OSET INSTITUTE
BRIEFING

Version 6

Practical Realities of Open-Source Technology in Election & Voting Systems

Prepared For:
States Election Administration Leadership

Prepared By:
E. John Sebes, CTO
Co-Founder

Deborah Scroggin, MPA
OSET Institute Strategic Advisor; Former Oregon Director of Elections

Dr. Clifford Wulfman, MSCS, Ph.D
Sr. Member of OSET Institute Technical Staff

February 2023

Abstract

There is growing discussion with national (*and even international*) visibility on the issue of opacity in election technology and the need for more transparency across equipment and processes. This is rapidly leading to legislation to mandate open-source software in voting systems. While this is an exciting prospect to increase confidence in elections and their outcomes, the Institute believes it's important to not get the proverbial cart before the horse. First, there is still considerable misconceptions about open-source and what it can and cannot do to help the transparency issue. There are even concerns about the role of open-source technology in the quality, integrity, and security of election technology. Perhaps more importantly, regarding the cart and the horse, there are no federally or state certified complete voting systems today based on open-source technology. There are, however, companies and Projects that do have certified components. So, legislation for entire voting systems may be a bit early to have an enforceable impact. However, there are other aspects of election administration technology that can more immediately benefit from open-source initiatives and mandates. This Briefing is intended to address all of this, and offer a modest four-part pathway forward to achieving more transparency in election technology infrastructure through open-source software.

© 2023 OSET Institute, Inc. All rights reserved.

Table of Contents

Context

The OSET Institute was founded on the idea that election technology infrastructure is a critical component of government IT; so critical in fact, that it should arguably be a public asset on which the commercial industry (*or election organizations themselves if properly resourced*) can build and deliver finished open standards, open data, and, accordingly, open-source based systems.

Historically inherent in our name, OSET ("*Oh-Set*") is a pair of words, "open" and "source." We have always maintained that open source is neither necessary nor sufficient for higher integrity, lower cost, easier to use election administration systems. However, publicly available technology (*i.e., open-source*) is an important ingredient to ensuring transparency and trust in the technology. In the 16-years since the Institute's founding, "open-source" as a phrase used in conjunction with voting systems has grown to be a provocative and in some limited situations, controversial topic. *It should not be.*

First, and foremost open-source does *not* mean "free source." Some will argue that it is free because there is no charge for the software itself. We acknowledge that but observe that free is more accurately "free" as in a "*free puppy*." The notion of cost is important when examining the realities of open-source in election and voting systems.

Open-source primarily addresses transparency, as the phrase's word-elements imply. It is equally important to understand that open-source refers to <u>*both*</u> a *process of development* and a *means of distribution*.

Applying open-source technology to elections infrastructure, the objective is elections whose processes are Verifiable, Accurate, Secure, and Transparent (*a principle called the "VAST mandate*.") If voting technology can be developed transparently, and made available in an unencumbered manner, with incentive to continually innovate while taking care to rapidly identify errors, flaws, and vulnerabilities, then there is a higher probability for public elections achieving the "VAST" mandate.

Recently, conversation about the necessity for transparency and thus open-source in election infrastructure has dramatically increased, to the point that in at least one state there is legislation to mandate open-source as a requirement for certified voting systems. The OSET Institute has been focused on open-source applicability for over 16-years. And we submit that while legislative intentions are laudable, the call for such action right now may be getting the proverbial cart in front of the horse. Yet there are steps that can be taken to move in the direction of open-source to increase trust in election infrastructure, as this paper explains.

First, we believe it is essential to understand what exactly open-source technology is and is not; can and cannot do; and the appropriate uses of open-source in mission-critical government computing, particularly election administration, which has become a matter of national security. With that groundwork, we can then examine the practical challenges of open-source mandates, and consider a roadmap for how to best get there. That is the objective of this Briefing, and we hope this paper is helpful to the reader's interest and pursuit of innovation in this vital aspect of democracy administration.

1. The Need for Public Technology in Election Infrastructure

Trust in today's voting systems is on the decline because they are closed, proprietary systems — some call them "*black box*" for their opacity.

These systems have inherent system-level design vulnerabilities; that is, given today's critical infrastructure standards, the overall system architecture as it exists in deployed systems today is inadequate to address known security issues. That was not always the case; requirements have significantly changed and "*attack vectors*"[1] have dramatically increased since these systems we're envisioned in 2002 to 2005.

