

Avoid Another HAVA Train Wreck:
Software Disclosure Requirements are a Good Long Term Goal
But Need to Be Redrafted in Current Federal Election Integrity Legislation.

U.S. Representative Rush Holt and Senator Bill Nelson have proposed publicly disclosing software in all voting systems in their bills (HR811 and S559)ⁱ

HR811 and S559 both say:

“(9) PROHIBITION OF USE OF UNDISCLOSED SOFTWARE IN VOTING SYSTEMS- No voting system used in an election for Federal office shall at any time contain or use any software not certified by the State for use in the election or any software undisclosed to the State in the certification process. The appropriate election official shall disclose, in electronic form, the source code, object code, and executable representation of the voting system software and firmware to the Commission, including ballot programming files, and the Commission shall make that source code, object code, executable representation, and ballot programming files available for inspection promptly upon request to any person.”

Definitions:

Source code is normally thought of as the humanly readable instructions that programmers use to create instructions for computers or DRE voting machines. There are dozens of different programming languages used to create source code software for any one voting machine, and programmers generally specialize in one to a few languages, so no one programmer is familiar with all the languages for writing instructions.

Object code is the translation of human-readable source code to something that the machine reads, often in the form of human-readable assembly code produced by the **compiler** program, and then turned into binary instructions for the computer by the assembler. "**Source**" and "**object**" code are relative terms. Given a device for transforming programs from one form to another, source code is what goes into the device, and object code (or "target" code) is what comes out. In summary, programs typically go through a series of transformations from higher level to lower level languages.ⁱⁱ

Binary Machine code is code produced by a **compiler** from the source code that can be directly executed by a processor, sometimes in assembly language. It is not possible for anyone after an election to read through binary machine code instructions which ran an electronic voting machine. The machine language code cannot be returned to the high level (source code) computer programming language that a programmer can understand. At most, the program code can be de-constructed to "assembly language".

Assembly Language code is time-consuming and tedious to read. Very few technical persons know how to read assembly code. Although it would be possible to tell what the code did after a very long time, a state-wide paper ballot recount would likely be faster. No court order could remedy this situation.

Open Source code is free publicly disclosed software.

Disclosed Source code denote public access to source code with a complete list of all third party code embedded in or accessed by the **disclosed software** when such **software** is run.

COTS software is Commercial Off-the-Shelf software

Easter Egg An Easter egg is hidden code in the machines that can only be activated by someone who knows how. A simple key stroke could activate it on Election Day. This hidden code will never be detected by any amount or kind of testing.ⁱⁱⁱ

Firmware is computer programming instructions that are permanently stored in a read-only memory unit rather than being implemented through software.

Ballot Programming Files give the definitions for what positions on the ballots specific candidates, parties, and races have. When these files contain mistakes, votes are switched to the wrong candidates.

**Avoid Another HAVA Train Wreck:
Software Disclosure Requirements are a Good Long Term Goal
But Need to Be Redrafted in Current Federal Election Integrity Legislation.**

Voting System Software Disclosure – A Great Goal

We assert that elections are public processes, and that there is no place for secret procedures anywhere in the voting system. We want to promote transparency in voting system technology, and better engineered systems that allow for post-election forensics examination when there have been errors introduced into the vote counts.

According to Alan Dechert, President of the Open Voting Consortium:

“When the government spends money, it is spending money that still belongs to The People. We, The People, have an absolute right to approve or disapprove of any expenditure of our money. When we are talking about spending state and federal “grant” money, taxpayers do not want money spent on secret proprietary technology for voting machines. If we’re going to spend public money on technology, it should be public technology.

Security by Obscurity is a failed computer security model. It just doesn't work. It only serves to facilitate vendor lock-in. The last place in the world that proprietary technology should be tolerated is the voting system. If we want to know exactly how the technology works, we can demand that before we spend the money.”

Open Source voting systems are often simpler, more reliable, less costly, more secure, and more transparent; and if designed using modern engineering security principles, will enable post-election forensic analysis of the voting system after an election and increase the likelihood of being able to obtain evidence to prosecute any perpetrator of electronic vote fraud. It is nearly impossible for an independent third party to verify programs were burned into the hardware if its program and circuitry are not open source.

The public wants all details of election administration to be made freely available to the entire public in a regular and systematic way.” This includes all details in the voting system technology, including source codes.

