A Report to the
CALIFORNIA LEGISLATURE *on*

Open Source Software in Voting Systems

JANUARY 2006

CALIFORNIA SECRETARY OF STATE **BRUCE MCPHERSON**

BRUCE McPHERSON | SECRETARY OF STATE | STATE OF CALIFORNIA
EXECUTIVE OFFICE | 1500 11th Street, Sacramento, CA, 95814 **tel** 916.653.7244 **fax** 916.653.4620 www.ss.ca.gov

With the passage of Assembly Concurrent Resolution 242 in 2004, the Legislature requested that the Secretary of State's Office investigate the possibility of the use of open source software in voting system technology and whether it might be beneficial to the integrity of the electoral process. Although the Legislature notified the Secretary of State in August 2004 that he was to undertake this task, it appears no work on this complex subject was conducted by the prior administration during its tenure.

It is also important to note that many events have occurred in parallel with this request, specifically: a transition to a new administration within the Secretary of State's Office just ten months ago, the administration of a special statewide election, an unprecedented number of initiatives being circulated in 2005, and the requirements to implement many new election systems processes to comply with state law and the Federal Help America Vote Act of 2002 (HAVA). Amid all of these additional duties, we have conducted the required study on the possibilities of reliance on open source software for voting systems.

In an effort to address the critical issue of voting systems security, the following steps have been undertaken:

- I hosted a national Voting Systems Testing Summit, at which the entire process and approach to testing and certifying voting systems was examined by an array of the country's leading experts on the subject;
- I convened a multi-state panel of Secretaries of State and their staffs to discuss the use of open source software in voting systems;
- I implemented a ten step process that identifies how a voting system will be examined to ensure accuracy, security and usability; and
- My staff conducted academic research and solicited public comment and validated the need for further extensive technical research into the use of open source software.

Presented herein is my report in response to the Legislature's request to investigate the viability of using open source software in voting systems.

If you have any questions on this matter, please do not hesitate to contact me, or my Assistant Secretary of State for Legislative and Constituent Affairs Theresa Taylor Carroll (916) 653-6774.

Sincerely,

BRUCE McPHERSON
Secretary of State

TABLE OF CONTENTS

I. Executive Summary

The California State Legislature, through Assembly Concurrent Resolution (ACR) 242, Resolution Chapter 171, Statues of 2004, requested the Secretary of State to investigate and evaluate the use of open source software in all voting machines in California and report his findings and recommendations.

Electronic voting technology was introduced with the expectation that election accuracy, security, accessibility, efficiency and public confidence would all be enhanced by its use. Among the many policy initiatives proposed to address the support of these efforts is the use of open source software. At the outset, open source software is software whose inner workings are available for public review and unfettered technical scrutiny. However, in practice, the business of technical innovation and the effectiveness of information security may both depend upon some measure of confidentiality, which open source, by its very nature, precludes.

Open source advocates point to impressive accomplishments for software developed and maintained according to their principles, with apparent benefits to costs, efficiency, quality and security; however, upon close examination, the open source experience is more limited in scope and specific in application.

In several important aspects, electronic voting systems may not be currently congruent with the model of open source initiatives. The technical requirements for voting systems are much deeper and broader, extending from hardware through several layers of software. In addition, the use of voting systems is quite narrow yet demanding, encompassing just several hours per year with little opportunity for delay or repetition. The security requirements necessary for an electronic voting system are particularly unforgiving, with the need to eliminate, not merely detect, the possibility of compromise. The experience to date with open source software does not provide much basis for evaluating the ability of the open source model to meet these requirements.

On the other hand, electronic voting systems are themselves in the pioneering stages; traditional closed source software development models, although far more extensive, are not far more proven for electronic voting systems. For that reason alone, the investigations and evaluations of this report cannot yet be conclusive.

Further, closed source electronic voting systems currently exist while open source systems do not. A policy decision to require open source software for voting systems would disrupt existing voting systems without providing an immediate alternative.

The Secretary of State therefore believes that, while it is premature to initiate a policy requiring open source software in voting systems, it would be beneficial to continue the studies and discussions on the topic of open source software for use in voting systems.

