A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE)

January 21, 2004

Dr. David Jefferson, d_jefferson@yahoo.com
Dr. Aviel D. Rubin, rubin@jhu.edu
Dr. Barbara Simons, simons@acm.org
Dr. David Wagner, daw@cs.berkeley.edu

Executive Summary

This report is a review and critique of computer and communication security issues in the SERVE voting system (Secure Electronic Registration and Voting Experiment), an Internet-based voting system being built for the U.S. Department of Defense's FVAP (Federal Voting Assistance Program). The program's web site is http://www.serveusa.gov/. While the system is called an *experiment*, it is going to be used to count real votes in the upcoming general elections. The authors are members of SPRG (the Security Peer Review Group), a panel of experts in computerized election security that was assembled by FVAP to help evaluate SERVE. Our task was to identify potential vulnerabilities the system might have to various kinds of cyber-attack, to evaluate the degrees of risk they represent to the integrity of an election, and to make recommendations about how to mitigate or eliminate those risks.

The SERVE system is planned for deployment in the 2004 primary and general elections, and will allow the eligible voters first to register to vote in their home districts, and then to vote, entirely electronically via the Internet, from anywhere in the world. Besides being restricted to overseas voters and military personnel, SERVE is currently limited to people who vote in one of 50 counties in the seven states (Arkansas, Florida, Hawaii, North Carolina, South Carolina, Utah, and Washington) that are participating. The program is expected to handle up to 100,000 votes over the course of the year, including both the primaries and the general election. (By comparison, approximately 100 million votes were cast in the 2000 general election.) The eventual goal of SERVE is to support the entire population of eligible overseas citizens plus military personnel and their dependents. This population is estimated to number about 6 million, so the 2004 SERVE deployment must be judged as a prototype for a very large possible future system.

Our conclusions are summarized as follows:

a) DRE (direct recording electronic) voting systems have been widely criticized elsewhere for various deficiencies and security vulnerabilities: that their software is totally closed and proprietary; that the software undergoes insufficient scrutiny during qualification and certification; that they are especially vulnerable to various forms of insider (programmer) attacks; and that DREs have no voter-verified audit trails (paper or otherwise) that could largely circumvent these problems and improve voter confidence. All of these criticisms, which we endorse, apply directly to SERVE as well.

b) But in addition, because SERVE is an Internet- and PC-based system, it has numerous other fundamental security problems that leave it vulnerable to a variety of well-known cyber attacks (insider attacks, denial of service attacks, spoofing, automated vote buying, viral attacks on voter PCs, etc.), any one of which could be catastrophic.

c) Such attacks could occur on a large-scale, and could be launched by anyone from a disaffected lone individual to a well-financed enemy agency outside the reach of U.S. law. These attacks could result in large-scale, selective voter disenfranchisement, and/or privacy violation, and/or vote buying and selling, and/or vote switching even to the extent of reversing the outcome of many elections at once, including the presidential election. With care in the design, some of the attacks could succeed and yet go completely undetected. Even if detected and neutralized, such attacks could have a devastating effect on public confidence in elections.

d) It is impossible to estimate the probability of a successful cyber-attack (or multiple successful attacks) on any one election. But we show that the attacks we are most concerned about are quite easy to perpetrate. In some cases there are kits readily available on the Internet that could be modified or used directly for attacking an election. And we must consider the obvious fact that a U.S. general election offers one of the most tempting targets for cyber-attack in the history of the Internet, whether the attacker's motive is overtly political or simply self-aggrandizement.

e) The vulnerabilities we describe cannot be fixed by design changes or bug fixes to SERVE. These vulnerabilities are fundamental in the architecture of the Internet and of the PC hardware and software that is ubiquitous today. They cannot all be eliminated for the foreseeable future without

some unforeseen radical breakthrough. It is quite possible that they will not be eliminated without a wholesale redesign and replacement of much of the hardware and software security systems that are part of, or connected to, today's Internet.

f) We have examined numerous variations on SERVE in an attempt to recommend an alternative Internet-based voting system that might deliver somewhat less voter convenience in exchange for fewer or milder security vulnerabilities. However, all such variations suffer from the same kinds of fundamental vulnerabilities that SERVE does; regrettably, we cannot recommend any of them. We do suggest a kiosk architecture as a starting point for designing an alternative voting system with similar aims to SERVE, but which does *not* rely on the Internet or on unsecured PC software (Appendix C).

g) The SERVE system might appear to work flawlessly in 2004, with no successful attacks detected. It is as unfortunate as it is inevitable that a seemingly successful voting experiment in a U.S. presidential election involving seven states would be viewed by most people as strong evidence that SERVE is a reliable, robust, and secure voting system. Such an outcome would encourage expansion of the program by FVAP in future elections, or the marketing of the same voting system by vendors to jurisdictions all over the United States, and other countries as well.

   However, the fact that no successful attack is detected does not mean that none occurred. Many attacks, especially if cleverly hidden, would be extremely difficult to detect, even in cases when they change the outcome of a major election. Furthermore, the lack of a successful attack in 2004 does not mean that successful attacks would be less likely to happen in the future; quite the contrary, future attacks would be more likely, both because there is more time to prepare the attack, and because expanded use of SERVE or similar systems would make the prize more valuable. In other words, a "successful" trial of SERVE in 2004 is the top of a slippery slope toward even more vulnerable systems in the future. (The existence of SERVE has already been cited as justification for Internet voting in the Michigan Democratic caucuses.)

h) Like the proponents of SERVE, we believe that there should be better support for voting for our military overseas. Still, we regret that we are forced to conclude that the best course is not to field the SERVE system at all. Because the danger of successful, large-scale attacks is so great, we reluctantly recommend shutting down the development of SERVE immediately and not attempting anything like it in the future until both the Internet and the world's home computer infrastructure have been fundamentally redesigned, or some other unforeseen security breakthroughs appear.