Limitations of Voting System Software Disclosure – what it will *NOT* do

“Source code inspection is simply a *quality assurance* technique we use in an environment where we are reasonably sure that the source code we're looking at will be the same as is run. It is not a *security mechanism*. For it to be so, we would have to trust the vendor, as well as every link in the rest of the very long chain of individuals involved in the end to end process of manufacturing, deploying, configuring, testing, operating, storing and monitoring the equipment and software, including all COTS software on the voting system.”^{iv} This broad level of trust is *inappropriate* when dealing with our voting system. The stakes --- control of the world's largest economy and the world's biggest military --- are simply far too high to trust that an attack on voting systems will not be made.

According to Professor Beebe, Center for Scientific Computing, U of Utah, and other computer experts, including Computer Professor David Dill of Stamford, the problems with electronic voting are NOT solvable to a sufficient degree that would make paperless electronic voting reliable. There are lots of components of the software where things can go wrong in a computer system, and even having open source software isn't sufficient.

Public software disclosure could mislead the public by making it seem as if IT professionals can "know" the source code is benign, "know" precisely what it will and won't do, and "know" where and how it is actually running in a particular device - when of course, they cannot.

ES&S voting systems require election specific memory chips to be inserted into every voting unit. ES&S does the programming that is downloaded to each memory chip before each election which tells the electronic voting system how to record votes that are entered either on a touch-screen or that are scanned by an optical scanner. We also know that they hire out that process to third parties because they do not have

by Kathy Dopp with help from dozens of technologists. kathy.dopp@gmail.com p 2 3/12/2007

**Avoid Another HAVA Train Wreck:
Software Disclosure Requirements are a Good Long Term Goal
But Need to Be Redrafted in Current Federal Election Integrity Legislation.**

adequate staff to handle the demand from all the various election jurisdictions throughout the country. There have been multiple reports from around the country in the last few years of candidate name misspellings on the touch-screens, omissions of candidates entirely, the crediting of votes to the wrong candidate, and memory chip failures. Those type of errors are egregious enough, but consider that when the memory chip programming is hired out to a third party that opens it, and the entire voting system into which it is later inserted, to deliberate subversion of the voting process undetected. It would be very difficult to sufficiently monitor this process via any certification and software disclosure process.

The Complexities of Voting Machine Software Disclosure

Should We Require That *All* Software be Disclosed or Exempt COTS software?

Representative Holt and Senator Nelson (HR811 and S559) seem to require disclosure of all software including commercial off-the-shelf (COTS) software including MS Windows operating system and all the drivers for all the hardware including the hard-drive, motherboard, and so on, but this isn't likely because Microsoft and other vendors typically will not publicly release their source code, hence this requirement essentially requires that that COTS software be either dropped from voting systems (a great idea, but in an unrealistic time frame that does not provide sufficient time for the development cycle). Alternatively, if COTS software were exempted from the disclosure requirements, then over 90% of the software that can hide vote rigging software is exempted. The complex and expensive enforcement provisions for software disclosure in this bill could waste time and resources and do little to make voting systems more trustworthy or secure.

The method of checking the programming instructions is to compare the machine language code on the voting machine with a second copy of the "same" high-level programming code (readable source code) independently compiled into machine code using an exact copy of all the hardware and drivers and compilation configuration settings as the original, making sure to have clean versions of the compiler software and all software-handling programs. Running a byte by byte check to compare the two versions of machine language takes one command and less than a minute once the test has been prepared. If two machine-language versions are the same then the programs must tell the same type of machine to do exactly the same steps.

Post-election verification of voting machine software can only be done if preparations are made prior to the election. It is not technically possible to verify the programming code used on the voting machine after an election unless the steps mentioned below are taken prior to the election. In fact, planning should really begin during the development phase of the voting machines.

With replicas of the exact hardware, firmware, software, compiler, and drivers of all voting machines in use, independent developers could then compile a binary image from the provided source code, and compare it to the image on the voting machine. Only if they match, could one certify that a machine used the provided source code.

How Many Components Are There to Check?

As part of the development process, the manufacturer of an electronic voting machine must provide full disclosure of both hardware and software, including version #s, as well as full documentation on how to create a new image from the provided source code. One needs the source code to the program, access to the scripts/build tools needed to build the source code, and a set of EXACT STEPS used to build the software (since varying these a little may give you a program that effectively does the same thing, but would be a different binary and would make comparison of binaries nearly impossible).

**Avoid Another HAVA Train Wreck:
Software Disclosure Requirements are a Good Long Term Goal
But Need to Be Redrafted in Current Federal Election Integrity Legislation.**

Today in just one county there could be dozens of variations of hardware, drivers, operating system and voting software for just one make and model of voting systems. For instance Diebold recycles voting systems rejected in one jurisdiction and sells them to another jurisdiction in another state. For public disclosure to work there would need to be quality control steps taken during the design and build phases.

