OSET INSTITUTE
Briefing

A New Architecture for Trustworthy Voting Systems

Prepared for:
U.S. Election Administration Leadership

Prepared By:

E. John Sebes
Co-Founder & Chief Technology Officer

&

Edward Perez
Global Director, Technology Development

Original Draft: June 2010 by Sebes, Miller
Revised Publication: April 2019 | v4.0

1. Introduction

It is well settled that current Voting System Technology (VST) is not considered "trustworthy" by definition in a secure computing context, because of its basis on commodity Personal Computing (PC) hardware and Operating Systems (OS) software that does not support trustworthy computing.

Of all the voting systems in use for U.S. federal, state, and local elections (as well as for elections in other democracies), none were designed and developed using "trusted computing" concepts and principles that have been used for decades in high-security computing for government critical systems. We believe this was in part due to two reasons:

1. There was an entirely different set of presumptions about threat models and so-called "attack surface" when companies seized on the opportunity of the Help America Vote Act to develop computer-based electronic voting systems in response to the desire to replace punch-cards and lever machines in 2002; and
2. Even if there had been some amazing insight in security-centric risk modeling, the companies who were already in the business of goods and services for election administration, as well as new ones from similar (but different) businesses such as ATM machinery or gambling machines, did not have the core capabilities or domain expertise of trusted computing design and engineering.

Trusted computing concepts help system architects and engineers isolate "mission-critical" components of a computer system (*including voting systems*) to always behave in expected ways (*such as properly counting and tabulating votes*).

In addition, none of the available voting systems in use across the U.S. have ever had a formal cybersecurity assessment using the standards and techniques used for commercial computer systems certified for critical infrastructure activities.[1]

Prior independent investigations of voting systems, such as the State of California's Top-to-Bottom Review (TTBR) [2] in 2007 and the State of Ohio's 2007 Evaluation and Validation of Election-Related Equipment, Standards and Testing (Everest) [3] report, documented a variety of vulnerabilities, including software quality issues and basic errors in security functionality of voting systems. For these stated reasons, a new architecture for voting systems design is required as a prerequisite to obtaining truly "trustworthy" voting systems technology.[4]

The new design proposed in this paper is centered on three principles:

1. **Interoperability**, with data exchange based on common data standards;
2. **Segmentation**, to separate complex systems into separate segments with distinct functions; and
3. **Minimized Architecture**, to reduce the attack surface for "mission-critical" functions.

1.1 Background

There are several documented causes of Voting System Technology (VST) security gaps, ranging from uncontrolled use of "untrustworthy" hardware components; poor software quality design; improper product delivery practices (e.g., hard-coded, and default passwords), and insufficient documentation for security administration of VSTs.

Although these and other factors apply to a variety of voting system products, there is one root cause these voting systems have in common:

Current VST was not designed with a cybersecurity focus.

Indeed, the basic architecture of current VST and design of various parts of VSTs were designed at a time when cybersecurity was neither a national-security issue, nor a time

[1] See: https://www.niap-ccevs.org/Ref/What_is_NIAP.CCEVS.cfm

[2] See: https://www.sos.ca.gov/elections/voting-systems/oversight/top-bottom-review/

[3] See: https://www.eac.gov/assets/1/28/EVEREST.pdf

[4] We note at the outset that an over-arching design objective is achieving the so-called VAST mandate of a trustworthy election. An election once performed is said to be trustworthy if and only if a four-corners test for trust is fulfilled: that the system is Verifiable, Accurate, Secure, and Transparent (in process). This is important for two reasons:

(1) The defense of democracy must focus on sustaining *trust* at multiple layers of the administration of democracy—from campaigns and electioneering (completely out of scope of the OSET Institute charter) to election administration and voting (the technology of which is the focus of the Institute's charter); and

(2) An essential and imperative principle in the next generation of election technology design is *transparency*—from the architecture to the development and engineering, or the implementation and deployment. For the work of the Institute (and the "OS" in our name, OSET Institute) transparency is achieved through publicly available (i.e., open source) technology.

when U.S. elections infrastructure faced evident homeland security threats from state and non-state actors. Thus, commercial voting system vendors were not expected to make significant investments in a process of green-field product development with cybersecurity as its foundations.

There were, however, powerful market forces motivating these vendors to bring product to market rapidly, in response to the passage of the Help America Vote Act of 2002. Particularly during the years immediately after the bill's passage, market demands focused on usability, accessibility, and operational efficiency, and there was no focus on trusted computing concepts or cybersecurity principles. For the same reason, the market did not provide vendors with a return on investment for efforts to return to the drawing boards for security-centric VST development.

More recently, some next-generation VSTs have shifted to a basis on commercial off-the-shelf (COTS) embedded systems platforms that have some system security features, notably a reduction in extraneous software packages, and methods for detection of tampering with software as stored. However, these embedded system platforms still use PC hardware, and have a basis in PC operating systems. As a result, system protection mechanisms have been proven they can be bypassed with malicious software, tampering with running software, or physical tampering.[5] Although these embedded system platforms have some security features, their vulnerabilities show that they were not designed to provide the level of robust cyber-security that is required for the current environment: systems that are designated as critical infrastructure (CI) by the U.S. Department of Homeland Security; election officials now being CI operators with a mandate that includes the cyber-defense of critical systems; and a demonstrated threat of nation-state adversaries that must be contained by cyber-defense.[6]

1.2 Scope and Outline

The scope for this new architecture definition is system-architecture level specification of the components of a trustworthy voting system, sufficient to guide trustworthy VST development to fill the unmet security and integrity needs of election CI, filling a technology gap that has significant national security implications. Filling this gap requires a return to the drawing board with regard to VST, starting with a security-centric architecture for VST that could yield trustworthy systems. This paper presents such a new architecture for voting systems to address the new threat environment for VST cybersecurity. A security-centric architecture has been adopted by the OSET Institute to develop security-centric VSTs. However, the architecture can be applied broadly to the development of other critical infrastructure (CI) systems that have similar operational requirements to VST (e.g., financial, power delivery, and transportation management systems).

