to be careful about other common sources of error, such as memory misallocation, pointer errors, and CPU-level exceptions.

However, mandating the use of "type-safe" programming languages would accomplish much of the detailed requirements in this section. A "type-safe" programming-language automatically checks for the exact kinds of problems enumerated by § 5.2.8 by classifying the syntax of source code and making sure that the values they compute do not result in these kind of errors.[59] Of course, mandating the use of a type-safe language would further restrict the design decisions of voting system developers. The trade-off here is that the various error checking requirements in § 5.2.8 are testing and certification-based substitutes for software properties that should be properly restricted by the technology.

Accordingly, we believe that voting systems should be required to use type-safe programming languages. Voting systems that do not use type-safe programming languages should be required to *demonstrate* that they've made a substantial effort to eliminate the types of § 5.2.8 errors in in their code. To accomplish this, the VVSG v1.1 should be modified to require manufacturers that utilize type-unsafe languages do the following:

1. They must use a widely-available COTS static source code analysis tool[60] to examine their software for the kinds of errors listed in § 5.2.8 and work to eliminate these issues;

2. They must submit a report as part of the TDP that shows what static tool they used, what warnings the tool finds in the code submitted for testing, an explanation for each unresolved warning or issue (*e.g.*, false positives) and a description of the exact configuration settings of the static tool so that the VSTL could conceivably re-run the same tool on their submitted code to verify no other § 5.2.8 issues are present; and,

3. The VSTL should be required to test the code with their own static tool, which does not have to be necessarily the same tool that the manufacturer used.[61] The VSTL should challenge any issues that it feels violate portions of § 5.2.8 and require the manufacturer to cure those defects.

This two-pronged structure of requiring either the use of a type-safe language or requiring manufacturers to demonstrate effective type-safety through the use of static analysis tools would be an unqualified leap forward in the security and reliability of election systems.

5.1.3 Executable Code and Data Integrity

The requirements placed on executable code and data integrity in § 5.2.7 of the draft VVSG v1.1 are a step in the right direction. Requirements in this section that require using COTS compilers and inter-

[58] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 5.2.7.iii–iv.

[59] Benjamin C. Pierce. *Types and Programming Languages*. 1st ed. The MIT Press, 2002. ISBN: 0262162091.

[60] Static analysis tools, such as Fortify, Coverity and Valgrind, are automated source code analysis tools that look for common errors and sources of vulnerability.

[61] If the VSTL finds anything covered by § 5.2.8 that the manufacturer's did not, the manufacturer will necessarily have to fix it and amend their test package.

preters,[62] prohibit self-modifying code and modification of executable code,[63] configuration data, vote data and audit records[64] all work towards the goal of ensuring that the code running on a voting system in the field is the same code that was tested and certified for use.

We believe it is important that any new code introduced into the system must have been tested and certified and must go through the VVSG v1.1's authenticated, signed software installation process.[65] Accordingly, the language in § 5.2.7.b emphasized below is unacceptable:

> All programmed devices shall prevent replacement or modification of executable or interpreted code [. . .] *except where this access is necessary to conduct the voting process*.

Code should not be in any way executable that was not part of the package submitted for certification and subject to VSTL testing. Executable code resident on a voting system should only be replaceable with code that was also certified. Memory cards, network protocols, and other data transmission paths should never have executable code moving back and forth during election processes unrelated to the installation of new software.

The guidelines state application logic should be free of "race conditions, deadlocks, livelocks, and resource starvation."[66] As is the case in a number of areas of this part of the guidelines, there must be some mechanism to demonstrate, to the greatest extent possible, that this property is true, either provided by the programming language and development toolchain, or by some third-party verifier that the manufacturer and VSTL use to confirm these properties.

We are also concerned that the new sections on executable code and data integrity (§ 5.2.7) and error checking (§ 5.2.8), despite their value in describing needed safeguards to prevent common vulnerabilities, are entirely imported from the defunct IEEE 1583 voting system standards effort. These draft standards are not publicly available, and neither is the deliberation around which they were developed, potentially masking flaws in this language from the public.[67] As the footnotes in those two sections make clear, this content is not recommended for use in conformance or compliance processes. There are other sources that have achieved wide acceptance for similar types of guidelines[68] and we urge the EAC and NIST to incorporate those into a revised VVSG v1.1, rather than rely on an unapproved draft standard.

5.1.4 Modular Software Testing

VVSG v1.1 § 5.2.4 sets requirements on modular design of code and an upper limit on the complexity of software modules. However, we believe this doesn't go far enough. Specifically, there should be a requirement that voting system source code is provided to the VSTLs with test suites—software written by the system developers to test each part of the code. Volume II of VVSG v1.1 requires manufacturers to "describe security tests performed to identify vulnerabilities and the results of the testing. This also includes testing performed as part of software development, such as unit, module, and subsystem testing."[69]

A crucial piece of such a test suite are "unit tests"—specific pieces of testing-related source code that exercise each piece of code. Standard Quality Assurance methodologies for software development

[62] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 5.2.7.a.iii–iv.

[63] Ibid., §§ 5.2.7.a.i and 5.2.7.b.

[64] Ibid., § 5.2.7.c.

[65] ibid., § 7.4.6. See discussion below in Section 5.2.

[66] Ibid., § 5.2.7.a.ii.

[67] The page for the IEEE 1583 draft standard states prominently, "Need password to get draft standard.", *see:* `http://grouper.ieee.org/groups/scc38/1583/p1583_-_stds.htm`.

[68] *E.g.*, the coding standards listed in U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, §5.2.3.

[69] Ibid., Volume II, § 2.6, 37.

involve unit tests where a software-based test suite developed in line with the functional code can be run in order to ensure that the functional and design requirements are being met.[70] These tests should be required and required to be included in the source code package delivered to the VSTL for testing and certification.

In a similar vein, manufacturers should provide regression tests. Software regression tests are similar to unit tests but test the code to make sure that known vulnerabilities that have been problematic in the past are not re-introduced at a later stage of development. Placing the burden on the VSTL to maintain and author these tests seems inappropriate and burdensome.