Assuming well executed custodianship and provenance of the equipment, these vulnerabilities are difficult to exploit. However, post 2020 we witnessed even that last perimeter of security — *physical custodianship and provenance* — compromised. (*See footnote 6 on page 10.*)

> With this being the state of the infrastructure, lies about the security and integrity are easily enabled by current voting systems design.

And this is exacerbated by the fact that current designs suffer both *opacity* and *obsolescence*. This perpetuates a presence of conspiracy theories that fuel the distrust. Until such time as election technology infrastructure is based on public software technology (*i.e., open-source*) that chaos will continue.

Thus, principal to this Briefing is the proposition that open-source technology is imperative to increase confidence in elections and their outcomes. Yet, today no such federally or state-certified open source-based *complete* election administration or voting systems exist.[2] We need a pathway forward for that to change.

2. Open-Source as a High-Quality Solution

The term "open-source" can be used and abused in ways that confound fact-based technology policy discussions about government procurement of critical software. Such confounded discussions are a real problem, because Open Source Software ("OSS") is already a nearly inevitable part of the foundation of all government-critical software. It also turns out that open-source is also a useful method for filling technology gaps with publicly available software for which there is no profitable commercial vehicle for its creation and/or support.

A (Very) Brief History on the Rise of Open-Source Software Projects

Open-source refers to both the *development* and *maintenance* of software as well as its *licensing* and *distribution*. Historically, this applied to software that was community-developed, with said communities largely, if not entirely, volunteer. Many of today's most significant on-going open-

[1] An **attack vector** is a method of gaining unauthorized access to a network or computer system. An **attack surface** is the total number of attack-vectors an attacker can use to manipulate a network or computer system or extract data. For discussing attack vectors, a voting system is the combination of several computing devices; today those devices are typically machines using a version of the Microsoft Windows operating system.

[2] By "complete" we mean an entire voting system from ballot casting, to printing, tally, tabulation, and its' election management system (e.g., all equipment). However, one company and one open-source Project do offer certified voting system components. **Voting.Works** and the **RCTab Project** offer state-certified tabulation components.

source software projects have their roots in universities, funded by government research grants. Some of these projects were spun-out of their academic incubators and non-profit or commercial corporations assumed their on-going development and/or curation. In other cases, as certain open-source projects became more widely used in other software, or became more mission-critical, organizations were formed and funded to ensure the availability and quality of the software and these organizations turned to corporations — many who often incorporated the software into their own products — to donate funding to provide sustainability for the independent open-source project.

Persistent Misconceptions About Open-Source Quality

Even understanding how open-source projects have proliferated and are managed today, there remains plenty of misconceptions about the quality, integrity, and security of open-source software as it is often or "classically" developed. Among the common misconceptions about open-source software are the following.

1. **OSS is communally developed without controls.**
   Reality: While OSS is frequently developed by highly distributed networks of contributors, integration of new code and revisions is carefully controlled by project managers using commonplace techniques employed by software-development organizations of all types. It is often overlooked that commercial technology companies regularly develop and participate in the maintenance of software that is freely available under an open-source license.
2. **OSS can be modified by anyone.**
   Reality: While many OSS projects use public software repositories that allow code to be duplicated and reused, the production repository remains entirely under the control of its custodians. New software is only incorporated pursuant to acceptance protocols.
3. **OSS depends on free donations from unvetted volunteers.**
   Reality: Repository custodians have complete control over what, if any, contributions are incorporated into the code base. Some projects have strict contributor vetting.
4. **OSS is free.**
   Reality: In many government-computing situations, *free* simply means that the contractor will not charge license fees for non-proprietary software. The term does not have ideological connotations. "Free" in OSS means more like free in "free puppy."
5. **OSS is haphazardly dependent on externally developed libraries and components.**
   Reality: Code adoption and reuse can happen on a spectrum from "disciplined minimality" to sheer "expediency for speed of delivery," depending on the choices of a project's leaders, and the team's effectiveness in executing on those choices.
6. **OSS is less secure and reliable than proprietary software.**
   Reality: The so-called "security-by-obscurity" model of software development was debunked long ago, and cyber-security experience has shown that source code disclosure does <u>not</u> advantage adversaries. (*See footnote 6, page 10*.) At the same time, while making source code available for public inspection increases the probability that security flaws will be discovered and repaired, it does not guarantee it, and does not obviate the

need for certification and accreditation (C&A) practices. Experience has shown that the technology community, both commercial and noncommercial, is able to provide resources for custodianship when it is evident that a body of software is truly critical.