[5] See: https://www.fracturelabs.com/posts/2017/exploiting-ms17-010-on-windows-embedded-7-devices/

[6] See: https://www.eac.gov/election-officials/elections-critical-infrastructure/

A security-centric architecture assumes two criteria to define the scope of a trustworthy voting system: an evidence-based voting system with voter-verifiable paper ballots of record; and support for risk-limiting auditing of ballots to determine whether election results may have been compromised by anomalies in voting system technologies. More specifically, this architecture addresses the most currently feasible definition of such a voting system: paper ballots that are either hand-marked or machine-marked (the latter typically for accessibility), or counted by a trustworthy optical scanner ballot counting system.

Section 2 provides an overview of the system architecture that is common to most current VST used in U.S. elections. Section 3 provides a new segmentation schema for trusted voting system architecture. Section 4 discusses the benefits that are realized by using minimal embedded systems for mission-critical components. Section 5 provides an overview of design principles that can guide the design of security–centric architecture for voting systems.

A security-centric architecture, like any architecture for a trustworthy distributed system, does not ensure that any specific system built on such architecture will demonstrate any particular trust properties of any particular cybersecurity functions. But trustworthy system architecture *is* a prerequisite for a trustworthy system. With a sound architecture, it becomes possible for system engineers to apply a number of design and implementation principles that are also needed for a trustworthy system.

1.3 The Role of Standards and Common Data Formats

One of the important aspects of the new architecture is the redefinition of a voting system as a combination of discrete computing appliances that interoperate with one another via well-defined data interfaces. This model is to be distinguished from current federal certification guidelines, which understand a "voting system" only as the *total combination* of specific mechanical, electromechanical or electronic sub-components (including software and firmware) used to perform a variety of voting functions.[7] In other words, current certification schemas for "a voting system" typically entertain interoperability with external systems, but they do not entertain interoperability among discrete, modular appliances that could come from a variety of sources, and that do not, together constitute a "voting system."

The newer component-based interoperability model became possible through the development of common data formats that serve as the common language for interoperation. Relatively recent and important work by the U.S. National Institute of Standards and Technology (NIST) has defined a standard set of common data formats specifically for election data interchange, documented in NIST Guidelines Publication 1500-100. These formats are essential for the feasible implementation of the type of modular architecture defined here. In this document, whenever modules or systems exchange data, the implicit assumption is that the data exchange is based on the NIST 1500-100 standards.

[7] See: https://www.eac.gov/assets/1/28/VVSG.1.1.VOL.1.FINAL1.pdf

2. Current VST Architecture: Non-Segmented

At its highest level, a new VST architecture presents an opportunity to incorporate an important architectural principle of *segmentation*, or the division of a complex system into separate segments with distinct functions, interoperating via well-defined interfaces. Segmentation provides many benefits in terms of security and flexibility, to cite just two examples.

In current VST system architecture, however, the benefit of segmentation is limited for several reasons.

First, most current voting systems were designed around only two parts:

- A multi-function software package often referred to as an Election Management System (EMS), often deployed on a personal computer in a back-office environment of a county registrar, or a local elections office, or a county IT department; and
- Individual voting appliances – hardware/software devices such as voting machines, ballot marking and recording devices, and optical scanning devices that count paper ballots.

Within this merged, amalgamated approach, segmentation is further limited in current commercial VSTs because:

- Interfaces are proprietary.
- The EMS typically incorporates a wide variety of functions (e.g., ballot design, device configuration, results reporting) into one comprehensive software family whose design (and potential flaws) is not easily disentangled; and
- Voting devices run on an untrustworthy operating system platform that was not designed for cybersecurity.

In the simplest workflow, each voting device performs a single function (e.g. ballot recording, ballot marking, ballot counting), and interacts only with the EMS, by both (a) receiving data from EMS via removable storage media, and (b) sending data back to the EMS also by removable storage media. In more complex workflows, some hardware devices have multiple functions, and communication services that use networking services (e.g., ballot counting devices that also serve to communicate unofficial election night vote tally data over a closed network (or via the Internet) to an EMS).

2.1 Monolithic EMS Structure

Aside from the functions of the voting machines, nearly every other function of a voting system, as well as other functions, are combined into the EMS as one software package with multiple functions.[8] There may be functional separation within such a monolithic body of EMS software, or even some software modularity with different functions in

[8] See: Section 5, https://votingsystems.cdn.sos.ca.gov/oversight/ttbr/diebold-source-public-jul29.pdf

separate modules; but each of these modules is vulnerable to faults originating in other modules.

Figure 1 illustrates a typical EMS deployed on general purpose, personal computer (PC) hardware with an operating system ("OS") (e.g., Windows or Linux). Figure 1 also shows the possible modularity within an EMS; the sub-components represent distinct functions within the EMS. The EMS itself is an additional application software package implemented to run on the PC's OS and hardware. In older voting systems, contrary to best practices, EMS software may also be deployed along with other typical PC applications for email, web browsing, document preparation, etc.