Finally, mission critical software testing, such as that performed on software that supports avionics equipment, also includes "code coverage" requirements where the test suite must execute a certain percentage of all lines of source code.[71] A requirement that the manufacturer's software test suite meet a coverage of 80% of all non-comment lines of code seems wise.

5.2 Software Validation Requirements

The VVSG v1.1 includes a new section entitled "Software Setup Validation" that appears to require voting systems to provide one of two methods for verifying the software resident on the device.[72] Both methods require checking the digital signature or cryptographic hash of the software against a reference copy of the software (or its reference signature or hash). Manufacturers are free to design their system such that either this validation process occurs *before* the software is installed or can occur *after* installation through the use of a validation port ("external interface") and COTS validation equipment. Considering the ease with which software is replaced on some current models of fielded voting systems, the concern that systems should better validate software for installation is well-placed.

However, there are significant issues in this part of the VVSG v1.1. The requirements are vague, and it is possible to read into it three distinct options for software validation; we outline this interpretation in subsections 5.2.1–5.2.4. After that discussion, we propose enhancements to the software validation framework in subsections 5.2.5–5.2.8.

5.2.1 The Software Validation Requirements are Vague

The introduction to § 7.4.6 in the draft VVSG v1.1 states that section "includes requirements for two software verification techniques." However, there appear to be three distinct types of software verification itemized in the requirements that follow:

1. *Software Setup Validation:* § 7.4.6.a–c itemize requirements for setup validation including, respectively, that this mechanism should verify that no unauthorized software is present, that lists all installed files and that ensure there have been no illegitimate modifications of the voting system software.[73]

2. *Pre-installation Software Validation:* § 7.4.6.e details requirements for validation methods that verify software prior to installation including a restricted installation path, authenticated installation, the form, signing and contents of software update packages and requirements for installation-relevant event logging.

[70] Paul Hamill. *Unit Test Frameworks*. O'Reilly Media, 2004. ISBN: 978-0-596-00689-1. URL: `http://oreilly.com/catalog/9780596006891/`.

[71] Radio Technical Commission for Aeronautics. *Software Considerations in Airborne Systems and Equipment Certificat ion (DO-178B)*. Dec. 1992.

[72] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 7.4.6.

[73] § 7.4.6.g requires setup validation methods to "verify that registers and variables of the voting system equipment contain the proper static and initial values."

3. *Post-installation Software Validation:* § 7.4.6.f covers validation methods that use an external interface through which an administrator can verify the voting system software, including properties of the interface and COTS audit tool.

In the following discussion, we will refer to these as methods 1, 2 and 3. § 7.4.6.d explicitly requires using either method 2 or 3 above. It is difficult to tell from the text if method 1, the "setup validation method", is a distinct type of validation method or an umbrella term under which either method 2 or 3 sits. It appears so, in that the requirements pertaining to setup validation appear to be required to trigger each time the voting system is started where as either pre-installation or post-installation software verification occurs at distinct times dictated by the installation of software—for method 2, or, presumably, during a software verification audit—for method 3.

There are significant issues with each of these methods. In the remainder of this section we will first detail limitations of each of the above validation methods and then cover a number of associated issues that are crucial to get right for validation to work well.

5.2.2 Validation Method 1: Setup Validation is Unprecedented and Rare in Other Applications

It is difficult to tell from the text of the draft VVSG v1.1 precisely, but it appears that setup validation is intended to function such that when a voting system is powered on, it checks that no unauthorized software is present,[74] that no software has been modified illegitimately,[75] that all registers and variables have been initialized properly[76] and lists the names, version numbers and installation dates of all files on the system.[77]

If our interpretation is correct, this would require that voting systems keep either hardware-based or software-based cryptographic control of the voting system's filesystem and executable memory pages such that no unauthorized software can be installed and that any modifications to installed software result in the voting system not completing its boot sequence. Such a design appears to incorporate parts of trusted computing and/or digital rights management technologies; that is, a hardware- or software-based cryptographic module that can sign and verify files on the voting system.[78] While trusted computing modules are now standard on many personal computers, they are currently only widely used for encrypted disk applications, and not signature-based system startup verification.[79] Digital asset and rights management technologies have seen wider use, but almost every such system has been subverted, including such systems used for digital video,[80] digital music,[81] video gaming[82] and controlling both

[74] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 7.4.6.a.

[75] Ibid., § 7.4.6.c.

[76] Ibid., § 7.4.6.g.

[77] Ibid., § 7.4.6.b.

[78] Sean W. Smith. *Trusted Computing Platforms: Design and Applications*. 1st ed. Springer, 2004. ISBN: 0387239162. URL: http://www.springer.com/computer/communications/book/978-0-387-23916-3.

[79] Trusted Computing Group. *Frequently Asked Questions*. http://www.trustedcomputinggroup.org/media_room/faqs. Sept. 2009. URL: http://www.trustedcomputinggroup.org/media_room/faqs, (details wide usage of trusted computing modules); Trusted Computing Group. *Press Release: Self-Encrypting Drives Based On New Trusted Computing Group Specifications Now Available*. http://www.trustedcomputinggroup.org/media_room/news/13. Mar. 2009. URL: http://www.trustedcomputinggroup.org/media_room/news/13.

[80] Barbara Simons. "From the president: to DVD or not to DVD". *Communications of the ACM* 43:5, 31–32 (2000). URL: http://doi.acm.org/10.1145/332833.332851, (explaining DVD decryption).

[81] Hymn Project. *Description of Hymn Project*. 2009. URL: http://hymn-project.org/, (description of project to decrypt Apple iTunes Music Store content).