Just as open-source software methods (*e.g., development, distribution, licensing, technology-transfer, commercial integration and support*) are not an intrinsic barrier to effective creation of government-critical software, neither are these methods a panacea or a guarantee of success.

A non-proprietary or open-source software organization must still set appropriate objectives for the adopters' needs, and successfully execute on those objectives during initial development. Thereafter, they must support successful C&A activities, and provide organizational continuity for ongoing custodianship of the software. These requirements for success are serious challenges in a government technology market that has been unable to support profitable commercial activity to develop needed solutions. (*Which is why the OSET Institute is pursuing such.*)

2.1. Critical Computing and Open-Source Software

Many do not realize this, but in the U.S., almost all government computing relies on open-source software (OSS) to a measurable degree. Perhaps the largest portion of U.S. government computing — homeland security, defense, and intelligence computing — is also one of the largest areas of computing anywhere to explicitly and strategically adopt OSS. This area of "*government-critical computing*" is an expedient adopter of non-proprietary or open-source software technology that meets a specific need without the encumbrances of commercial software acquisition.

More than a lead adopter, U.S. government-critical computing is one of the largest drivers of the creation of public (*software*) technology: that is, software developed under government contracts, with taxpayer funds, resulting in publicly-owned technology. In many situations, making such publicly owned technology available is accomplished by open-source licensing and distribution methods.

There is much more to explain about the role of open-source in government computing, and for that we refer the reader to another of the Institute's papers published five years ago to the date of this paper. It is full of content that is still completely applicable, relevant, and timely for this discussion today. That paper is entitled, "*The Appropriate Use of Open-Source Technology in Government Mission Critical Computing*"[3] and we have borrowed from that paper (*authored by two of the same authors here*) in this Briefing. We recommend it as a companion to this paper for those readers most directly charged with shaping technology policy development.

3. The Case for Open-Source in Election Technology Infrastructure

Beyond the federal government and three-letter agencies, all kinds of critical government computing systems also, in fact, use open-source software (OSS). This is important because OSS is an essential part of these critical systems, whether a computer operating system or simply cryptography software.

[3] https://www.osetinstitute.org/research/2018/3/12/appropriate-use-oss

Here is a key point: certified voting systems today already incorporate open-source software. The software components used to protect the transfer of election data between devices are, in fact, open-source. This is discussed a bit more in Section 4.3.

Indeed, today just about everything has open-source software as an ingredient, from refrigerators, to smart energy meters, to entire power grid controllers; literally all over, and that includes critical computing systems. So, why not election administration systems too?

Given that election administration and voting systems are not just tantamount to, they are actually designated as critical infrastructure, they qualify for, and should be compelled to adhere to design and engineering standards for all critical government computing. We have set forth the reality that critical government computing broadly adopts, adapts, and deploys OSS in their systems. And they do so, adhering to development practices according to standards used in the military and national security apparatus.

> Therefore, not only should election technology be subject to the same practices and standards, election technology should be based on—not just incorporate components of—open-source software.

In fact, there are five important reasons for open-source in election technology infrastructure.

1. Verifiability—Open-source software lends itself to proving that the technology is operating as intended; rather than trusting what the vendor has built, one can believe for themselves what has been built, and how it operates.

2. Accuracy—Open-source lends itself to readily easy analysis and testing, because the technology is open to anyone's assessment tools and analytics.

3. Security—Open-source is neither more secure or less secure; the very same developers using the very same skills (*and making the very same mistakes*) can write software regardless of its licensing or distribution models (*open/public at no charge or closed/proprietary for a fee*). However, the clear distinction is the "glass box" nature of open-source versus the "black box" nature of closed-source, because one cannot hide a trap-door in a glass box. In other words, bugs, malware, design or implementation vulnerabilities are readily discoverable in open-source. It often said by the thought-leaders of open-source: "*When thousands of eyes are upon it, all bugs become obvious*." Given the required trust (*and belief*) of election infrastructure as such systems are used to administer our democracy, public technology for proof of security is imperative.

4. Transparency—Open-source is the very best way to ensure belief in our election technology infrastructure. Transparency builds trust and belief. Transparency in all aspects of election administration is essential, and that is mandatory for the technology.

5. Cost—Open-source should result in a dramatically lower total-cost-of acquisition of election administration and voting technology. With open-source, the cost of the software technology layer is essentially free (*for the actual software itself*). We opened this Briefing by suggesting that "free" as the term applies here is more akin to the notion of a free puppy. In other words, there remain costs for local adaptation, integration,

deployment, service, support, etc. In fact, the commercial and enterprise open-source technology business is huge. Consider that IBM ultimately purchased the open-source software company RedHat for $34B USD in 2019.[4] Regardless, if voting systems were using an open-source software technology layer this should (*we believe would*[5]) reduce the cost of voting system acquisition.