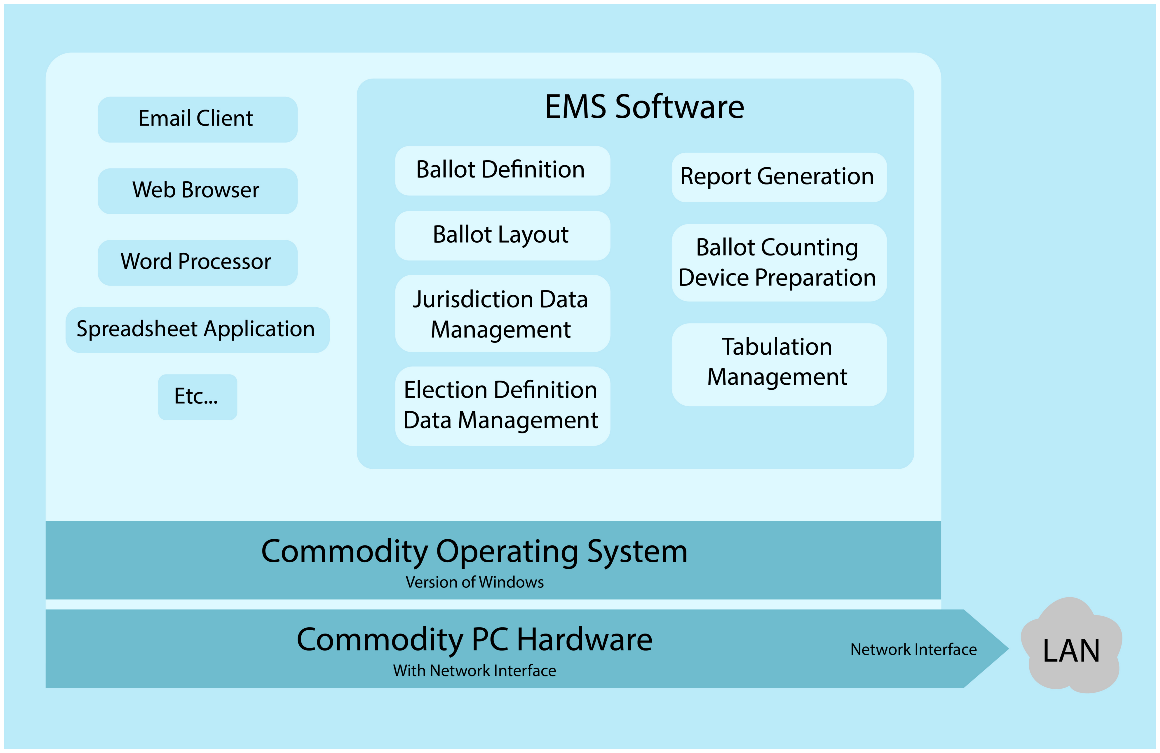

Figure 1: Current Monolithic EMS Software Architecture

Two of the EMS functions (ballot-counting device preparation and tabulation management) are "*mission-critical*." In particular, the preparation of ballot-counting devices is a mission-critical function; incomplete or incorrectly formatted configuration datasets (sometimes called election definition files) can cause ballot-counting machines to process ballots incorrectly.

Contrary to current federal voting system definitions, some EMS functions, such as ballot layout and report generation, for example, need not be part of a voting system or EMS at all, as these functions can be supported by other commonly used commercial-off-the-shelf ("COTS") software. Though any individual EMS might have multiples of these ancillary functions, most EMSs have at least one: a report preparation module, which converts election results data into a variety of reports and formats (e.g., PDF documents of vote totals, spreadsheets of precinct-level vote tallies, etc.)

The light blue rectangles represent the several kinds of election-specific data management functions that are typical of most EMSs. Election preparation involves the management of a number of types of data, including jurisdiction data management (district and precinct definitions, geospatial information), and ballot items (contests, candidates, and ballot questions). Election management also includes the combination of this data with ballot rules and logic (e.g., numbers of valid choices; rotation; and voting variations such as ranked choice or cumulative voting, etc.), to create a specification for each ballot. Ballot layout is the process of creating a printable document for the paper ballot of each ballot specification, as well as preparing data for devices that present digital, on-screen ballots.

All of these preparation activities culminate in the generation of datasets that each type of mission-critical voting machine needs in order to present or count ballots.

The other mission-critical function is that of tabulation, which is typically performed by software that is part of the EMS. Ballot counting devices produce cast vote records (and, in some systems, ballot images). Tabulation software consolidates all this data, and combines vote tallies to create election results and other reports. The final result of the tabulation process is a set of data that includes the consolidated tallies as well as more detailed data, such as precinct-specific results. This data is sometimes published, and is always post-processed by a report generation process.

2.2 Monolithic EMS Consequences

From a design and engineering perspective, the result of the architecture shown in Figure 1 is a security conundrum for the following reasons:

- *Shared vulnerabilities.* All of the software in a monolithic EMS can harm any part of the EMS.
- *Low assurance*. At least some of the software was not developed specifically for high security, and cannot be regarded as trustworthy (this includes a computer's operating system (OS) that hosts the EMS software).
- *Large attack surface*. The amalgamation of many EMS functions in one software package creates a large attack surface.
- *Operating system vulnerabilities*. The OS, networking capabilities, and a variety of non-election-related applications software enlarge the attack surface further. Software vulnerabilities in non-election related software applications can be used to compromise the whole system, and enable further attacks on the EMS.
- *Potential for modifications*. Besides the un-intentional vulnerabilities present in the OS, the OS by design is capable of running any compatible software, even though the specific configuration of a voting system is required to be unmodified from the certified version.

- *Non-isolated EMS*. Successful attack via any one part of the attack surface can then be used to compromise the mission-critical functions of the EMS.
- *Non-isolated voting components*. A compromised EMS can be the basis for attack on the remainder of the voting system.

As a result, current Election Management Systems are highly vulnerable whether operated in a networked environment or "air-gapped." Significant vulnerabilities result from uncontrolled exchange of data via removable media that can contain unauthorized software applications, malware, or data constructed to exploit vulnerabilities in the OS, or in the EMS.

3. New VST Architecture: Segmentation of the EMS

The first step toward designing a new security-centric architecture is increased segmentation, which involves dividing the EMS into four (4) parts. In addition to segmentation, this first step also includes some degree of *minimization*: removing from a target system some inessential functions to make the system a simpler, smaller attack surface target.

The segmentation of the EMS has two essential outcomes:

1. *Reduction in scope of "voting system" software*: The scope of what has traditionally been considered "the EMS" that undergoes certification is significantly reduced, and it includes only *device configuration* and *tabulation*; and
2. *Segmentation of voting components*: Mission-critical voting components are distinguished from non-mission-critical election administration, such as ballot design and customized results reporting, which are no longer part of the "voting system" that undergoes certification.