[82] Andrew Huang. *Hacking the Xbox: An Introduction to Reverse Engineering*. 1st ed. No Starch Press, 2003. ISBN: 1593270291. URL: http://hackingthexbox.com/, (explaining techniques to subvert the Microsoft Xbox gaming platform).

network access and filesystem access on cell phones.[83] These techniques cannot *ensure* integrity, but can make subversion exceedingly difficult.[84]

There are a few narrow examples of academic prototypes[85] and narrow limited-function commercial systems that perform some of this functionality. As an example, we know of architectures used for gaming devices, lottery machines and point-of-sale credit card terminals that use custom-designed motherboards with a PGP-signed filesystem and with the manufacturer's public keys soldered into the board and rendered unremovable.[86] At run-time these systems continuously recheck the filesystems' PGP signature using the chip-based public key. Likewise, they continuously check the PGP signatures on "text" segments (memory pages that have executable code) to ensure that they are unmodified during execution. Any failed check of the filesystem or text causes an immediate shutdown of the operating system. These are exceedingly expensive techniques that are far from foolproof—almost every such system has been subverted—and require massive investments in design, implementation and testing.

Placing a requirement on voting systems to perform a task that has rarely before been successfully executed in other areas of application seems unwise.

5.2.3 Validation Method 2: Verifying Software is a Difficult Problem

The pre-installation software verification method of § 7.4.6.e requires only one method for installing, changing or removing software,[87] that only an authorized administrator can install software,[88] that software version information to be stored and displayed,[89] that signed and verified software update packages in a publicly-available standard form are used,[90] that software downgrades can be prevented,[91] and that the voting system log record specific information about the update.[92]

There are a few basic concerns with software verification of this sort. Software that performs the verification operation can be constructed to misreport the list of software on the device or can miscalculate the digital signatures.[93] In addition, designing usable forms of software validation have proved

[83] Andy Greenberg. "Unlocked iPhones For All". *Forbes* (Sept. 2007). URL: `http://www.forbes.com/2007/09/12/iphone-hacking-software-tech-wire-cx_ag_0912iphone.html`, (Apple's iPhone "unlocked"—hacked to operate on any carrier's GSM cellular network); Marin Perez. "iPhone Dev Team Releases Jailbreak Software". *InformationWeek* (July 2008). URL: `http://www.informationweek.com/news/personal_tech/iphone/showArticle.jhtml?articleID=209400178&subSection=All+Stories`, (Apple's iPhone "jailbroken"—hacked so that arbitrary code can be loaded onto the device).

[84] Almost every system that we know of which employ such architectures has been compromised. Once a product like this makes it into the hands of the public, it is only a matter of time and interest before hobbyists find methods to bypass these security measures and load arbitrary code. *See:* Dan Goodin. "Texas Instruments aims lawyers at calculator hackers". *The Register* (Sept. 2009). URL: `http://www.theregister.co.uk/2009/09/23/texas_instruments_calculator_hacking/`, (Hobbyists use a distributed computing platform to discover the "signing keys"—cryptographic keys used to sign the operating system—of Texas Instruments' graphing calculators).

[85] Arvind Seshadri, Mark Luk, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. "Externally verifiable code execution". *Communications of the ACM* 49:9, 45–49 (2006). URL: `http://doi.acm.org/10.1145/1151030.1151054`.

[86] Paco Hope, Cigital, Inc., *personal communication*.

[87] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 7.4.6.e.i.

[88] Ibid., § 7.4.6.e.ii.

[89] Ibid., § 7.4.6.e.iii.

[90] Ibid., § 7.4.6.e.iv–vii.

[91] Ibid., § 7.4.6.e.viii.

[92] Ibid., § 7.4.6.e.ix–x.

[93] A good example of this is the Sony CD DRM Rootkit, that would hide files that began with `$sys$` from the operating system. *See:* J. Alex Halderman and Edward W. Felten. "Lessons from the Sony CD DRM Episode". *15th USENIX Security Symposium* (Aug. 2006). URL: `https://www.usenix.org/events/sec06/tech/halderman.html`.

difficult.[94] For example, voting equipment that includes hundreds or thousands of files and/or resources that must be displayed with version information per the new requirements,[95] might not add significantly to security but might significantly impact usability.

Generally this seems like a good idea; it will help place barriers that viruses will have to overcome to spread onto voting machines. However, this method has serious limitations that should be explicitly recognized in the VVSG.

First, this method will need to be implemented by a combination of both hardware and software; hardware that provides the path for reading installation media and software that performs the validation and then installation. If either of the hardware or software designed for method 2 are themselves corrupt or flawed, then they will not be able to prevent the unauthorized installation of software on the system. Unfortunately, we can't eliminate these potential sources of vulnerability and validating the validation hardware and software creates an endless cycle of pushing trustworthiness onto other mechanisms.

Second, although § 7.4.6.e.i requires "no more than one method for installing... software on a system," it is not clear that this is technically feasible. Any reasonable hardware design for computerized voting machines will require a low-level hardware method to bypass the installation process of method 2. This low-level method would need to be used when the system is manufactured in order to install the software that implements method 2. It would also need to be used in case the pre-installation validation software becomes corrupted in some way, so that this software could be re-installed. If the voting system is manufactured without some mechanism for re-installing the installation software, there is a chance that the voting system could be rendered useless—"bricked", to use a technical term—by a software error or hardware fault.

Software assurance and validation is an ongoing field of active research in computer science and it is unclear if the VVSG v1.1's standard, as currently written, are feasible.

5.2.4 Validation Method 3: The Difficulty of External Interface Validation

In lieu or in addition to a software-based installation validation mechanism, the VVSG v1.1 also allows an external validation interface.[96]

There are some severe limitations to the effectiveness of this method. The following limitations are so severe, that we feel that the VVSG should remove this from the draft VVSG v1.1 entirely, or at least require method 2 above and not allow method 3 here to substitute for that function.

As we highlighted above in the discussion about method 2, the external interface, just like the installation interface, may consist of fraudulent or erroneous software and hardware.

Modern computing devices contain low-level software in hardware components that often cannot be read out by any type of external interface. Some of these hardware components contain software ("firmware") that is not known or available to the voting system manufacturer. However, corruption of these components can deliberately modify the way the voting system, for example, counts votes and records event log data.