Public Disclosure of Voting System Software requires publicly disclosing components of the system and their version identifiers, source code for each component including any file needed to build a complete version of the system, object code image for each component of system, checksums of object code image, specifications, documentation, internal and external document formats and sample documents, hardware dependencies, specifications, and requirements. For each commercial off the shelf (COTS) software component, specifications, version numbers, dates of manufacture, requirements and uses are needed plus image; feature checklist, license(s) for the system, reports of non-internal tests, and an attestation that all components and descriptions submitted are accurate and represent the versions identified. The vendor may retain all copyrights, trademark rights, and patent rights needed under Copyright law (17 USC 106) but not trade secret rights. Non-disclosure agreement (NDA) requirements should be allowed only for COTS components. However COTS binary machine code would still need to be checked.

"...complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable..."

The program isn't the sole location where problems can reside. Intermittent failure, shared libraries, kernel modules, network connections, storage device connections, and others, any of which could be used to intentionally subvert the voting process or invalidate the vote. The whole set of circumstances and procedures in which they are used are important. Potential for these problems need to be minimized in any voting system design.

To verify the system, you have to use a trusted compiler sufficiently similar to produce the same output. Use one that's too different and you risk false alarms. Use the compiler that the vendor gives you and you are trusting an un-audited tool in the chain that could add a backdoor login into the system. Any program-handling program such as an assembler, a loader, or even hardware microcode can have the same problem. As much as possible, off the shelf hardware should be used. It might be possible to subvert the computer by using a chip that interprets the machine code differently. Supply one version for audits, use the other in distribution. But if that chip is a standard x86 from the neighborhood computer shop, you're talking a pretty big conspiracy before it can be subverted.

There are tens of thousands, if not hundreds of thousands of devices to potentially check, so not only is this extreme degree of forensic comparison impossible to contemplate doing for every voting device prior to every election, it's also ultimately pointless. Since real-world computer systems involve complex inventories of hundreds or even thousands of application program modules, firmware, device drivers and operating system components, static inspection alone will never be able to reliably determine what those components will actually *do* at any given point in time. There's simply no reason to believe that a given executable corresponds to the given source code, and no way to truly know what the executable is doing - except by running it.

There is the compiler, O/S, loadable kernel modules, the file system, the network, interrupt handlers, a collection of shared libraries, and so on. These simply cannot be validated to a level that is required for electronic voting to be acceptable.

**Avoid Another HAVA Train Wreck:
Software Disclosure Requirements are a Good Long Term Goal
But Need to Be Redrafted in Current Federal Election Integrity Legislation.**

Other Problems

Transmission of the tally is probably the most vulnerable point. By its nature, it has to be possible to change the count. Any point in storage and transmission of that count is vulnerable, especially when an attacker has physical access.

No one can tell by inspecting the official source code whether a particular PC voting machine has malware, spyware or, worst of all, a rootkit. Much less can we possibly know precisely how a particular application will behave at an arbitrary time during the election by looking at source code. It is the same for any real-world computing device.

According to Bruce O'Dell, Computer Security Consultant^v:

“Even in the highly-controlled, regulated and audited environment of a bank or brokerage house, it is extraordinarily difficult in practice to know precisely what software components are executing on which devices at any given time. But in theory, with specialized equipment and intensive hands-on effort, a skilled computer forensics auditor could rigorously examine each hardware component of an election system and compare the detailed contents of the installed software components to the version of the executables registered with the EAC, if they trust their forensic software of course, and if they always accurately report their findings. Care to bet the Republic on that?”

To ensure that the certified software actually ran on voting machines during elections, the software on each voting machine would have to be verified after receiving it from the vendor and before using it in an election, and one would have to trust the persons doing the verification not to install any Easter eggs or other malicious software themselves; no technical work should be done afterwards on the machines; and after each election, the machines' software would be re-verified.

Some Reasons Why the Software Disclosure Provisions of Representative Holt and Senator Nelson bills (HR811 and S559) Will NOT Work

1. It requires disclosure of all software including commercial off-the-shelf (COTS) software but this is not going to happen because Microsoft and other vendors of other commercial software, hardware and firmware will not publicly disclose their software, so in essence HR811/S559 requires that COTS software be dropped (a great idea) but in an unrealistic time frame because this would need at least several years for the development cycle and funding for new equipment, or
2. Alternatively it exempts COTS software, so that over 90% of the software which can hide vote rigging code is exempted - so these complex and expensive-to-enforce provisions for software disclosure in these bills are ineffective at best, and

“As written, HR 811/S559 would require depositing tens of millions of lines of source code with the EAC; which is far more than any one person could hope to read, much less understand with complete clarity even if we *could* somehow get our hands on an accurate copy of the hundreds of thousands of pages of all the vendors' closely-guarded source code including the official source code for Windows, Microsoft Office, and all the hundreds of other software applications and components installed over time on the voting machines.”^{vi} Their disclosure provisions would tell us precisely nothing about the true, current state of an individual voting machine used in an election.