Thus, we find the case for using open-source is effectively open and shut — its already broadly happening across all sectors of critical government computing, and within voting systems too. The questions now are, "*How much of a voting or election administration system should be open-source*?" and "*How do we get there*?"

4. The Practical Challenges of Evolving Election Technology Toward OSS

We appreciate increasing legislative efforts to require more transparency in critical election technology infrastructure. We've long observed that *trust* is what black-box proprietary systems *demand*, while *belief* is what glass-box public systems *deliver*. However, it is insufficient (*and problematic*) to craft legislation demanding that voting systems utilize open-source (*software*) technology when no such certified systems yet exist. Yet, there are ways to advance this laudable goal. In crafting any policy about election technology transparency, it is important to have a working understanding of three (3) major challenges, which we address below in Section 4.3. Before we get to that we should understand the impact of technology licensing starting with open-source as intellectual (public) property.

4.1. The Nature of Open Source as Intellectual Property

Open-source software is software with source code that anyone can inspect, modify, and enhance. Source code is the part of software that most computer users never see; it's the code (*like a recipe to bake a cake*) that software developers write and manipulate to change how a piece of software — an "App"— operates. Developers who have access to a software application's source code can improve that App by adding features, or repairing or modifying parts that are not properly functioning. For voting systems, open source promises the greatest degree of trust (*and belief*) because the underlying software is available — from the start — for public access, inspection, and modification (*subject to processes and procedures for incorporating said modifications into the production version*). However, from the perspective of the OSET Institute, open-source is far more than the specific applications source code or recipe.

[4] https://www.redhat.com/en/about/press-releases/ibm-closes-landmark-acquisition-red-hat-34-billion-defines-open-hybrid-cloud-future

[5] The OSET Institute has studied costs of election technology since its founding in 2006. Working with consultancies and accounting firms such as Accenture, we've determined that if there was a software public technology layer for election administration and voting systems that was broadly adopted by vendors through customer requests via their procurement processes, the cost of voting technology to taxpayers would substantially decrease. Pro-forma business modeling suggests that could be by as much as 60% over today's costs. In fact, the Institute, through its TrustTheVote® Project has built prototypes of several next-generation components of election technology demonstrating that far higher-quality machinery could be produced for roughly one-third of today's costs.

For the technology to be genuinely "open-source," we strongly assert that *all* elements comprising the process of design, development, and production of the software must equally be published and available alongside the software source code itself. This includes public and technical specifications; algorithm designs; software user interface designs; design and engineering reviews; software testing and code audit results; independent test labs evaluation and testing; compilation (build) instructions; of course, all technical and user documentation and revision histories; government certification and field tests (*ensuring that fielded systems have the certified version of the software*); and finally, easily accessible public disclosure of the entire process. It's actually quite a lot of work to make election administration software truly open-source.

4.2 Alternative Legacy Licensing Schemes That Do Not Work

The following section may be of interest to those more involved with or responsible for technology acquisition and licensing; feel free to disregard otherwise.

Of course, open source is subject to an open-source license, which specifies the terms and conditions for how a licensee may use and further distribute the software on a royalty-free basis. The hallmark principle of open-source licensing is the perpetual harvest and that means any modifications, enhancements, or repairs made to the software must be contributed back to the Project for all to benefit from. In this way, a rising tide raises all ships (*assuming all licensees adopt, adapt and deploy every improvement made by others*).

In this way, open-source is the most permissive and easiest low-cost way to acquire (*but not including cost to maintain*) software. Depending on one's skill additional (*commercial*) resources to assist in its use may be required. Yet, that is in stark contrast to alternative mechanisms one might attempt to rely on in order to satisfy their concerns about the nature (*correctness, integrity, quality or security*) of software source code, which requires grappling with more restrictive licensing schemes. Here are the other two principal licensing schemes that may provide access to the licensed source code. Candidly, one is next to impossible to achieve the desired access; the other fails to meet the goals of transparency.

1. Proprietary—Closed Source. The software is subject to protection under trade secret law. It may well also be subject to patent protection, but that gives rise to some disclosure issues beyond the scope of this paper, and not helpful to understanding the state of software ownership as it relates to achieving transparency. The bottom line to closed source is that pursuant to the software right-to-use license, disclosure is impossible outside of some special negotiated arrangement between the owner-vendor and the customer-licensee (*e.g., a "source-code escrow"*). Another such method of access is contained in the next type of legal structure.