An administrator auditing a voting system by means of this external interface must plug a computer into it. That computer then must interrogate the voting system through this interface to read out the (claimed) voting system software. It must then compare this data with reference information about the authorized software. If this interrogation device is corrupted, then it can erroneously report that the software is authorized, regardless of any mis-match in the comparison. Here, the problem of checking

[94] Ryan Gardner, Sujata Garera, and Aviel D. Rubin. "On the Difficulty of Validating Voting Machine Software with Software". *USENIX/ACCURATE Electronic Voting Technology Workshop 2007* (July 2007). URL: `http://www.usenix.org/events/evt07/tech/full_papers/gardner/gardner.pdf`.

[95] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 7.4.6.e.iii.2.

[96] Ibid., § 7.4.6.f.

that the voting system software is authorized has been transformed into a problem of determining that the interrogation device software is authorized.

In addition, because the validation happens *after* the installation of software has been performed, the opportunity to prevent malware from gaining a foothold on the system has passed. Unlike the case of pre-installation validation discussed in the last section, this method has numerous limitations and the fundamental problem that checking software *after* installation is too late to be effective. These concerns would seem to advise strongly against any form of external interface validation as some varieties of malware can be exceedingly pernicious and difficult to detect and remove.[97]

5.2.5 Who Should Sign Software Updates?

The relevant requirements for the software verification method state that the update package shall be signed by the National Software Reference Library (NSRL), the voting device owner *or* a designated notary repository (presumably, a binary software escrow company).[98] The "or" in this requirement is problematic.

We would like to see one repository authority sign all EAC-certified software updates; the NSRL would seem to be a natural choice. Most vendors that participate in federal certification already use the NSRL as a repository for binary software and verification metadata.[99] Standardizing the use of NSRL as a primary repository for voting software binaries makes sense for a number of reasons. The voting device owner—often state or local election officials—might not have the requisite expertise or capability to maintain chain of custody of the software update package from the vendor to their offices and then sign the packages without exposing them to an opportunity to replace or otherwise modify the software update packages. In addition there are benefits for centralizing this function at NSRL, instead of distributing it amongst a number of private escrow companies in terms of maintainability of standardized verification metadata and making this metadata available to voting system auditors and investigators, in perpetuity, regardless of the relationship between the manufacturer and escrow entity.

However, the section on external interface validation includes a similar but different requirement. External validation through an interface does not require checking the digital signatures of a software update package; there is no update package to be checked as the software has already been installed. The requirement here states that the "verification process shall either (a) use reference information on unalterable storage media received from the repository or (b) verify the digital signature of the reference information on any other media."[100] This is problematic as there is no requirement, unlike the one that exists for the software validation process, that the resulting updated software is signed by a repository such as the NSRL. Further, it is unclear here what "reference information" is being checked. We might assume, by context, that this reference information is that included under the software verification process requirements,[101] but this should be added explicitly to the proper section of the external validation interface requirements, or pulled out as an item applicable to both methods.

Finally, § 7.4.6.e.vi allows software update packages to be signed using any NIST approved algorithm with a security strength of 112 bits. This is one area where the VVSG v1.1 should pick a specific standard hash or digital signature algorithm. As one example, if a manufacturer uses an algorithm that is not publicly-available and/or proprietary, the digital signature will not be useful by the election official

[97] Halderman, Rescorla, Shacham, and Wagner, see n. 50; Checkoway, Feldman, Kantor, Halderman, Felten, and Shacham, see n. 51.

[98] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 7.4.6.e.vi.

[99] National Institute of Standards and Technology. *Voting Software Reference Data Set (September 3, 2009)*. National Software Reference Laboratory. Sept. 2009. URL: `http://www.nsrl.nist.gov/votedata.html`.

[100] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 7.4.6.f.2.

[101] Ibid., § 7.4.6.e.vii.

or election auditor unless they have access to this software. There should be no ambiguity here; the VVSG v1.1 should specify exactly which hash algorithm is used for signing and verifying updates.

5.2.6 Installation During Sensitive Times

This section of VVSG v1.1 correctly restricts validated software installation, via any method above, to times other than when the polls are open.[102] However, there are other sensitive times under which spurious software installation could be dangerous. For example, some voting systems have a "pre-election state" where the machine has been reset, new ballot definitions loaded, logic and accuracy testing performed and the machine set as ready for the "open polls" operation. In some cases, machines are placed in this pre-election state and placed in environments that are less physically secure than the election warehouse, such as sent home with poll workers or delivered for storage in polling places. The same danger presents itself for the "polls closed" state where the machine has been shut down, ballot records removed and placed back in temporary storage in the polling place to await pick up. In both these cases, software installation could be very dangerous, either leading to corrupt or misbehaving voting systems during the voting period or corrupting valuable forensic data.

At a minimum, the VVSG v1.1 should be modified such that installation is restricted in both pre-election and post-election states where the voting device might be stored in less-secure environments.

5.2.7 Only Authorized Installation

The VVSG v1.1 requires that only authenticated administrators are allowed to perform system upgrades.[103] We feel that the VVSG v1.1 should go into more depth about the characteristics of this kind of authentication. For example, in some older systems, installation required no authentication or at least access to the inner workings of the machine via a key lock that was very easily picked with common knowledge and tools. Considering the very sensitive nature of software installation, it is essential that the VVSG v1.1 provides more detail here in terms of strength of the authentication mechanism.

5.2.8 Preventing the Installation of Obsolete Software

The requirements for software verification also require that the mechanism must have the capability to prevent a previous version of the software being installed.[104] Presumably, if a subsequent downgrade in the software from a more recent version to an older one might implicate system security, this mechanism should be invoked to prevent the downgrade. However, there is no indication as to who will make this decision: the EAC, the VSTL and/or the vendor? It would seem that any update that implicates security concerns or patches potential security vulnerabilities should be required to invoke this mechanism to prevent software downgrades. VSTLs will have to take special care with these kinds of updates as an upgrade that adversely affected the system would be irreparable until a new upgrade were introduced, assuming the botched upgrade did not affect the installation mechanism.