The magnitude of the potential task to undertake the development of open source systems, as well as the potential impact on the public domain of voting systems, dictates that California not proceed unilaterally. The Secretary of State, therefore, recommends that the California State Legislature invite interested parties, such as information technology experts, elections officials, voting system vendors, other states, academic institutions and other interested citizens to participate in further discussion, study, and research on the issue of open source software in voting systems.

II. Introduction and Background

California Assembly Concurrent Resolution 242 (ACR 242) introduced by Assembly Member Jackie Goldberg on June 3, 2004, and chaptered on August 31, 2004, requests the Secretary of State to "investigate and evaluate the use of open source software in all voting machines in California and report his or her findings and recommendations to the Legislature by January 1, 2006."

The primary goal of this report is to summarize for the Legislature the complexities of source code use in voting systems in California, identify relevant questions about open source software and highlight areas requiring further review. In addition, this report will address the standards currently in place, in both law and practice, which ensure voter confidence in the California's voting systems.

Existing California law requires the Secretary of State to establish the specifications for, and the regulations governing, voting systems and vote tabulation, and any software used for each, as well as the deposit of an exact copy of the source code for all ballot tally software programs in an approved escrow facility prior to their use. State law also requires the Secretary of State to adopt regulations relating to the definition of source codes, specifications for the escrow facility, procedures for submitting ballot tally software program source codes, and criteria for access to ballot tally software program source codes.

As requested in ACR 242, this report consists of findings and recommendations resulting from research, review of materials submitted by professionals and the public, and discussions held with several Secretaries of State around the nation. Technology is continuously evolving, making the topic of open source software in voting systems a particularly complex issue.

It is of the utmost importance that voters have confidence in the integrity of the electoral process, and the Secretary of State has taken aggressive steps to protect that most fundamental tool of our representative form of government. Special attention has been, and will continue to be, devoted specifically to testing and approval of the voting equipment on which votes are cast in California.

III. Voting Systems in California

The elections process and the mechanics by which votes are cast in the United States have constantly evolved. The right to vote has been revised and extended by constitutional amendments and various judicial and legislative actions over the course of history. Voting systems have been designed, tested and approved, employed, revised, and rejected since their debut in the mid-19th Century.

On June 1, 1869, Thomas Edison received a patent for an electronic voting machine, but widespread use of electronic voting equipment is a recent phenomenon. Direct Recording Electronic (DRE) voting systems were first introduced in the 1970's. Optical scan systems were applied to voting systems in the 1980's. By the 2004 general election, about 64 percent of the nation's registered voters were casting ballots on these two types of systems – 35 percent on optical scan systems and 29 percent on DREs.

Currently, California has the most rigorous voting system certification process in the nation. Our procedures are designed to ensure security, reliability, usability, and accessibility by all voters. As federal requirements evolve and technology advances, the Secretary of State has worked to ensure California voters have the most reliable, secure and highest quality voting systems available.

As state and federal laws, as well as technological development, have moved California and the rest of the nation away from earlier mechanical voting systems, some members of the public have raised concerns over the security of the new Direct Recording Electronic (DREs) voting equipment. Some argue that DRE voting equipment relies on a "black box" computer with proprietary source code and object code hidden from public scrutiny. They allege voting system vendor "insiders" could insert malicious code and similar security violations into the machinery's software that could manipulate vote totals and affect the outcome of elections without being detected. Due to the rigorous testing that occurs prior to federal and state certification of voting systems, in conjunction with pre and post election procedures, it is unlikely that such a vulnerability, even an inadvertent one, would be undetected.

Additionally, California is moving to ensure that its voting systems testing and certification process meets the evolving conditions pursuant to HAVA. The Secretary of State has put into place a number of stringent conditions, which must be met by vendors if they wish their voting systems to be eligible for use in California. In addition, the Secretary of State's Office is in the process of developing a best practices blueprint for state-level testing of voting systems as an outgrowth of the national Voting Systems Testing Summit the Secretary of State hosted in November 2005. Finally, in October 2005, Secretary of State Bruce McPherson announced the creation of the Office of Voting Systems Technology Assessment. The office will be a permanent, professional, and fully

staffed component of the Agency serving as a one-stop shop for voting system examination and certification.