3.1 Essential Segmentation

As shown in Figure 2 (Page 10 below), the four kinds of segmentation are:

1. *Separation of mission-critical device configuration function from other election administration functions*. For purposes of Figure 2, this separate component is referred to as a *Device Manager*. With this new architecture, this mission-critical component runs on dedicated hardware, physically separated from the remainder of election administration software. The ballot counting devices noted in Figure 2, the *Precinct Ballot Counter* and the *Central Ballot Counter* are segmented in the same manner. The ballot counting devices receive data created in the 4th segment below, provided via removable media, containing data in the NIST 1500-100 common data format's pre-election use case.
2. *Separation of mission-critical tabulation function from other election administration functions*. For purposes of Figure 2, this separate component is referred to as a *Tabulation Manager*. With this new architecture, the *Tabulation Manager* runs on dedicated hardware, physically separated from the *Device*

Manager and the ballot counting devices. The *Tabulation Manager* receives datasets written by ballot counting devices onto removable media (i.e., USB thumb drives, or CDs), and performs the same consolidation and validation functions as in current, non-security centric VSTs.

3. *Elimination of customized reporting from mission-critical tabulation functions.* Election officials will still require reports of various kinds to be generated from election results data. However, reporting can be done by separate report generation software that receives election results data, which we recommend to be formatted into the NIST 1500-100 Common Data Format for election results reporting. Election officials could choose which COTS report generation system to use, and they may also have the option of choosing a proprietary report generation product from a voting system vendor. Under this new architecture, we assume that in the future, such products could be developed by vendors in a such a way that they would no longer be integrated into a monolithic EMS, and thus able to be more readily customized or extended without having any effect on the certified tabulation component.

4. *Elimination of other election administration functions from mission-critical voting functions.* The remaining EMS functions consist of data management (i.e., jurisdiction- and election-specific data), and ballot layout (i.e. ballot "definition"). Consequently, Figure 2 includes an *Election Data Manager* and a *Ballot Layout Tool*, instead of a traditional, monolithic EMS package. These functions are relatively static, providing a user interface for viewing and modifying stored election definition data.

The ballot layout tool might benefit from being developed by organizations with graphical design and usability test skills that are quite distinct from the skills needed for a data management system. Having a separate ballot layout tool means that election officials could have options for obtaining distinct tools from vendors with design-centric and data-centric skill sets, respectively. Data exchange and inter-operation would be enabled by use of the NIST 1500-100 common data format for pre-election needs.

A separate ballot layout tool is also advantageous because ballot layout requirements vary across election jurisdictions. As a result, the ballot tool could be customizable to individual states' needs, separate from the more data-centric tasks of data management.

A critical result of the segmentation methods discussed in this section concerns the concept of "mission-critical" functions in a voting system. The fundamental mission of a voting system is to:

- Facilitate the casting of ballots with choices marked based on voter intent;
- Accurately count ballots and produce election results
- Demonstrate evidence that the results are based on voters' choices on their marked paper ballots.

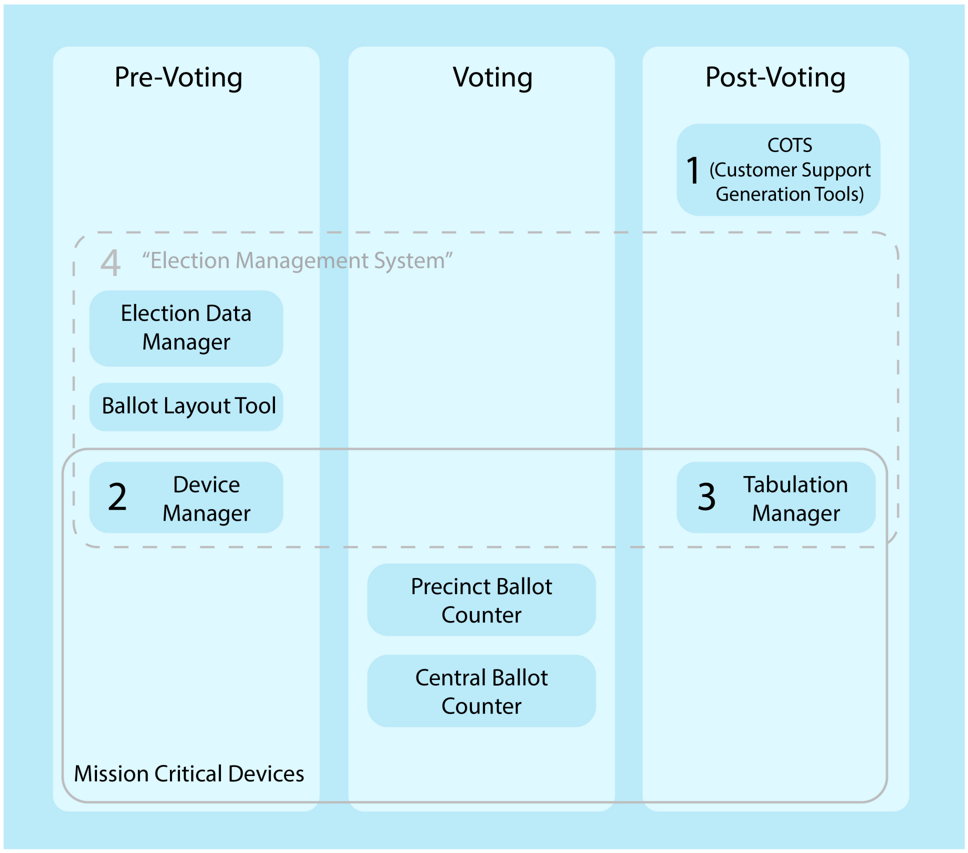

Figure 2: Essential System Architecture Segmentation

Ballot scanning devices that count and tabulate ballots are clearly essential to this mission. The *Tabulation Manager* device is essential since it is the means of creating election results from tallies, and consolidating data that serves as evidence of cast ballots. In both cases, these devices depend on election-specific data that is delivered on removable media that is prepared for these devices by the mission-critical *Device Manager*.