HR811/S559 requires disclosure to the US EAC, rather than complete public disclosure on the Internet to the public. We would need to have ultimate trust in the US EAC and the EAC would need to have significant resources to handle “public” disclosure in this manner. The language of HR811/S559 is very vague on when the EAC would allow anyone from the public to review the source code.

**Avoid Another HAVA Train Wreck:
Software Disclosure Requirements are a Good Long Term Goal
But Need to Be Redrafted in Current Federal Election Integrity Legislation.**

Nor is publicly disclosed software a practical solution to the problem of malicious programs or undetected program errors, if people who are allowed to read it must sign nondisclosure agreements, because few, if any, experts in voting software can examine such disclosed software due to intellectual property contamination issues. Both open source and proprietary voting systems experts would be barred from signing the nondisclosure to examine the code because they could be sued if similar features end up in the programs they work on.

It is very difficult to verify that the disclosed software was actually used on the voting machines during elections, and election tampering or errors can also occur via any one of the numerous software drivers or firmware or even via an operating system file, especially if the machine's hardware and firmware components and drivers are proprietary or secret.

HR811/S559 allocates insufficient resources and provisions for source code review especially as it seems to leave this up to fairly non-public efforts within the EAC.

HR811/S559 requires that many thousands of persons in various countries and states all be subjected to background checks (unless it exempts all COTS software which is over 90% of the software on today's commonly used voting systems).

“The source code disclosure requirement of HR811/S559 is unenforceable in practice - there are a *lot* of hardware and software components inside voting equipment. Not only proprietary software developed by voting equipment vendors, but also mass market consumer products like Microsoft Windows, and also a host of highly complex, very specialized software from vendors, many of them offshore. Surprisingly those other vendors simply have no interest at all in giving away their crown jewels to their competition. (If source code inspection could allow us to reliably predict how a particular instance of a program will actually work in the field, Microsoft Windows would be a rock-solid, bulletproof product - after all, *tens of thousands of programmers* spend their professional careers scrutinizing its source code every day.)”^{vii}

It is very difficult to detect cheating with hardware^{viii}.

What Legislative Measures Could Achieve Full Software Disclosure of Voting Systems and Open Source Software on Voting Systems?

The following are loose thoughts and rough ideas. Experts in this arena need to be consulted. The main point is that incentives, rather than requirements, might work better, and be more economical to move us towards open source code for all U.S. voting systems.

Nevada Gaming Commission has a procedure for qualifying computerized slot machines that may be similar enough to help when it comes time to develop an actual evaluation process to certify the software on voting machines. However, with today's voting systems, attempting to routinely verify and certify all the software, including all the COTS software, would be virtually impossible with any realistic amount of resources.

Require technology disclosure for new certifications, and also provides an incentive for vendors that bring open source products for state certification.

Require software to be archived in a public repository on a web site, not just “on request to the EAC” such as:

1. Secretaries Of State/Lt. Governor
2. Any third party repository such as Source Forge or the Open Voting Consortium

**Avoid Another HAVA Train Wreck:
Software Disclosure Requirements are a Good Long Term Goal
But Need to Be Redrafted in Current Federal Election Integrity Legislation.**

Source software Review by:

1. Public, or
2. Project ACCURATE has funding for software review, or
3. Universities, or
4. Independent Testing Authorities

Certification would require:

1. Software Review
2. Public Disclosure of all voting system software in repository
 - a. Source code, object code, machine code
 - b. Hardware product data sheets for each and every voting machine by serial number
3. Verification of all key software on voting systems via hash code verification, including COTS software, prior to certification and following any updates to software

Two ways to give incentives to use open source software^{ix}:


1. Pay for the certification of open source software (\$1 mill+ per year) - It makes sense for the public to pay for testing for open source software since it's free public software owned by the people.
2. Pay for a staffer (\$200,000/year+-) with a Quality Assurance/software testing background

It is possible that one technically competent person in each state, or several staffers at the federal level, with software/hardware test and quality assurance experience leading the certification effort could do the job of certifying open source code by calling on help from the ACCURATE project, other university campuses, and the open source community.