For the first time in California, vendors who wish to present their voting systems for use in this state have a clear set of requirements and standards, outlined in advance, that guarantee all are treated equitably and held to the highest possible standards. These new conditions provide a step-by-step process that identifies how a voting system will be examined to ensure accuracy, security, and usability. The requirements, which will be codified into state regulation, are:

1. An application that includes proof of federal certification, (i.e. copies of the United States Election Assistance Commission [EAC] approved Independent Testing Authority [ITA] reports and certification number issued by the EAC.)
2. All applications must include full documentation, including technical and operational specifications, operating and maintenance manuals, training materials, and copies of all promotional materials from the vendor.
3. Each system must have comprehensive use procedures applicable for use of the system in California elections in conformance with state law and Secretary of State guidelines for voting system use procedures.
4. Each vendor must establish a California County User Group and hold at least one annual meeting where all California users and Secretary of State staff are invited to attend and review the system and ensure voter accessibility.
5. In addition to depositing the source code in an approved escrow facility, each vendor must deposit a copy of the system source code and binary executables with the Secretary of State. The Secretary of State reserves the right to perform a full independent review of the source code.
6. Voting system vendors shall provide to the Secretary of State, on request, a working version of the components, including all hardware, software, and firmware, of the voting system that is proposed for use in an election, for the purposes of analysis and testing. These components shall be maintained in working order by the vendor.
7. Multiple independently tested and certified voting systems may be used together to meet federal and state requirements so long as their interface is limited to exchange of aggregated vote totals and/or ballot layout.
8. Components of a certified voting system may not be combined with components of a previously but separately certified system without certification by the Secretary of State as an entirely new system.
9. The vendor for each system that utilizes paper ballots will be required to provide printing specifications for those ballots to the Secretary of State. The Secretary of State will certify printers to print ballots for each system based upon their demonstrated ability to do so. Vendors may not require exclusivity in ballot printing and must cooperate fully in certification testing of ballots produced by other ballot printers.
10. All systems will be subject to volume testing as defined by established Secretary of State standards prior to certification in California.

IV. Background on Open Source

The most important thing to understand about open source software is that it is no different than other software. Therefore, open source software cannot be objectively compared to traditionally-developed software performing the same function, and the benefits, if any, of open source software cannot be directly measured. Instead, the comparison must be made largely on the basis of theory and anecdotal experience.

Fundamentally, open source software is simply a software product whose code is available for public view. Computer systems are almost always a combination of hardware and software; software being the instructions that tell the hardware what to do and how to do it. As is now commonly known, computers understand only languages of bits—ones and zeroes—while humans find it much more effective to work in more sophisticated languages. These more complex, programming languages are translated using software tools such as compilers and assemblers into the bit streams—often called "object code"—the computers can read. It is usually not possible to recover the source code from the object code, nor is it feasible to read the object code to determine what the computer is being instructed to do.

A. Implications of Open Source and the Marketplace

Software intellectual property is generally protected by copyright rather than patent. A competitor could potentially replicate a software product without incurring the full research and development costs by examining the source code to learn its functions and internal mechanisms. Consequently, software manufacturers believe it is critically important to protect the secrecy of the source code to defend the value of their product. They almost always provide only the computer-readable object code to their customers.

Regardless of the ultimate business viability of a software product, it costs a great deal of time and effort to develop and maintain a modern commercially-useful software product. In the example of the open source operating system Linux, much of the work was and is performed by idealistic volunteers, using readily available consumer-priced hardware to develop a product they could all use personally. The Apache web server environment has a similar development and maintenance mechanism. In both cases, though, one or more not-for-profit organizations serve to raise and distribute funds for activities and equipment that would otherwise tax the resources of a purely volunteer cooperative. Netscape, on the other hand, although initially developed in a similar manner, was carried into maturity as a commercial profit-making venture, with a twist. Inverting the classic "give away the razors, charge for the blades" approach, Netscape hoped to profit from the sale of web server software and related products, while giving away the browsers that would make use of them.