6 New Requirements for Accuracy and Reliability Testing

The VVSG v1.1 significantly enhances the accuracy requirements, acceptable misfeed rate, and reliability requirements and benchmarks. The current draft of these requirements, however, lacks an explanation of how they were developed and why they are adequate.

[102] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, §§ 7.4.6.e.i.2, 7.4.6.f.i.3.
[103] Ibid., § 7.4.6.e.ii.
[104] Ibid., §7.4.6.e.viii.

The accuracy requirements are modified from being a certain number of ballot position errors to a "report total error rate" of no more than one in 125,000 (8×10^{-6}).[105] This standard is derived from the HAVA-mandated maximum of one error in every 500,000 ballot positions.[106] This change has the benefit of measuring sources of error other than ballot position errors.

The notable change on the reliability front is a revision of the acceptable multiple feed rate from 1 in 10,000 to 1 in 500.[107] This change is poorly explained in VVSG v1.1; a quick reading makes it appear that the VVSG 2005 standard has been relaxed by a factor of 20. This is not the case, as the new standard includes other types of errors—jammed and rejected ballots—in addition to multiple feeds. Whether including these other types of errors in a newly defined, total misfeed rate warrants such an increase in the maximum acceptable rate is unclear based on the contents of the draft standard. At minimum, the EAC should state the reasons for its implicit conclusion that paper jams and rejected ballots are much more common than multiple feeds. In addition, the EAC should consider requiring a lower acceptable error rate for central count optical scanners, since those devices are typically used in more controlled environments (lower humidity, etc.) and by more skilled personnel than precinct count optical scanners.

The reliability benchmarks and performance requirements are also substantially enhanced, going from a mean time between failure (MTBF) of 163 hours, to a nuanced taxonomy of failure types with associated criteria and benchmark values.[108] The more detailed standard is a welcome change; but, again, the draft standard does not contain an adequate explanation of how the maximum acceptable rates were derived.

Finally, we note that VVSG v1.1 omits a volume testing requirement—a valuable way of testing systems' reliability and accuracy.[109] Volume testing—where testers cast votes on many voting devices in simulated election conditions—was proposed in the Draft VVSG II as a testing method that would "contribute substantially to the evaluations of reliability, accuracy, and misfeed rate".[110] It simulates the load that a typical machine might encounter during its peak use period and does so on many devices at once, allowing testers to observe the ways voting systems fail and measure the consistency of certain types of failures across an inventory of exemplar machines. Volume testing has become an essential part of certification in California, where it has exposed subtle, hard-to-diagnose flaws with voting systems that would not have been detected otherwise.[111] It would be more efficient, however, to require volume testing at the national level. States would not duplicate the cost of conducting these tests, and manufacturers can fix flaws found during national certification and submit them as part of an ongoing certification process, rather submitting them as a modification to a certified system, which could prompt new certification testing. Because it permits an effective and significant measurement of reliability and because of its high expense, volume testing should be conducted at the national level instead of piecemeal at the state or jurisdictional level.

[105] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 4.1.1.

[106] *See* HAVA § 302(a)(5), 42 U.S.C. § 15482(a)(5); U.S. Election Assistance Commission. *2002 Voting System Standards*. 2002, §3.2.1.

[107] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 4.1.5.1.e.

[108] Ibid., § 4.3.3.

[109] Note that volume testing is not intended as a measure of vulnerability in the face of malicious adversaries. Adversarial vulnerability testing (OEVT) gets much closer to simulating a system's capabilities in that respect.

[110] U.S. Election Assistance Commission, Technical Guidelines Development Committee, *VVSG II*, see n. 3, Part 3:5.2.3-D.

[111] David Jefferson, Matt Bishop, Loretta Guarino Reid, and David Wagner. "Lessons from the Diebold TSx 'Sliding Finger' Bug". *Unpublished*. 2005.

7 The Need for Performance-based Usability Benchmarks and Testing

Usability and accessibility are crucial properties of voting systems. While we can cite a number of examples of lost votes due to technical voting system error, experts estimate that poor ballot design and instructions—themselves only one facet of voting system usability—result in ten thousand, perhaps as much as a hundred thousand, lost votes every year.[112]

The changes to the usability and accessibility requirements in the VVSG v1.1 are taken almost entirely from the draft VVSG II. Besides minor changes,[113] the biggest difference between the two in terms of usability and accessibility is the removal of the usability performance benchmarks and associated testing framework. Accordingly, the section of our 2008 comment on the substance of these standards still applies.[114]

This testing framework was championed by the Human Factors and Privacy (HFP) subcommittee of the TGDC in order to provide a novel type of conformance testing for usability that would be able to set limits on usability performance These benchmarks set limits on five areas of voting system usability and accessibility: general usability, usability for voters with no vision, usability for voters with low vision, usability for voters lacking dexterity or with only limited dexterity and poll worker usability. The TGDC deliberated and then adopted a resolution to develop such benchmarks and testing in 2005.[115]

NIST committed to an intense investment in time and resources towards developing the usability performance benchmarks and associated test protocol. This type of conformance testing for usability is unprecedented, broke new ground in the field of usability and raised some foundational questions about how to conduct this sort of test. NIST researchers worked through these challenges and have begun to report the output of the work on general usability and poll worker usability.[116] This is complex, detailed work that requires substantial validation and user-testing; it would be a grave disappointment if this effort were for naught and the benchmark work remained unused.

It is unclear if the testing protocol for usability benchmarking is now ready. If so, the requirement for such benchmark testing must be put back in the VVSG v1.1. If these testing protocols are not yet complete, we are disappointed that they were not ready for this iteration of the guidelines and request that the EAC and NIST issue a public report on the status of the development of these tests; we hope to see it adopted in the next round of the VVSG. If the general usability and poll worker tests are complete, as suggested by the public reports on this work cited above, we strongly urge that these two tests be

[112] Lawrence Norden, David Kimball, Whitney Quesenbery, and Margaret Chen. *Better Ballots*. Brennan Center for Justice at NYU School of Law. July 2008. URL: `http://www.brennancenter.org/page/-/Democracy/Better%20Ballots.pdf`.