2. Proprietary—Disclosed Source. The software, while subject to legal protections for unauthorized redistribution, use or modification, can be made available for purposes of inspection, assessment and peer-review, audit, or other purposes. Disclosed source does *not* suggest it is freely available for use, redistribution, or deriving other works. Some argue that this is a fine solution to the transparency issue. We disagree for a variety of reasons, but not the least of which is that there is no agreed-to standard or uniform process and result of source code disclosure. That depends on how much disclosure the

author is willing to allow to achieve a given level of trust satisfaction. The challenge in the case of voting systems is that requiring disclosure at this stage is likely unhelpful.

We believe it is not a good policy to force the existing voting system vendors to disclose their source code — at least not if the desired benefit is to *increase* public confidence. These systems are already in a hole of mistrust, and disclosure probably won't help with public confidence — much less government-mandated forced disclosure, which has some serious downside as well. For disclosed source to achieve its intended objective of increasing trust, source code publication (*disclosure*) should begin at the birth of the product. So, let's pause to offer a well-known (*although dated*) example of successful source code disclosure (*vs. open source*).

PGP, Inc.,[6] back in the day, was a commercial company in the business of selling proprietary enterprise security software including the very well-regarded PGP ("*Pretty Good Privacy*") software for eMail and file encryption. From the start, trust was a critical issue for PGP, because in some cases people's lives literally depended on the software operating correctly. Thus, from the beginning, PGP, Inc.'s software was *disclosed-source*, and based on earlier open-source (*non-proprietary*) PGP implementations. The past and continuing ability of experts to read the code helped to build and maintain the required trust in its performance.

It is possible that the same benefits could accrue to vendors of proprietary voting systems that disclose their source code. If the source code is disclosed from the very first release of a product version, and can be read, built (*compiled to produce the running applications*), and tested without restriction, then such a voting systems vendor would have a good starting point for public trust. *Doing so from the start is important.*

> Once a closed proprietary system comes under suspicion, the vendor is in a mistrust hole; and disclosed-source will not help to climb out of that hole.

Thus, it is unclear at best, whether required disclosure is a great idea. There may be some benefits in doing so; however, there are drawbacks, starting with the likelihood that some critics or other mendacious actors will raise claims of previously undisclosed software problems – with lots of arguing over whether those claims are valid. Some will be valid, some won't, but the din will further erode confidence without actually making the products any better or instilling any more trust. Similarly, we believe there would be claims that "customized attacks" on the software would be enabled, with arguments against such claims – again with public confusion. To be sure, we've written previously about the dangers of unauthorized disclosures to this extent.[7] To be sure, we're talking about the effects on perception here, not prognosticating about "facts" that might be revealed. If more than 4-decades of software product management for several of the

[6] PGP Inc. was co-founded in June 2002 by Jon Callas and Phil Dunkelberger, based in Menlo Park, CA. — adjacent to the OSET Institute's home in Palo Alto, CA. The company owned the Pretty Good Privacy (PGP) software, which was originally developed by Phil Zimmermann (who had earlier started a company of a similar name). Originally written in 1991, PGP was one of the first freely and publicly available implementations of public-key cryptography. PGP Corporation was sold in 2010 to Symantec and subsequently dealt to Broadcom in 2019.

[7] See: https://www.osetinstitute.org/research/2022/1407/votingsys-disclosures/

Institute's leadership has taught anything, we know perception is part of confidence and perception is 90% of the perceiver's reality.

4.3. Existing Voting Systems Issues

Proprietary Systems Already Use Open Source

The very first point to recognize is that all voting systems already incorporate open-source technology. This is mentioned earlier in this Briefing, and is important because the initial arguments raised by those less informed about the benefits and case for open-source is that there is no precedent for open source, and open-source poses a security issue (*neither of which, are true*).[8] However, as we've explained, open-source technology is pervasively present throughout government technology, and voting systems are no exception.

In voting systems, vendors incorporate open-source encryption software[9] (*at least the algorithms*[10]) to handle basic data hygiene tasks. For instance, the data incoming from the election management system (EMS) must be authenticated and verified as untampered, and the same is true for tally data incoming from counting devices back to the EMS. This open-source software is also used (*in older voting systems*) to produce verifiable digital signatures and data encryption for these machine-to-machine transactions. The use of open-source software for these crypto-duties is easier, more trusted (*because the source-code is available to verify how the software libraries work*), and of course, free is very cost-effective.