In any case, the effort to create and sustain a viable, professional-quality software product is quite substantial, and if the bulk of that effort is to be borne by volunteers, a large and competent community must be cultivated and nourished. Even where there is precedent for software development by a cooperative of volunteers, the extension of the software so developed into real-world productive use has required substantial infusions of cash, up front and ongoing. And, in most of those cases, the software was something that could be developed, maintained, and used on readily-available and relatively-cheap hardware platforms. In the case of Linux, the vast majority of the ongoing work is fixing bugs, and adding support for new hardware—printers, scanners, monitors, etc. This is work that is not only relatively quick and easy to perform on mature software, but has immediate benefits to the programmer: he can make his new printer print, or fix a bug that's bugging him, all while contributing to the open software cause.

B. Technical Quality and Security of Open Source

Advocates frequently claim that open source software is higher in quality: including more reliable, better performing, and lower in total cost of ownership than commercial products; however, the objective evidence to support these claims is intriguing, but hardly conclusive.

More important is the assertion that open source software is inherently more secure than similar closed source commercial products. If the source code is available for all to review, it is argued, a large community of professionals will naturally scrutinize the code for security flaws, and because this community has no financial stake in the software product, will feel no constraint on publicizing their findings. The community will then, it is further claimed, act quickly to repair the flaw, and in the meantime, everyone is aware of the risk and can take proper countermeasures.

In addition to the open source operating systems and browsers already mentioned—more on those in a moment—the important real world examples here are the various encryption algorithms. Nearly all of the important encryption algorithms are open: the exact mathematical calculations necessary to encrypt and decrypt a message, document, or chunk of data are freely published. The products based on these algorithms have become successful because the underlying algorithms have been, and continue to be, subjected to intense scrutiny by the community of professional and academic cryptographers.

But there are at least two ways in which the encryption algorithm example may not be analogous to other software markets. First, is that the openness applies to the encryption algorithms, not to the software products that use the algorithm to encrypt and decrypt email, communications links, data files, and the like. These products do not share with the algorithms, and their commercial viability is not affected by, the open availability of their source code.

More important to the goal of security, the benefits of openness for encryption algorithms depend on the existence of a very large, robust, and capable community of cryptographers, whose members share a professional desire to be the first to crack a particular algorithm. If such a community does not exist, or is not large, capable, or motivated enough to quickly find and address security flaws, the open publication of the code will merely assist potential attackers. Any potential open source attacker may already have a powerful advantage: he must find only one vulnerability to exploit; the defenders must find and fix them all.

The Linux and Netscape examples are again instructive: both Linux and the Netscape browser (and its successors) are frequently cited by open source advocates as more secure than their commercial alternatives. The arguments are largely theoretical: that open software products are more secure because they are open to scrutiny, and if a vulnerability is found, it will be immediately publicized so users can take appropriate precautions. In practice, the results are far from conclusive. Both Linux and Netscape-based browsers are certainly subject to security flaws, some of which have led to actual attacks exploiting those flaws. There is no clear evidence that the attackers used the open source code to find the flaws in the first place. There is, however, at least circumstantial evidence that attackers used the open reporting of the existence of a flaw to develop an exploit before the vulnerability could be corrected.

In sum, the effectiveness of opening source code as a security measure has not been convincingly demonstrated. This is not to say, however, that opening source code may have no value for security. But these benefits must be weighed against the potential risks of openness in careful analyses of the intended application and use of the software.

C. System Components

Much as the practical examples of open source do not present a complete picture, to date open source software has not been used to develop a complete system. Computer systems are combinations of many software components, combined together and stacked on top of one another like layers on a cake. Each additional piece of hardware; printer, network adapter, mouse, touch screen; adds at least two more pieces of software: code in the device, and corresponding software in the computer that uses it.

At the lowest level, every processor chip has embedded code—firmware—built into the chip, stored on an adjacent chip and loaded at startup, or both. Firmware is almost always the province of the hardware manufacturer; the source code is rarely published.

At the highest level are the application programs; spreadsheets, word processors, web browsers, vote recorders, etc. These products are specific to

the task; the more specialized that task, the smaller the potential number of users, developers and funding sources—but there is not necessarily less complexity or reduced costs.

In between is the operating system, connecting the application programs to the computer, and performing all the routine tasks of sending data to and from disks, networks, and peripheral devices, sharing the computer between different applications, and underlying every function. These products include Microsoft's Windows family, the various versions of Unix including Linux, and a number of less familiar, more specialized operating systems for computers as small as a smart card and as big as a mainframe. The operating systems themselves are so large and complex that they contain dozens or hundreds of components that derive from different sources, may date back years, and have authors outside the software publisher's control.