[113] The color and contrast guidelines were enhanced. *See:* U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 3.2.5

[114] A Center for Correct, Usable, Reliable, Auditable and Transparent Elections, *Public Comment on VVSG II*, see n. 4.

[115] U.S. Election Assistance Commission, Technical Guidelines Development Committee. *Resolutions adopted by the TGDC*. May 2006. URL: `http://vote.nist.gov/tgdcresolutions0506.pdf`, Resolution 05-05, at 11.

[116] *See*, in general: U.S. Election Assistance Commission, Technical Guidelines Development Committee. *Document Map*. URL: `http://vote.nist.gov/docmap.htm` Specific output of the work towards usability benchmarks and test suites include: Dana Chisnell, Susan Becker, Sharon Laskowski, and Svetlana Lowry. "Style Guide for Voting System Documentation: Why User-Centered Documentation Matters to Voting Security". *USENIX/ACCURATE/IAVoSS Electronic Voting Technology Workshop/Workshop on Trustworthy Elections 2009 (EVT/WOTE' 09)* (Aug. 2009). URL: `http://www.usenix.org/events/evtwote09/tech/full_papers/chisnell.pdf`; Dana Chisnell, Susan Becker, Sharon Laskowski, and Svetlana Lowry. *Style Guide for Voting System Documentation*. NISTIR 7519. National Institute of Standards and Technology, 2008. URL: `http://vote.nist.gov/NISTIR-7519.pdf`; Maureen Stone, Sharon Laskowski, and Svetlana Lowry. *Guidelines for Using Color in Voting Systems*. NISTIR 7537. National Institute of Standards and Technology, 2008. URL: `http://vote.nist.gov/NISTIR-7537.pdf`; Janice Redish, Dana Chisnell, Ethan Newby, Sharon Laskowski, and Svetlana Lowry. *Report of Findings: Use of Language in Ballot Instructions*. NISTIR 7556. National Institute of Standards and Technology, 2009. URL: `http://vote.nist.gov/NISTIR-7556.pdf`; Janice Redish and Sharon Laskowski. *Guidelines for Writing Clear Instructions and Messages for Voters and Poll Workers*. NISTIR 7596. National Institute of Standards and Technology, 2009. URL: `http://vote.nist.gov/032906PlainLanguageRpt.pdf`

reinstated in VVSG v1.1, even if the full usability test suite is not ready.

Whatever the status, given the inescapable reality that usability problems result in tens of thousands of lost votes each year, we urge that work towards incorporating usability benchmark testing into the VVSG be given high priority and even accelerated, if possible.

8 System Documentation and Technical Data Package Requirements

8.1 Standardized Configuration Checklists as a Foundation for Assessing Auditing Functionality

Requiring manufacturers to submit documentation establishing that the configuration of operating system components is "consistent with current best practices for operating system security" is a potentially powerful way to guard against security problems attributable to misconfiguration.[117] It is unclear, however, whether the guidance in § 2.1.5.2 will cover all relevant voting system components.

In the past, a serious obstacle to the proper use of voting systems has been the lack of specific instruction from manufacturers about how to set up and configure COTS components used in their voting systems. At minimum, this lack of guidance can make voting systems difficult to set up and operate properly in the field, which can itself lead to insecurity.[118] Worse, in one prominent case, an apparent misconfiguration of an election management system's audit logging capacity deprived post-election auditors of evidence that might have been helpful in understanding events during an election tally.[119] Such errors obviously interfere with the audit trail's purpose of "provid[ing] the supporting documentation for verifying the accuracy of reported election results" through a "concrete, indestructible archival record of all system activity related to the vote tally" that not only serves as "evidence in the event of criminal or civil litigation" but also helps establish "public confidence in the accuracy of the tally."[120]

VVSG v1.1 begins to remedy this problem through new guidance in § 2.1.5.2.[121] This section states that manufacturers must submit documentation establishing that any "multitasking operating system"[122] used in its voting system configured in a way that is consistent with "current best practices for operating system security." The EAC, in turn, will determine what these best practices are, using the configuration checklists from the System [sic] Content Automation Program (SCAP) as a baseline.[123]

Though these checklists provide an excellent foundation for evaluating the operating systems for which they have been written, that list is limited. It does not include custom operating systems that

[117] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 2.1.5.2.

[118] In part to address this security concern, the California Secretary of State requires manufacturers to provide voting system use procedures as a condition of certification. California Secretary of State. *Voting System Use Procedures for California Template*. 2006. URL: `http://www.sos.ca.gov/elections/voting_systems/use_procedures_2006.pdf` Creating such requirements on a state-by-state basis, however, creates the prospect of conflicting requirements from states, or multiplicative work for manufacturers, or both.

[119] Candice Hoke. *Monitor Report: Possible Legal Noncompliance in the November 2006 Election*. CSU Center for Election Integrity/Public Monnitor of Cuyahoga Election Reform, 2007. URL: `http://josephhall.org/ccoh/20070108_publicmon_report.pdf`, 12.

[120] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, § 2.1.5.

[121] This new guidance is in addition to the existing audit trail-related functional requirements in VVSG 2005, which state that all multitasking operating systems must (1) use authentication on "on the local terminal ... and on all external connection devices" and (2) enable logging "for all session openings and closings, for all connection openings and closings, for all process executions and terminations, and for the alteration or deletion of any memory or file object." Ibid., § 2.1.5.2

[122] This phrase is defined to mean: an operating system "capable of executing multiple application programs simultaneously." Ibid., § 2.1.5.2

[123] SCAP actually stands for *Security* Content Automation Protocol. The correct name of the program should be replaced throughout the VVSG v1.1. *See:* National Institute of Standards and Technology, Information Technology Laboratory. *The Security Content Automation Protocol (SCAP)*. URL: `http://scap.nist.gov/`

are used in some current voting systems that may serve as a foundation for systems submitted for certification under VVSG v1.1.[124]

Section 2.1.5.2 also states that the acceptability of deviations from the relevant checklist "will be decided on a case-by-case, model by model, revision by revision basis, primarily by the VSTL, and then presented to the EAC for approval." A commitment by the EAC to make the basis for these decisions public would be helpful to all stakeholders: manufacturers, VSTLs, election officials, and voters.