Obviously, closed, proprietary, black-box voting systems containing open-source crypto management software does not make the system open-source by any tortured sense of the exaggeration. The point is, however, there is precedent for open-source.

The Inherent Design Challenges

Before we can consider what is required to produce an open-source voting system we need to also understand the existing issues with current voting systems that are "blocking agents" to progressing to better, public technology. We have long held our system architecture views for what would constitute a more verifiable, accurate, secure, and transparent voting system. And those principles drove our design of ElectOS™—a voting system we're engineering and developing with public technology at its core, using off-the-shelf, slightly modified commodity hardware. Those principles, contained in a white paper originally drafted in 2010 and updated in 2019, are worth considering.[11] Accordingly, there is far more to achieving a truly trustworthy voting system than legislating the presence of open-source software. If trust is to truly be achieved, then more than code alone, the design and

[8] See this short, cogent explanation of why open-source presents no greater opportunity for compromise than closed source: https://twitter.com/OSET/status/1626383987274031108

[9] https://www.openssl.org/

[10] OpenSSL contains an open-source implementation of the SSL and TLS protocols used primarily on the World Wide Web for encrypted transactions. The core software library implements basic well-understood cryptographic functions and provides various utilities. https://en.wikipedia.org/wiki/OpenSSL#Algorithms Software wrappers—a type of delivery container—allow the use of the OpenSSL library in other software applications, such as those in voting systems.

[11] https://trustthevote.org/wp-content/uploads/2019/04/01Apr19_VotingSystemNewArch_v4Final1.pdf

engineering processes for next-generation voting systems are equally important. So, to get there we first need to acknowledge the issues with existing systems design.

Here are three major inherent design flaws (*there are more*) that exist in today's voting systems that must be addressed to achieve a truly verifiable, accurate, secure, and transparent system, wherein public availability of the software source code alone is insufficient:

1. Today's voting systems depend on an antique desktop computing model.
   Today's systems are essentially software applications running on more-or-less standard Microsoft® Windows® operating systems (*whether desktop or embedded versions*), and in many cases, this means outdated versions of Windows. The flaw is that voting system devices should be purpose-built strictly for their role in the system and contain only the software minimally necessary to serve its intended purpose—from the very kernel of the operating system through the top-end of the user-experience. Running an "App" on top of a Windows-based system introduces all the vulnerabilities characteristic of that desktop operating system.

2. Today's voting systems are typically comprised of multi-purpose computing devices.
   In today's multi-purpose systems, generally, software can be added or removed from a device (*which is often a desktop, laptop or tablet computer*). The flaw is that it is unsafe and against critical computing principals to allow mission-critical apps to reside in common tenancy with other apps—especially apps of unknown origin, or at-risk of malware. Voting systems should be comprised *only* of single-purpose devices.

   For instance (*and we have witnessed this*), video games can be run on the *same* machine where a voting system App (*for instance the EMS*) operates. Seriously: we have seen systems supposedly "locked down" for use as an EMS device containing video arcade games with code produced by adversarial nation state developers that incorporated malware that could extract data from other applications! In fact, the device should never support running any apps other than the intended mission-critical app for its designated role in the voting system.

3. Today's voting systems have no traceability on their development history.
   Today's systems typically depend on lazy logic and accuracy testing for the base software despite the systems in production regularly going through system-level quality assurance testing. The flaw is that there is no evidence that security-centric engineering principles and software development practices are employed in the production of current certified voting systems.

   It is worth noting that the current U.S. Election Assistance Commission's Voluntary Voting System Guidelines (VVSG) does require Voting Systems Test Laboratories (VSTL) to review software quality assurance (SQA) plans and results. However, this is apparently not enforced, given what independent reviews by security experts have shown: generally, low quality code and especially with regard to digital security norms.

The bottom-line is that while we fully and wholeheartedly endorse open-source software in all election technology infrastructure, we believe that until there are better systems designed and engineered, using open-source technology, mandating open-source *alone* will not achieve the goal of increased trust, belief, and confidence in elections and their outcomes. There is more, much more that needs to be done in concert with such mandates—starting with how systems are designed and developed.

However, we still believe there is a pathway to getting there.

4.4. Achieving a Certified Open-Source Based System

The following section may be of interest to those more deeply involved with or responsible for election technology policy strategy; feel free to disregard otherwise.