The remaining functions (data management, ballot layout, customized reporting) are certainly needed for election administration procedures, but they are not mission-critical elements of a voting system, strictly speaking.

3.2 Cybersecurity Benefits from Segmentation

In a security-centric architecture, the mission-critical functions have two tangible benefits that are important enablers for cybersecurity. First, all of the mission-critical functions of a voting system are isolated from the non-mission-critical functions. As a result, it becomes possible to implement the mission-critical functions so that they are not affected by dysfunction caused by non-critical functions (*regardless of whether the dysfunction is caused by software anomalies or malicious tampering*). This also reduces the attack surface of voting systems from threat actors.

Second, each of the mission-critical functions of a voting system are isolated in their own hardware/software computing components separated from each other, and from other voting system software for other election management functions as a whole. This isolation provides the basis for other steps in the definition of the new architecture that will be discussed in the next section.

3.3 Increased Flexibility for Non-Critical Functions

With this new architecture, election data management (i.e., the *Election Data Manager* component), and ballot management (i.e., the *Ballot Layout Tool* component) remain outside the mission-critical portion of a voting system. In reference to the data management and ballot layout functions, mission-critical voting devices are entirely data-driven based on data formatted in a common data format (CDF). Any CDF-compliant system or tool could be used to prepare the election definition data and ballot definition data needed by the mission-critical devices. Thus, the data does not need to come from any particular tool or system. Hypothetically, the data could be prepared largely manually via an XML editing tool and manual data entry (the most common representation of the CDF is XML).

Of course, in practice it is extremely convenient to have purpose-built tools for election data management and ballot management. In a narrow definition of a voting system, these non-critical systems might no longer even be part of a voting system *per se*. If so, the benefits could include a significant reduction in the code footprint of a voting system, and corresponding reduction in effort and expense of both preparing for a voting system test and certification process.

However, voting system certification is a state-level process, and different states will make different choices about what collection of components constitute a voting system, and what testing activities are required for certification in the state.[9] For example, the state of Washington has begun a transformation of its election infrastructure where there is a similar segmentation to that described here: much of election management is performed in state-managed central IT systems deployed in a private/public cloud setting, while ballot tabulation remains the only function performed on dedicated hardware managed by county election officials.[10] In other states, there might be a different model, for example, where election data management is considered to be a voting system function that requires certification of software deployed on specific hardware.[11]

Whatever a state's certification policies may be, a security-centric architecture provides the flexibility for a variety of choices. And in cases where the narrow definition applies, the benefits are not limited to the reductions in effort and cost of certifying, delivering,

[9] See: http://www.ncsl.org/research/elections-and-campaigns/voting-system-standards-testing-and-certification.aspx

[10] See: https://www.sos.wa.gov/_assets/office/rfp-18-04.pdf. Detailed information and ongoing history of the project status is here: http://waocio.force.com/ProjectDetail?id=a06U000000eT6MBIA0

[11] See: http://www.ncsl.org/research/elections-and-campaigns/election-administration-at-state-and-local-levels.aspx

and operating certified voting systems. Additional flexibility results from the exclusion of most election management functions from the certified software base, and/or the base of software that must be performed with dedicated hardware managed by county election officials. For the non-critical but practically required voting systems, state-specific customization of election-management software is enabled without the constraint of re-certification, as is continual improvement and feature extension over time. Also, there is greater flexibility from choices among deployment options ranging from cloud-based services (public, private, or hybrid-cloud services), or datacenter based (where government datacenter employees have full control of IT assets), or based on dedicated hardware.

3.4 Segmented, Minimized Architecture

Figure 3 below illustrates the segmentation, the flexibility for non-critical components, and the remaining scope for a security-centric architecture. In the upper box of Figure 3, non-critical components are simply ordinary software packages. Below each is the deployment platform layer, which illustrates several deployment options of software on COTS platforms including: any of several cloud-based environments; a government-managed datacenter, or dedicated hardware on a local election official's PC. All these election administration components are considered Election Administration Services, and are no longer part of the voting system *per se*. As a result, the Election Administration Services are not subject to a certification process with a much greater flexibility in procurement, customization, and deployment.

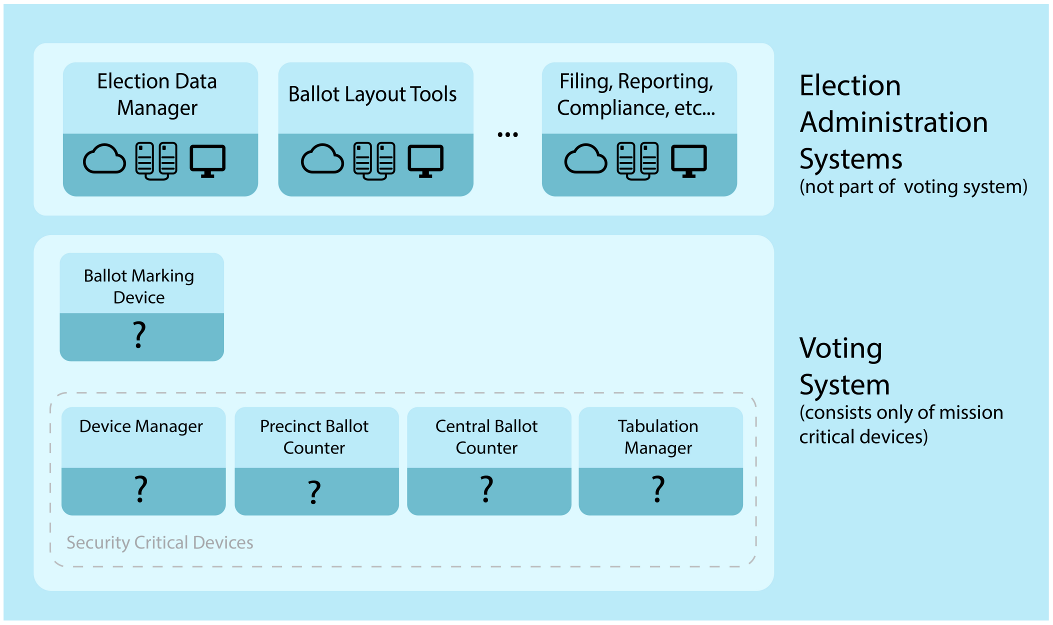
Election Data Manager
Ballot Layout Tools
...
Filing, Reporting, Compliance, etc...
Election Administration Systems
(not part of voting system)
Ballot Marking Device
?
Device Manager
?
Precinct Ballot Counter
?
Central Ballot Counter
?
Tabulation Manager
?
Security Critical Devices
Voting System
(consists only of mission critical devices)