Finally, between the operating system and the hardware is a class of programs called drivers. Drivers translate the generalized operating system functions to the commands and functions specific to a particular product, be it printer, scanner, digital camera, or something not yet invented. Typically, drivers are the province of the hardware manufacturer, although in practice they are largely integrated into the operating system.

Consequently, a thorough examination of a system for security vulnerabilities must include a perusal of each and every one of these components. If that examination is to happen through the open source process, then every one of these components must, theoretically at least, be open. Yet open versions of many of these components do not exist.

V. Electronic Voting Systems

Perhaps the most compelling case for open source software in electronic voting systems is the need to assure the public that the portion of the election process that occurs in the system is as open, fair, and honest as the democracy that depends on it – an assurance that is achieved currently through California's stringent security and use procedures. But a more specific analysis of the benefits and costs of open source voting systems does not make a clear case for the adoption of open source. While there may be benefits, they are neither empirical nor measurable. There is no precedent that fully supports the feasibility of a potential effort to develop, deploy, and maintain an open source election system.

A. Development

As noted earlier, the practical examples of open source software products strongly suggest that those products will not be developed by commercial enterprises. If true, this factor has two major implications. A source of funding, or at least of skilled volunteers, must be found and sustained. Complete electronic voting systems are very complex; the development and support effort will be very large. Should this effort be successful, it will at least damage, if not destroy, the viability of the existing voting systems structure, and with it the voting system industry's accumulated experience and expertise.

Moreover, unlike the existing open source software models, a true open source voting system arguably must open every bit of code on the entire system, for nearly every piece of software in the system, from processor instructions to firmware to drivers to operating systems to the voting application itself, can affect results, either by accident or design. To date, most proposals for open source electronic voting systems assume the use of the Linux operating system with a newly developed voting application. Quite a bit of closed source software would likely remain on such a system and its peripherals.

B. Security

The particular considerations of voting systems also complicate the potential security benefits of opening the source code. As others have noted, publishing source code only improves security if enough competent scrutiny of the code occurs to ensure that flaws are found. Otherwise, opening the source advantages the attacker. Voting systems must change frequently to adapt to changing laws and other requirements. Even without functional changes, software is regularly updated to address flaws, security and otherwise. Each time the software changes, at least the changed part must be carefully reexamined. The code review burden on a voluntary community would be heavy and endless.

Further, even if the open review process results in the discovery of a dangerous security flaw, that very discovery can threaten the elections process: if the vulnerability is not found in time to correct it, to reexamine the corrected code, and to deploy the fix onto perhaps thousands of affected devices, should the election continue anyway when that flaw surely is visible to attackers? Postponing elections is not a trivial matter, and may in itself constitute success for a potential attacker.

C. Practical Experience

Overall, compared to the general history of information technology, the total experience with open source software, although in many ways promising, is sparse. Prudence strongly suggests that open source software is not ready to be cast in voting systems policy.

There remains, however, considerable power in some of the arguments for open source. Both public confidence and security are vitally important in voting systems, and the potential benefits of outside review of otherwise closed and proprietary voting systems are obvious. But, just as evident are the risks to security and to the existing, and robustly innovative, voting systems industry of requiring the open publication of voting systems source code.

D. Disclosed Source

It may not be necessary to commit completely to open source for voting systems to obtain its key benefits for voting systems. A variation on open source, sometimes called *disclosed* source, involves the unrestricted review of source code by an independent team of experts, who are otherwise bound to protect the trade secrets and unaddressed vulnerabilities in the software.

A disclosed source strategy, while avoiding some of the most troubling problems with open source, is not without difficulty. It can be difficult to assemble an "independent panel of experts". The number of true experts in any field, especially elections, is limited, and some are at least biased through a need to retain gainful employment. Moreover, it is human nature to form opinions, often strong ones, especially in one's field of expertise. Consequently, a team of experts formed to review voting systems source code is more likely to be balanced than truly impartial, with controversy at least as likely an outcome as consensus.