First (*at initial impression*), the dimmer view. The challenge, simply put, is that there is no commercial incentive whatsoever in the industry, as the market is structured today, to build and sell a complete voting system based on public technology that the vendor would develop on their own. Just as there is no incentive whatsoever to even publish their software as *disclosed source*, an open-source system built on their own accord (*and we argue not even if mandated*) is highly unlikely without significant change in the marketplace and a reformation of the commercial business of sale, delivery and deployment.

Second (*at closer-look*), the brighter view. There is clear opportunity (*and requirement*) to design, develop, produce, and make freely available the underlying software technology to power the next generation of voting systems. In fairness of full disclosure, that is the core of our Institute's 501(c)(3) mission—as well as the mission of several other initiatives. Let's set our work aside and focus on the opportunity and potential.

If the market were flooded with this public (*open-source*) technology and freely available (*like OpenSSL crypto libraries discussed above*), then the existing vendors could take advantage of that technology, build their systems upon that technology and commercially deliver, deploy, service, and support voting systems—*just as they always have*.

The quid pro quo is simple: the heavy-lifting of the required research and development to produce that open-source software is removed from the vendors' shoulders (*they save millions of R&D dollars that, given the market dynamics, they might never recover*), and in exchange, their business model adjusts and evolves: their business no longer includes proprietary licensed software, and a good deal of that portion of revenue shifts to the required adaptation of the software to the needs of their customer-jurisdictions, and its subsequent deployment, service and support (*the balance of savings should be reflected in lower pricing of the systems — and a more competitive market will ensure that; also see footnote 5 on page 8*).

In other words, the vendors incorporate this public technology with off-the-shelf hardware to deliver finished systems. This is called a *systems-integration* business, and that business model flourishes today throughout government I.T. sectors. Of course, this evolved business model would open the market to new competition for any systems integrator to enter given the availability of the foundational software layer. That means a more competitive market and better results for all—voters, election administrators, and vendors. And that is our *theory-of-change*.[12] So, what does it take?

[12] Setting aside the OSET Institute's mission, see: https://www.osetinstitute.org/our-theory-of-change

Such an approach is a significant undertaking to be sure, but it is a tractable project requiring a couple of years of intense effort, and for a fraction of the amount of money the federal government has doled out year-over-year for election security. Indeed, the remaining costs in our own program at the Institute to produce ElectOS™ is *well less* than $10M for the voting system and less than $30M for the entire software ecosystem across the entirety of election administration. Consider how much money is invested in campaign cycles and how much has been spent in election security funding from year-to-year. This is a bargain. But again, set aside the work of our Institute. Let's consider other approaches.

1. One way this could easily be done is to press Congress to set aside a one-time fund to be administered by (*for example*) the National Science Foundation to award grants to develop this public technology as a public asset like so many of the critical infrastructures we depend on today. Such a program to develop the public technology might require at most $100M of program funding. That's just one eighth (1/8th) of the all the election security funding granted by Congress over the past three election cycles.
2. Let's ideate even further. Assume (*and the case can be qualitatively made*) that the cost to produce a public voting system technology layer runs in the range of $10M—$15M. What if states began to collaborate, apportion the cost amongst themselves and produce the technology they would permanently own, and then hire industry to adapt, deliver, and deploy? How much lower could the cost drop on a per-state basis? You see the idea.

There *are* ways to get this done—the production of public technology (*open-source*) for all voting systems and vendors to adopt, adapt, and deploy. Yet, it will take time and "*thinking different*" about funding.[13] And while that potential is progressing, there is something that can be done today—*right now*—to start the process toward a completely transparent election technology infrastructure.

5. A Pathway Forward

There is no doubt that achieving a public voting system—one based on open-source technology—has a journey of hurdles ahead; yet it is tractable, it is do-able, and it requires some creative thinking as outlined above. In the meantime, the very good news is that there are initiatives—steps that jurisdictions can take now, today, to start building more transparency into America's election administration technology infrastructure at every state and county level. And it does *not* require waiting for someone or something to produce a finished voting system based on open-source software.

Crawl, Walk, then Run Toward Belief

Today, there are many aspects of election administration wherein technology is necessary that requires trust in how it operates, but does not necessarily require the complexities of state or federal certification. And these technologies are typically custom or semi-custom built by

[13] Meanwhile, the OSET Institute will press forward on its own initiative that is less than 2-years from completion. At that point, ElectOS certification can be pursued in at least one jurisdiction and then we can flood the market with this free technology to catalyze the breakthrough improvement our theory-of-changes argues for. See: https://www.osetinstitute.org/our-theory-of-change and contact us to discuss this change-agency more.

vendors already. Here is just a sampling of the sub-systems of election administration that can immediately benefit from open-source technology to achieve a far greater level of verifiability, accuracy, security, and above all, transparency:

1. Voter registration systems and services
2. Election results reporting systems and services
3. Digital Pollbooks
4. Election management systems (EMS)

A Modest Proposal

We propose a four-step process to fostering more transparency in election infrastructure.