8.2 The Benefits of More Complete Security Specification Requirements

8.2.1 Revise VVSG v1.1 to Require a Full Threat Analysis in TDPs

In addition to the configuration-related requirements discussed above, VVSG v1.1 contains several more general documentation requirements that should be useful not only in evaluating voting system security but also in helping manufacturers structure their overall approach to security. Still, some further changes are necessary to bring these parts of VVSG v1.1 into alignment with current computer security practices.

The most important of these changes is the set of documentation requirements in VVSG v1.1 Volume 2, § 2.6, Table 1, which require manufacturers to submit the following as part of their TDPs:

1. Design and Interface Specification
2. Security Architecture
3. Development Environment Specification
4. Security Threat Analysis
5. Security Testing and Vulnerability Analysis Documentation

Taken together, the requirements for the Design and Interface Specification, Security Architecture, and Security Threat Analysis come close to requiring a full threat model. But the full descriptions of these documents are vague and omit some important elements of a full threat model. As currently written, § 2.6 could lead manufacturers to miss important threats or leave them unaddressed. The current wording of the Security Threat Analysis requirement is a good example of this potential. It begins: "This document shall identify the threats the voting system protects against... "[125] Identifying the threats the system protects *against* provides little assurance that all relevant or feasible threats have been addressed.

Fortunately, existing threat analyses for voting systems may provide helpful guides for VVSG v1.1. A recent example is *A Threat Analysis on UOCAVA Voting Systems*, which sets forth the general steps in threat modeling[126] and goes through them for numerous voting system types.[127] This report shows that a full threat analysis moves from a description of the system, including the kind of information it handles and its security goal, to an identification of threats—"events or circumstances that are potential violations of security."[128] Then a more detailed analysis of each threat begins, including the identification

[124] For example, Hart InterCivic, Inc. uses an operating system, MQX, which they own and maintain. *See:* Joseph Lorenzo Hall and Laura Quilter. *Documentation Review of The Hart Intercivic System 6.2.1 Voting System*. July 2007. URL: `http://www.sos.ca.gov/elections/voting_systems/ttbr/hart_doc_final.pdf`, 30.

[125] A fragment of this requirement appears in the Design and Interface Specification requirement: "This document shall identify the threats the voting system protects." This sentence is out of place for a design specification. In any event, it contains a significant typographical error; it should read: "This document shall identify the threats the voting system protects *against*."

[126] Andrew Regenscheid and Nelson Hastings. *A Threat Analysis on UOCAVA Voting Systems*. NISTIR 7551. NIST, 2008. URL: `http://www.vote.nist.gov/uocava-threatanalysis-final.pdf`, 23-26.

[127] Ibid., 27-66.

[128] Ibid., 23.

of threat sources, followed by assessments of *effort* ("the level of difficulty of performing a successful attack based on a threat"), the level of difficulty of *detecting* an actual attack, the *impact* of a successful attack, and possible *controls* to mitigate attacks.[129]

Thus, the UOCAVA threat analysis emphasizes the importance of two elements that are not clearly required by VVSG v1.1 § 2.6. First, as stated above, this section does not appear to require identification and analysis of all *feasible* threats.[130] Second, the current Security Threat Analysis requirement does not clearly require any assessment of the effort or impact of a particular attack. These measures need not be quantitative; a value of high, medium, or low effort may be sufficient.[131] Such qualitative assessment would provide significant help in manufacturers' and others' assessments of the overall level of security risk in a particular voting system.

The remedy for § 2.6 is simple: the Security Threat Analysis description should be revised to include the six elements discussed above:

- Characterize of the voting system's architecture
- Identify threats
- Identify threat sources
- Evaluate effort involved in carrying out an attack
- Evaluate difficulty of detecting an attack
- Evaluate impact of an attack
- Identify and evaluate possible controls

8.2.2 Requiring Documentation of Physical Security Mechanisms Will Aid Security Evaluations

A second significant improvement in documentation is the requirement that manufacturers "provide a technical data package that documents the design and implementation of all physical security controls for the voting system."[132] Given the recent attention devoted to physical security mechanisms on voting systems,[133] this requirement will serve to alert VSTLs of the use of physical seals, locks, and other mechanisms used in the voting system. Indeed, VVSG v1.1 goes further to require that this documentation spell out what each physical security countermeasure protects, which can be cross-checked with the required list of "all voting system components to which access must be restricted." We believe this requirement is well constructed and will aid voting system testing by calling attention to physical security controls, rather than allowing them to remain obscure to evaluators.

8.3 Expanded Requirements for Technical Data Package Contents Are Warranted

As ACCURATE has pointed out in previous comments submitted the EAC, the Technical Data Package (TDP) is a critical element of a manufacturer's submission to a VSTL.[134] As pointed out in Section 4.1,

[129] Regenscheid and Hastings, see n. 126, 23-26.

[130] Compare with ibid., 23 (stating that "[t]hreat sources are groups or individuals that could feasibly attack a voting system").

[131] Ibid., 25.

[132] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, Vol. 2, § 2.6.5.

[133] Andrew W. Appel, Maia Ginsburg, Harri Hursti, Brian W. Kernighan, Christopher D. Richards, Gang Tan, Lehigh University, and Penny Venetis. "The New Jersey Voting-Machine Lawsuit and the AVC Advantage DRE Voting Machines". *Electronic Voting Techology Workshop/Workshop on Trustworthy Elections 2009 (EVT/WOTE 2009)* (Aug. 2009). URL: `http://www.usenix.org/event/evtwote09/tech/full_papers/appel.pdf`.