Further, the volume of source code to be reviewed remains very large and unending. A pure open source approach at least has the potential, if mixed, benefit of reducing the number of separate products to be reviewed. If a disclosed source approach achieves the goal of preserving the competitive market, it will also ensure that at least several separate voting systems environments will require review. This review must recur every time the source

code changes. Each time, the goal must be to identify every security vulnerability in the code, and ensure that every calculation and result, under every circumstance, will be correct. This effort is very large.

The volume of work involved in the code review, and the need for experts who are probably otherwise in demand, virtually ensures that the panel members must be compensated for their time. Consequently, a disclosed source review program, perhaps unlike a true open source effort in which the required scrutiny depends at least partly on the efforts of a voluntary community, will be costly. Whether or not the review program is funded by a fee charged the software manufacturers, the ultimate costs will be borne by the taxpayers who pay for the voting systems.

Finally, it is far from certain that all of the manufacturers of all of the software components on a complete voting system will agree to submit their source code for review. Many, including commercial operating system and hardware (firmware) providers, are involved only because the voting system manufacturer chose to use their products to build the voting system. A thorough disclosed source policy might operate either to limit the ability of existing voting systems manufacturers to participate because they can't bring forward all of the source in their systems, or may have to be a code review that is limited to the voting systems application software itself, not the other software used on the system.

The potential risks of open source are impossible to quantify or to constrain. A careful program of source disclosure, however, may provide nearly all the likely benefits of open source, with much less uncertainty of outcome and disruption to proven paradigms.

VI. Investigation and Findings

The request from the California State Legislature to investigate and evaluate the use of open source software in voting systems produced the need for a technical examination of a complex subject that represents viewpoints from many quarters.

With the increase of new technology designed to improve efficiency in virtually every aspect of daily life, it is no surprise that computer scientists, members of the academic community, voting rights advocates, vendors, and all levels of government are focusing on ways to utilize those advances in making the voting process more secure, accessible, reliable and user-friendly.

The Secretary of State's Office sought input from a field of experts in its quest to investigate the subject of open source software in voting systems.

Resources

Voting Systems Vendors:

- Sequoia Voting Systems
- Hart InterCivic

Open Source Software Advocates:

- Alan Dechert, Open Voting Consortium
- Richard Dawson
- John Miche

Elections Officials:

- David Tom, San Mateo County

Academia:

- Michael Alvarez, CalTech-MIT/Voting Technology Project
- Michael Shamos, Carnegie Mellon University

Associations:

- Information Technology Association of America

Voting Rights Advocates:

- Kim Alexander, California Voting Foundation

Computer Scientists:

- David Jefferson

Research shows the generally perceived advantages:

- Making code readily modifiable and adaptable to meet state's specific needs and requirements;
- Reducing the cost for creation, modification, and licensing of the software used in voting systems;
- Improving public access ensures software evolves and improves, resulting in a better product;
- Developing modifications more quickly when deficiencies are discovered;
- Improving the availability of support services and options;
- Ensuring ongoing viability of the system by eliminating the danger that a system becomes unsupportable because a vendor goes out of business – (i.e. freedom from reliance on a vendor);
- Creating an alterntative source of support if vendor support fees become exorbitant; and
- Eliminating licensing requirements, the source of many legal issues and costs, as well as the costs and administrative workload associated with tracking of software copies and usage.

Research shows the generally perceived disadvantages:

- Making the software vulnerable to hacking and security breaches by making it widely available to the public;
- Eliminating the incentive for high quality research, which is motivated by the possibility of financial gains;
- Incurring the significant research costs for products that make the business case for closed source code economically sound – and there's value (marketability) to your "secret";
- Losing the advantages gained by vendor expertise and experience; and
- Employing a public policy alternative to proprietary software, which may have a negative impact on intellectual property rights.