1. Explore open-source opportunities in all aspects of election administration. States and/or counties (*where authority resides*) should immediately start investigating where and how open-source technology can be used to upgrade or replace existing election administration Apps and services. This can immediately begin to restore trust in the infrastructure. Start small and gradually progress. There are already several open-source software components out there today that can be adopted, adapted, and deployed.

2. Consider how to modernize and open-up the procurement processes including regulations that allow leveraging of smaller technology ventures and initiatives. For example: these innovators are typically blocked from offering their capabilities simply because they lack the financial stature to compete in the procurement and bidding process. More often they are sidelined by strict requirements of having a pre-existing installed-base of "reference customers" that perpetuates the existing (*sometimes obsolete*) solution status-quo (*making it very attractive for incumbents but perpetuating existing design flaws or missing out on opportunities for better technology*). This ultimately prevents new technologies from gaining an equal toe-hold of opportunity. That needs to change to obtain a higher-caliber of mission-critical election technology. Similarly, adding a preference for open-source software (OSS) in all RFPs and RFQs will open opportunities for vendors and jurisdictions alike.

3. Prioritize pilot programs that allow new technologies to be tried, tested, and proven. Fund those programs to award pilot grants to technology makers to prove the capabilities of their innovations. In some cases, states may be able to first acquire election innovation and security grants from the U.S. EAC, which they can, in turn, invest in these pilot programs. These pilot programs can also serve as the basis for establishing requirements and specifications for next generation solutions and help develop the requests for proposals and quotes.

4. Start with open-source mandates in non-voting aspects of election administration. There are initiatives that can have immediate impact.   Legislating open-source mandates on the voting system vendors today, as the market exists now, unfortunately, will not net the desired impact. So, instead, consider the four areas of opportunity itemized just above, starting with central voter registration systems.

6. Summary

This paper has presented the case for public technology (*open-source*) in election administration. There are plenty of misconceptions about open-source, beginning with a misunderstanding of its existing widespread use in government I.T. systems. We provided some background on the different intellectual property structures that protect, while making available the technology. We explained the advantages and disadvantages of disclosed source versus open-source. We walked through just some of the challenges facing today's voting system. There is plenty to consider in achieving the goals of system transparency and accordingly, trust.

We clarified a half-dozen misunderstandings about open-source developed and distributed software technology. An important observation is that open-source technology is now an essential part of all computing. Defense and national security-critical systems use non-proprietary (*open-source*) software as *foundational* technology. Public, non-proprietary (*open-source*) software also has an important role to play in filling technology gaps for which the commercial market lacks an incentive to fill. Such is the opportunity with election technology.

There is a path to achieving an open-source voting system that is likely to catalyze adoption, adaptation, and deployment by the existing industry, although it will take some creative thinking to determine the best way to fund and finish the required work (*setting aside our own initiative in this regard*).

However, the most important point of this briefing is in its final section. We observe that while laudable, legislating open-source in voting systems, absent the existence of any such system will not yield desired results. However, there is a way to get started: begin requiring open-source technology in systems of election administration that are not dependent on certification. A prime example is *central voter registration systems and services*. There are others—from election results reporting to digital pollbooks.

Use a crawl-walk-run approach to increase transparency in election infrastructure, by first requiring open-source in other systems of election administration. Next, progress to innovating procurement processes and institute pilots of new technologies. Finally, legislate transparency but focus on systems where results can be had immediately. Such an approach will go a long way toward increasing confidence in elections and their outcomes.

Meanwhile, for states focused on increased transparency in voting systems, consider how projects to produce such a system could be creatively funded. One approach is to press Congress to stand-up and fund a national research and development program to award grants to finish building this imperative public technology. The cost to do so would be a fraction of what has been (*and will continue to be*) spent on election security for the existing opaque and increasingly obsolete installed base of equipment across the country.

Election technology is a matter of national security. Public technology is a highly appropriate method and means of innovating election technology. Open-source is the way to increase transparency and lower cost. While there are no certified open-source voting systems today, until there are, the focus should be on basing other equally critical parts of election administration systems on public technology.