Figure 3: New Voting System Architecture — Minimization

In the lower half of Figure 3 (above) is the voting system consisting only of individual voting system components each with a specific, mission-critical function. These components include a *Device Manager*, a *Ballot Marking Device*, a *Precinct Ballot Counter*, a *Central Ballot Counter*, and a *Tabulation Manager*. Each of these voting system components has their own embedded application software (*illustrated as a solid rectangle*). A further distinction is made between "security-critical" devices, which record votes, and the ballot-marking device, which does not.[12]

The specific nature of each device's underlying platform (i.e., hardware, firmware, and operating system software layers) is illustrated as a question mark ("**?**"). It is clear that any untrusted commodity operating systems are *not* a suitable device platform. A trusted systems platform is needed, as described in the Section 5.

A comment is in order regarding one device: the BMD. Our policy position is that BMDs should be restricted to accessibility applications and that in any event they must produce a complete and full ballot of record suitable for hand (re)counting, without encoded choices, and initially counted by technology parsing for marked choices (i.e., blackened circles next to the voter's choice).

3.5 Worked Example: Los Angeles County Vote-By-Mail

A recent worked example of this architecture is a voting system developed by Los Angeles County, and certified by the state of California, for use that is limited to by-mail voting.

This certified voting system consists only of a central ballot counter, a device-manager-like component that prepares election-specific configuration data for the counter, a tabulation-manager-like component that combines vote tally data from multiple runs of the ballot counter, and a component for sanitizing the USB removable media that is used to transfer data between these components.

Outside of the certified voting system is the election management system (EMS) that Los Angeles County previously developed for its own use. The device-manager-like component acts as a bridge between the EMS function of election definition, and the packaging of election definition data for use by the central counter.

4. Minimization of Components and Interactions

"Minimization" at the component level of this architecture is defined as the limitation of interfaces and dependencies to only those that are required for a narrow set of voting functions, and no more.

[12] The fact that Ballot Marking Devices do not (*and must not*) record votes means that BMDs could be considered by some states as not *security*-critical (i.e. in contrast to precinct- or central-count ballot scanners, for example). For states that choose to regard a BMD as security critical, this paper provides guidelines for minimized embedded systems. However, for other states that choose to regard a BMD as *mission*-critical, but not *security* critical, there will be considerable flexibility in hardware and software platform options for BMDs.

With that in mind, at this point in the definition of a security-centric voting system, we have:

- Segmented the voting system so that the mission-critical parts of the voting system are isolated into individual hardware/software devices that perform a single function.
- Completed the architecture of a voting system as a system-of-systems with inter-operating components.
- Minimized the functionality of the security-critical voting system components.

The next step is to analyze the architecture of each of these security-critical devices as individual systems in order to:

- Identify dependencies among these devices, and how they inter-operate with one another.
- Provide goals for the security-centric design of these devices, sufficient to support implementation of trustworthy devices.

4.1 Dependencies and Inter-Operation

The security-critical voting system components inter-operate with one another mainly via data exchange on removable media (i.e., USB thumb drives, SD card, optical media, etc.). The main exception is in the case of the BMD, which interoperates with ballot counting devices (i.e., Precinct Ballot Counter and Central Ballot Counter components) via paper ballots produced by the BMD and consumed by the ballot counting devices.

The external interfaces are:

- Tabulation Manager — which exports election results data to be used by any external system. The only functional dependency is that in order for the data to be used, an external device that either uses the election-specific data, and/or distributes it to other systems that utilizes the data, must read the removable media, which is exported by the Tabulation Manager.
- Device Manager — which depends on an external election data management system to provide an election definition dataset that is the partial basis for configuration of other devices. As the single point of interface with untrusted systems, the Device Manager's critical security functional requirement is to validate the data imported from removal media, to ensure that it conforms to the NIST 1500-100 common data format (CDF).

These are the interdependent interfaces among the devices:

1. The Device Manager exports election definitions and configuration data to other devices via removable media.
2. Each device depends on the Device Manager for this data.
3. Ballot counting devices interact with paper ballots cast by hand or produced by a BMD.

4. Ballot counting devices export vote tally data.
5. The Tabulation Manager depends on the ballot counting devices to provide vote tally data.

Each of these interfaces and dependencies are required to implement a “system-of-systems” within a voting system.

5. Minimization & Security Goals for an Embedded System Platform

This new architecture enables (but cannot ensure) a technical approach in which a voting system can be designed and implemented for cybersecurity. The component-level architecture ensures that the voting system as a whole is reasonably minimized in terms of components. In addition, each component is an embedded system that consists of one embedded application (ballot marker, ballot scanner, etc.) hosted on an operating system and hardware platform that provides important security and integrity properties at the level of a single hardware/software device.

The remainder of this new architecture focuses on the platform, both security related design goals, and functional goals.