[134] A Center for Correct, Usable, Reliable, Auditable and Transparent Elections, *Public Comment on VVSG II*, see n. 4, 17-18; Aaron J. Burstein, Joseph Lorenzo Hall, and Deirdre K. Mulligan. *Public Comment on the Manual for Voting System Testing*

the TDP can help VSTLs (and other reviewers, if they obtain the TDP) to identify quickly software components that a manufacturer may have chosen over more trustworthy alternatives. Along these lines, adding compilers and interpreters to the TDP listing requirements is a welcome change.[135]

Still, as pointed out in Section 4.1 of this Comment, VVSG v1.1 should give firmer guidance to VSTLs to view the list of software components as the beginning of an inquiry into the system's security, rather than mere compliance with a cataloging requirement. In addition, tests that manufacturers themselves performed are potentially valuable sources of information for test labs. For example, test suites for unit testing source code and performing regression testing are crucial tools for developing software for mission critical environments such as voting.[136] Therefore, we recommend that the EAC expand the requirements for the Development Environment Specification and Security Testing and Vulnerability Analysis Documentation[137] to include any test suites the manufacturer used to perform testing.

8.4 Requirements for Identifying Protected and Confidential Information

Volume II of the proposed Guidelines provides instructions for submitting confidential and proprietary information that balance protecting competitive interests in this information with public access to it.[138] The requirements are broadly consistent with those found in Chapter 10 of the *Voting System Testing and Certification Program Manual*, and § 2.1.3 of VVSG 1.1, Volume II, should cross reference the *Manual*. In particular, § 2.1.3.iii[139] should make it clear that analysis of competitive harm is only required in the event the manufacturer objects to the EAC's proposed release of information, or the EAC refuses to release information to a third party requester on grounds of the Freedom of Information Act exemption for trade secrets and confidential business information.[140]

9 Conclusion

While we believe the intent of the EAC to update the VVSG 2005 is well-placed, we are not convinced that the draft VVSG v1.1 can serve this purpose without substantial revision. More significantly, we believe that putting off complex and controversial changes merely drags out what is necessary to achieve the goal of driving trustworthy voting systems and likely raises the price tag due to serial revisions that will be required under this incremental approach. The stated goals of the revision are not met by the text of the guidelines, especially in that there are a number of requirements that will necessarily translate to complex hardware and/or software changes.

Of the four core elements we discussed at length in our last public comment on the draft VVSG II—software independence, adversarial vulnerability testing (OEVT), usability benchmark testing and volume testing—this proposal makes no progress.[141] Some of the predicted substantial cost of these proposed revisions results from the more directive and detailed requirements proposed to edge closer toward this end. Specifically, a software independence requirement would shift some of the burden of security assurance away from the new security requirements (discussed in Section 5); that is, if voting systems

& Certification Program (submitted on behalf of ACCURATE to the U.S. Election Assistance Commission). Oct. 2006. URL: `http://accurate-voting.org/wp-content/uploads/2006/11/ACCURATE_VSTCP_comment.pdf`, 9-10.

[135] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, Vol. 2, §§ 2.2.1 & 2.5.5.2.

[136] *See* the discussion of modular software testing in Section 5.1 above.

[137] U.S. Election Assistance Commission, *Draft VVSG v1.1*, see n. 3, Vol. 2, § 2.6.

[138] *See*, in particular, § 2.1.3.

[139] "If necessary, provide additional documentation or information. For example, if the manufacturer claims a document contains confidential commercial information, it would also have to provide evidence and analysis of the competitive harm that would result upon release.would result upon release."

[140] 5 U.S.C. § 552(b)(4).

[141] A Center for Correct, Usable, Reliable, Auditable and Transparent Elections, *Public Comment on VVSG II*, see n. 4.

are designed such that no undetected software error can cause an undetected error in election results, the certification process need not go into the level of detail of, for example, § 5.2.8's lengthy requirements for error-checking.[142] The new requirements for supporting auditing (discussed in Section 3), standardizing cryptography (discussed in Section 4) and the software workmanship and validation requirements (discussed in Section 5) represent important but modest steps towards robustly auditable systems, software independent or not.

We are confident that the testing-related revisions in VVSG v1.1 will prove beneficial, yet here too a more comprehensive approach is preferable. The record shows that both volume testing and adversarial vulnerability testing (OEVT) are invaluable for discovering reliability flaws and potentially exploitable security vulnerabilities.[143] It is unclear why these types of testing are not included in some form in VVSG v1.1, even if scaled back from the proposals in the draft VVSG II. Relatively, we find the greatest merit in the new accuracy and reliability testing (discussed in Section 6), usability and accessibility requirements and documentation requirements (discussed in Section 8). We believe these proposed revisions have the potential to reform the testing and certification process substantially. As we argue in Section 7, we are puzzled that even modest requirements for usability benchmark testing were not included in VVSG v1.1; if any of the five benchmarks are ready, they should be required now so that manufacturers, election officials and voters all see the clear benefits from usability conformance as soon as possible.

We hope that this analysis of VVSG v1.1 has been helpful in illustrating our concerns with the substance of the revision in light of the stated goals of the EAC in putting forth an incremental VVSG update. Ultimately, the VVSG will have to go through the overhaul contemplated by the VVSG II in order to further drive the level of trustworthiness our election systems require. Interdisciplinary academic researchers like those at ACCURATE stand ready to assist this mission.

[142] While NIST has put forth robust auditing as a possible middle-ground in the path towards software independent systems, they recognize that *detecting* errors is quite different from *preventing* them. *See:* National Institute of Standards and Technology, *Reasearch Areas for the TGDC VVSG Recommendations*, see n. 7.

[143] California has performed volume testing on voting systems submitted for state certification for a few years. California, Ohio and Florida have engaged in adversarial vulnerability testing to independently evaluate the integrity of their voting systems.