None of this would help open source systems to come into play in time for the 2008 elections, but, with incentives, such systems would rapidly gain market share because they are ready to go now (except for certification), and they will be much less expensive and much more trustworthy. Perhaps Congress should do a thorough study first, making sure to know what all the costs are, prior to requiring software disclosure.

Possibly use some of the language in this proposed CA bill as a model http://www.leginfo.ca.gov/pub/07-08/bill/asm/ab_0851-0900/ab_852_bill_20070222_introduced.html but correct the language in Section 2 (h) to say

“(h) For products submitted for state certification that are open source *except for* all unmodified COTS components, the Secretary of State may, at his or her discretion, elect to forego the federal certification requirement and certify the product using a special process established by the secretary for this purpose.”

 A post-election or pre-election software check could possibly be prepared from a central location and distributed to local technicians prior to the election. A complete copy of the system drives and memory on each voting machine could be made ASAP after the voting period and before connecting to a network or transmitting data. The copy program and backup drives would have to be examined for potential problems before using them. A study of ways to prevent tampering could be done and recommendations made to election districts.

Perhaps there should be a requirement that software be developed according to a quality process - something akin to what the FDA requires of software to control medical diagnostic equipment. The general principle is that every step of the process is documented and that there is an audit trail that can be verified by FDA auditors.

Procedures to assure that the user documentation, the functional descriptions of the software and the testing is all done on the same version of the software because there have been problems in this aspect with current

**Avoid Another HAVA Train Wreck:
Software Disclosure Requirements are a Good Long Term Goal
But Need to Be Redrafted in Current Federal Election Integrity Legislation.**

machines. In the design of each software and hardware component should be a test procedure. One approach would be to create another subcommittee to specify requirements for the software development process to produce voting devices.

Conclusion

Current voting system software disclosure provisions of proposed federal legislation need to be dropped and rewritten in separate legislation in consultation with experts with diverse technical backgrounds^x.

It might be wiser to pass legislation that could be implemented by 2008 requiring sufficient manual audits (and auditable voting systems for all states by providing funds to replace paperless DRE voting systems and replace them with optical scan paper ballot voting systems^{xi}) and require states to provide public access to election records that are necessary to verify election outcomes; and to defer to another bill the requirements for long-term improvement of voting equipment to be publicly disclosed that would require significant time for development cycles.

Acknowledgements

This information was written with input from dozens of computer scientists and technicians, including information from Joseph Hall, Arthur Keller, Jim Soper, Bruce O'Dell^{xii}, Alan Dechert, Danny Swarzman, Nelson Beebe, members of the Salt Lake Linux Users Group and many other technologists whom I have not properly acknowledged here because they were contributors to an on-line article that I wrote in 2003 without attributions. Bill Bucolo provided editing help.

ⁱ The text for HR811 and S559 can be found on [Thomas.loc.gov](http://thomas.loc.gov)

ⁱⁱ <http://www.cs.cmu.edu/~dst/DeCSS/object-code.txt> "Source vs. Object Code: A False Dichotomy", David S.

Touretzky, Computer Science Dept, Carnegie Mellon University

ⁱⁱⁱ Definition by Jim Soper <http://www.countedascast.com/issues/testing.php>

^{iv} Bruce O'Dell, Computer Security Consultant, Digital Agility, and member of Election Defense Alliance, OpEdNews.com article: http://www.opednews.com/articles/opedne_bruce_o_070221_holt_s_hr_811_a_dece.htm

^v *ibid* iv, O'Dell

^{vi} *ibid* iv, O'Dell

^{vii} *ibid* iv, O'Dell

^{viii} There is an article on "Malware Loader" as described in this study. (incomplete endnote)

^{ix} These incentives were suggested by Alan Dechert, President of the Open Voting Consortium.

^x Perhaps someone like Barbara Simons could lead this effort. See <http://electionarchive.org/ucvInfo/US/ExpertsList.pdf>

^{xi} Optical scan voting systems with voter-verified paper ballots protect ballot privacy, unlike DREs with paper rolls. Optical scan voting system software cannot subvert manual audits via software hacks that wrongly alter post-facto voter-verifiable paper records, unlike DREs with paper rolls. Optical scan voting systems use durable, archival weight paper ballots that are good for audits, unlike DREs with paper rolls. Optical scan voting systems are less expensive to operate and maintain than DREs with paper rolls. Federal funds should never fund systems with invisible electronic ballots that able-bodied voters cannot verify directly, such as DREs.

^{xii} *Ibid* iv, O'Dell