5.1 Security-Centric Design Goals

The starting point of security-centric design is minimization of the software on each voting system component, and using a minimized operating system (OS) as the platform for the embedded software of each device. This approach is in contrast to older VST, which have a low degree of computing assurance in part because they include general-purpose operating system distributions containing unnecessary software, which allows both the OS itself, and its applications to be modified externally. Some more recent VSTs, based on COTS embedded system platforms, do have some degree of minimization of extraneous software, but the embedded system platforms retain PC-era vulnerabilities for modification of software, or bypass of mechanisms to detect modification.

Because of vulnerability to software tampering, both these platform choices are antithetical to the requirements for voting systems that must be certified before use, and may not be modified during use. As with any type of system that has a specific function, they must pass a certification and approval process (either at the federal or state level), and voting devices may only be used in the certified configuration. Once the specific configuration has been certified, modified components should not be used, as doing so would risk compromising the entire voting system.

By contrast, this new architecture places security-critical voting system devices in a new, fairly recent threat model that includes nation-state adversaries capable of tampering with voting systems. In this threat model, each security-critical device must be designed “from the ground up” for robust cyber-defense against tampering.

There are several objectives relevant to the design of each of security-critical/mission-critical devices:

- Minimizing the platform hardware to the exact set of hardware components, and peripheral devices that are required for the voting system component, and enabling the manufacturer of the component to have a small scope for a hardware supply chain risk management (SCRM) program.
- Minimizing the operating system's kernel software to include only device drivers for the specific required hardware devices, and to limit kernel software modules for enhanced security controls.
- Minimizing the operating system's non-kernel software to be limited only to:
  - The software needed to launch and execute the embedded application software of the device, and
  - Software that the embedded application depends upon, notably excluding all background or daemon processing, and network –related services.
- Minimizing the functionality of each embedded application, to perform only the minimally required functions (e.g. data import/export, data validation, ballot interpretation, and vote tally aggregation depending on the purpose of the voting system component), and minimizing the number of existing software packages needed to support these functions.

5.2 System Security and Integrity

The final element of a security-centric architecture is a set of goals for platform-level functions for security and integrity, specifically, the functions required to enable individual voting system components to meet some specific system integrity requirements derived from U.S. federal voting system certification.

In brief, that platform of each voting system component (a single hardware/software device) must be able to ensure that the device runs software that is all and only the software that was tested and certified, without any tampering of or modifications to the software in the certified system configuration. This requirement applies to all the software, from operating system kernel to embedded application software. The breadth of the requirement is one reason for the trusted systems design approach of segmentation and minimization, to reduce the amount of software and the corresponding attack surface for vulnerabilities that could be exploited to modify a component and hence no longer match its certified system configuration.

In addition to platform level tamper detection mechanisms, truly effective system integrity protection requires mechanisms that cannot be bypassed by the injection of malicious software or the abuse of physical access. Current COTS embedded platforms used in VSTs do not meet this high bar, although that bar has been met for some national security systems and some commodity technology (e.g., the hardware and firmware layer of iOS-based devices).

As a result, no voting system in use today can meet these requirements at the platform level, because of the basis in COTS platforms that were never designed to be basis for an embedded system with a fixed software base that must be robust in the face of nation-state adversaries.

A number of distinct technical mechanisms are required to meet these requirements, and the details belong not to voting system architecture, but rather to security architecture of the platform for individual voting system components.

A noteworthy point is that these requirements, and the platform level mechanisms to meet them, are not specific to voting systems. Rather, they are common to many kinds of critical computing systems for which acquisition, deployment, and usage are constrained by a certification and accreditation system. In military computing and other government operated or regulated critical-computing domains, these requirements have been met by usage of existing high-security system technology.

6. Summary

In this paper we've described a next generation voting system architecture based on trusted computing concepts, which differ from those underlying the design of voting systems currently in use in the U.S. at least. This new architecture for voting systems is a pre-requisite for the design of a voting system that is composed of security-critical voting system components, and for the design of each component as a "trusted system."

We submit that this new approach meets or exceeds unmet requirements for the development of critical democracy infrastructure and the protection of our national security, and is centered on the following three design principles:

1. **Interoperability**, with data exchange based on common data standards
2. **Segmentation**, to separate complex systems into separate segments with distinct functions; and
3. **Minimized Architecture**, to reduce the attack surface for "mission-critical" functions.

In addition to defining a voting system based on these principles, this system architecture also specifies design objectives and functional requirements for individual component-level system security and integrity. Further technical details are beyond the scope of this paper, and part of another forthcoming paper addressing security architecture at a component level.

Although this new voting system architecture describes how to make significant improvements in security, integrity, and simplicity compared to current non-security-centric voting systems, significant work remains and is underway at the OSET Institute's TrustTheVote Project.

In addition to component-level security and integrity mechanisms noted in Section 5, both

- **Formal methods**; and
- **Strong cryptography**

…are required for high-assurance voting system components.

Moreover, the use of cryptographic hardware is an opportunity to provide robust protection of the cryptographic operations necessary for both code and data provenance and integrity, including a trusted boot process (device start-up), as well as data security for critical data such as device configurations and vote tallies.

In this regard, we welcome the recently announced[13] work of DARPA in collaboration with one of the OSET Institute's security engineering collaboration partners, Galois. Their work on SSITH[14] to prove firmware and hardware-level verification is essential if not imperative to all sectors of critical infrastructure technology—including devices of voting systems (and to that end Galois will provide some prototypical voting system component demonstrations to prove the value of this necessary advance in trusted computing for critical infrastructure, for which elections technology is now part of one of the 16 sectors). The OSET Institute's work on the software layer is intended to integrate with the work of DARPA and Galois over time.

Hardware aside, focusing on the overall voting system technology architecture described in this paper, on which the TrustTheVote Project ElectOS platform is based, challenges and opportunities remain in the development of specific designs and implementations. And such challenges all rest upon the fundamental principles described in this paper.

The new system architecture aims to minimize attack surfaces where possible, and to provide much-needed flexibility and cost reduction for non-security-critical functions. We submit that providing a balance between security-centric design and the affordance of choice and flexibility is one of the greatest strengths of the new architecture for voting systems.