Internet resources: www.opensource.org, www.rediff.com/cms/print.jsp?docpath=/money/2005/dec/20guest.htm, www.oreillynet.com/lpt/a/4807,http://webopedia.com/TERM/o/open_source.html www.wired.com/news/ebiz/0,1272,61045,00.html

Additional research found that to date, the only electoral jurisdiction (of which we are aware) that has developed and used open source software for a voting system is the Australian Capitol Territory (ACT). The ACT Electoral Commission conducted their 2004 Legislative Assembly election using open source software, which utilized Linux code for their electronic voting and counting system. The system was chosen to ensure that election software is open, transparent, and available for public scrutiny by candidates and other participants in the electoral process. However, it is reported that the manufacturer has closed the source. (http://www.computerworld.com.au/index.php/id;537673802;fp;16;fpid;0)

Research on the subject of open source is also part of the Caltech/MIT Voting Technology Project's report, "What Is, What Could Be", July 2001, available at http://vote.caltech.edu. The report sought to assess the problems and causes of the problems that occurred in the 2000 presidential election and suggests new technology solutions. The section on Ballot Security offers several recommendations, including one to "make source code for all vote recording and vote counting processes open source" but to make the "source code for the user interface proprietary." (Page 42.) The report suggests, "that a national commission consisting of experts on security from outside the voting industry, including other industries such as banking and Internet security, should determine the appropriate protocol for open source in the voting equipment industry." (Page 46-47.)

The Secretary of State's Office hosted a Voting Systems Testing Summit in Sacramento on November 28 and 29, 2005, which brought together nationally recognized experts from local, state, and federal elections administrations, academia, research and public interest institutions, testing companies, and voting machine vendors to discuss testing as it applies to the state certification processes for voting equipment.

The Summit's nine sessions and panels, along with a keynote speaker, encompassed discussions on a wide range of topics, including the federal role in voting systems, the testing process conducted by the Independent Testing Authorities (ITA's), and voting systems security, paper trails, and accountability. Panelists included scientists, academics, representatives from the Election Assistance Commission (EAC), and other experts in the field of voting systems. Dr. Michael Shamos, Co-Director of the Institute for eCommerce and Director of the Center for Privacy Technology at Carnegie Mellon University participated in the Security/Paper Trails/Accountability Panel and offered his viewpoint that "all voting system software should be disclosed to the public."

In October of 2005, the National Center for Open Source Policy and Research was launched at a Government Open Source Conference in Portland, Oregon, where over 200 government entities and software developers met at the Oregon State University's Open Source Lab. They report that government has already taken advantage of open source code in development of a jail management system in Mississippi, and the United States Department of Defense is considering an open source approach to some research and development. They also report an initiative to partner with universities in research and development relating to open source code.

In addition, the Secretary of State convened a working group of state representatives to examine this issue on a national level, because often what occurs in California leads the nation. A conference call was held on November 10, 2005 with seven states and a representative from the National Association of Secretaries of State (NASS) participating. Of the states involved in this initial discussion, none have yet taken an in-depth look at open source software for their voting systems. It is clear that significant further examination and investigation are needed.

VII. Conclusion and Recommendations

The Secretary of State's Office has spent considerable time exploring the issue of open source software in an effort to arrive at recommendations that are both reasonable and viable. The list of available information on the subject is enormous. Expert opinions on the topic span the spectrum from unconditional support to serious opposition.

After extensive investigation, it is clear that significant further examination of open source software in voting systems is needed. While there may be benefits associated with using open source software, they are neither empirical nor measurable. There is no precedent that fully supports the feasibility of a potential effort to develop, deploy, and maintain an open source election system.

The creation and maintenance of secure, reliable, and usable voting systems is crucial to ensuring voter confidence. And while, to date, California has the most rigorous voting systems certification process in the nation, anything that furthers the integrity of the voting process deserves careful consideration. Decisions on this subject cannot be rushed, especially in an environment where technology is constantly evolving.

Unresolved issues include a number of determinations. For example, should open source be available for all to review as opposed to being restricted to certain individuals or groups of individuals; should the source code for the voting machines and vote tallying systems be disclosed or should access be restricted to the software for the voting systems? Also for consideration is the question of who should be empowered to test and critique such systems. In addition, if modifications are necessary, should there be a public review component incorporated into the process? These are but a few of the complex issues that must be fully reviewed before any recommendations relating to the use of open source code in voting systems can be provided.

In conclusion, the use of open source software in voting systems is a topic that requires substantial further review. It is a far more complex issue than simply determining if open source software should be used in voting systems. Therefore, the Office of the Secretary of State recommends further review of the issues and that the California State Legislature establishes a panel of experts from both the public and private sectors (including voting system manufacturers, academia, and others) to further evaluate the future of open source software in voting systems.