While there is much discussion (and reasonably so) about the on-going cybersecurity triage necessary to prepare for the 2020 election, we believe the longer arc of required innovation bends in the direction of what has been described in this paper.

This work requires a balance of engineering resources and modest level of philanthropic funding to finish, but in the spirit of our mantra that "*Code Causes Change*" we remain steadfastly confident that its completion will be a very positive disruption in election technology advancement in the digital age of foreign-state attackers, with unlimited resources, and hell-bent on imploding western democracies.

Please join us in this fight for defense of democracies—worldwide.

[13] See: https://www.militaryaerospace.com/articles/2019/03/trusted-computing-cyber-security-hardware-design-tools.html

[14] See: https://www.darpa.mil/attachments/SSITHProposersDay.pdf

About the Authors

E. John Sebes

Mr. Sebes is one of two co-founders and Chief Technology Officer ("CTO") for the U.S. based OSET Institute. He leads all aspects of technology strategy, vision, architecture, engineering and development for the TrustTheVote Project – the flagship effort of the Institute.

Prior to that, John has been a software developer, technical consultant, and CTO, working in several areas, including network infrastructure, application frameworks, embedded systems, critical infrastructure, data center operations, with strong common themes of risk management, security, privacy, and reliability. Innovation and technology transfer have been another consistent theme, in settings as varied as government-funded R&D, venture-backed start-ups, professional services, academia, and non-profits.

For parts of his career, John provided independent consulting services related to information security and IT operations assurance, for a variety of organizations ranging from technology start-ups and venture capital firms to major government agencies and established financial services firms. At other times, John has been a Principal Investigator in R&D projects, ranging from DARPA projects performed in the pre-Web era, to recent work with DHS on public (open source) security technology.

He has been working in the non-profit world with a focus on election technology for over a decade, partly from a desire to do public service with his professional skills, and partly because it is a surprisingly good fit for several seemingly disparate parts of John's work history and interests. John's passion for defending democracy is based on a rich history as a 1st generation American, with a Hungarian family history rooted in escaping Nazi-occupied Europe.

Previously CTO at Solidcore Systems, Inc.; VP Strategy at Securify; Technology Officer of Network Associates Labs; and variety of consulting, development, and R&D management roles at commercial InfoSec pioneer Trusted Information Systems.

John is a co-author of 12 patents and 20+ publications and a graduate of Yale.

Edward Perez

Mr. Perez is formerly director of product development for a major commercial voting system vendor. After retiring from the commercial sector, he joined the nonpartisan nonprofit OSET Institute as Global Director of Technology Development.

He holds degrees in Government and Political Science from Georgetown University and the University of California at Berkeley and has over 16 years direct experience in the design, development, delivery, deployment and servicing of commercial voting systems.

In his capacity as a member of senior leadership and reporting into the Office of CTO, "Eddie" is responsible for all technology development management (similar to product management in the commercial world), as well as leading efforts in open standards development contributions to the U.S. EAC and other international standards groups as appropriate. Mr. Perez is also focusing on models of certification, formal methods, peer-review assessment, and advocacy work for public (open source) technology. In addition, Eddie is working with the Institute's Technology Public Policy team on development of public policy research and work with state and federal governments and the media as a subject matter expert.

Acknowledgements

By Principal Author, E. John Sebes

This content of this paper is the foundation of our work at the OSET Institute and TrustTheVote Project. The essential concepts behind the architecture presented here are, to me, rooted in classical trusted systems concepts from the old Orange Book, [15] applied to embedded systems in critical infrastructure computing, and the special sub-segment of election critical infrastructure. As a result, I feel very thankful for the education and inspiration that I've received from many people over the years:

- My compatriots in 1990's trusted systems work, including Terry Benzel (especially her work in embedded system-build security); Hilarie Orman; Sue Rho; Rich Feiertag; Curt Barker; David Balenson; Jeremy Epstein; many other Former Trusted Information Systems company [16] folks; and Peter G. Neumann — several of those who led me into the critical infrastructure cybersecurity and homeland cybersecurity worlds;
- Taher Elgamal and other colleagues at former Securify [17] as well as at former Solidcore Systems,[18] with whom I worked on early system-hardening technology, especially Douglas Maughan of DHS S&T whose SBIR funding and guidance to Solidcore Systems was extremely helpful;
- A host of cybersecurity and election technology experts both in the Election Verification Network Society and NIST's public working groups, including, but certainly not limited to, David Jefferson; Barbara Simons; David Wagner; Dan Wallach; Matt Blaze; Harri Hursti; Joe Kiniry; and especially Jeremy and PGN for following a similar trajectory into election cybersecurity and providing many years of encouragement and constructive critical support.

Finally, I add a special thanks to past and present OSET Institute teammates especially Pito Salas; Aleks Totic; Stafford Ward; Hilarie Orman; Clifford Wulfman; Brendan Hemingway; my supporting author here and Director of Technology Development, Eddie Perez; and several others involved in software development of the systems described in this architecture.

[15] The Orange Book is the nickname of the Defense Department's *Trusted Computer System Evaluation Criteria*, a book published in 1985. The Orange book specified criteria for rating the security of different security systems, specifically for use in the government procurement process. For more see: https://en.wikipedia.org/wiki/Trusted_Computer_System_Evaluation_Criteria

[16] Trusted Information Systems (TIS) was a computer security research and development company during the 1980s and 1990s, performing computer and communications (information) security research for organizations such as NSA, DARPA, ARL, AFRL, SPAWAR, and others. McAfee acquired TIS in 1998.

[17] Securify, Inc. was a computer security company during the late ‘90s, acquired by Secure Computing Corporation, and now part of McAfee.

[18] Solidcore Systems, Inc. was a software company based in Cupertino, California, that developed software to detect and prevent unwanted change. McAfee acquired Solidcore Systems in 2009.