We want to make clear that in recommending that SERVE be shut down, we mean no criticism of the FVAP, or of Accenture, or any of its personnel or subcontractors. They have been completely aware all along of the security problems we describe here, and we have been impressed with the engineering sophistication and skill they have devoted to attempts to ameliorate or eliminate them. We do not believe that a differently constituted project could do any better job than the current team. The real barrier to success is not a lack of vision, skill, resources, or dedication; it is the fact that, given the current Internet and PC security technology, and the goal of a secure, all-electronic remote voting system, the FVAP has taken on an essentially impossible task. There really is no good way to build such a voting system without a radical change in overall architecture of the Internet and the PC, or some unforeseen security breakthrough. The SERVE project is thus too far ahead of its time, and should not be reconsidered until there is a much improved security infrastructure to build upon.

1. Introduction

This report is a review and critique of computer and communication security issues in the SERVE voting system (Secure Electronic Registration and Voting Experiment), an Internet-based voting system being built by Accenture and its subcontractors for the U.S. Department of Defense's FVAP (Federal Voting Assistance Program). FVAP's mission is to reduce voting barriers for all citizens covered by the Uniformed and Overseas Citizens Absentee Voting Act (UOCAVA). UOCAVA covers Uniformed Services: U.S. citizens who are members of the Uniformed Services and their family members, and Overseas Citizens: U.S. citizens who reside outside the United States. Uniformed Services are defined as the U.S. Armed Forces (Army, Navy, Marines, Air Force and Coast Guard), merchant marine, commissioned corps of the Public Health Service and the National Oceanic and Atmospheric Administration.

A group of experts in computerized election security, called the Security Peer Review Group (SPRG) was assembled by FVAP to help evaluate SERVE. The task was to identify potential vulnerabilities the system might have under various kinds of cyber-attack, to evaluate the risks these attacks represent to the integrity of an election, and to make recommendations about how to mitigate or eliminate those risks.

The analysis and conclusions outlined here are based on two three-day meetings of the SPRG with the FVAP sponsors and the primary technical architects of SERVE; these were held in July, 2003 at Caltech in Pasadena, California, and in November, 2003 at Accenture in Reston, Virginia. The authors of this report consist of the subset of the SPRG that attended both meetings. Many issues and design improvements were proposed and accepted in those meetings; this report concentrates on the remaining security issues that were not resolved.

1.1 What is SERVE?

Part of the mission of FVAP is to reduce the barriers to registration and voting for two groups of eligible voters: (1) American citizens living outside the U.S., and (2) military personnel and their dependents, regardless of whether they reside in the U.S. or overseas. For Americans living overseas, voting can be a daunting task; it can take five or more trips through U.S. and foreign mail services to request voter registration forms and absentee ballots from the home county, to receive them, and then to send them back—a process that is time-consuming and unreliable at best, and must be accomplished in timely manner to avoid missing legal deadlines. The process is so clumsy for soldiers who are mobile, or who are located where mail service is poor, that their voting rates are believed to be quite low.

The SERVE system is planned for deployment in the 2004 primary and general elections, and is designed to allow UOCAVA voters both to register to vote in their home districts, and also to vote, entirely electronically via the Internet, from anywhere in the world. Although it is referred to by FVAP as an *experiment*, and is thought of that way by its developers, it is important to realize that it is not just a trial or mock voting system. SERVE will be a complete, medium-scale, federally-qualified and state-certified voting system front end, and it will collect real votes.

To participate, an eligible voter first must enroll in the SERVE program, which can be done completely electronically if the voter has suitable military ID (a Common Access Card), or by presenting suitable citizenship and ID documents face-to-face to a trusted agent, e.g. a military officer or other designated official who plays a role similar to that of a notary public. After enrollment, the voter will be able to register to vote, and then to vote, in one or two short sessions from any Internet-connected PC. The PC must run a Microsoft Windows operating system and either the Internet Explorer or Netscape web browser. The browser must be configured to enable JavaScript and either Java or ActiveX scripting, and it must also permit session cookies; however, no additional hardware or software is required.

SERVE is designed as a Web-based service. Voters connect to a central server using a standard browser, as just described. Both registration and voting are accomplished through the web interface. SERVE requires direct interaction between the voting service and the Local Election Official (LEO) in the voters' home precincts. Thus, when people register to vote, their information is stored on the central web server for later download by the LEO, at which point the LEO updates its database. When someone votes in the election, the completed ballot is stored on the central server, and later downloaded by the LEO, who stores it for later canvass.

The communication between the user's web browser and the voting application on the central server is protected using the encryption and authentication built into the Secure Socket Layer (SSL) protocol. Once that connection is established, an ActiveX control (described later) is downloaded to the voter's PC, because the voting application requires functionality that is not available in current standard browsers. For Netscape users, a Java applet runs that interprets the ActiveX. For Internet Explorer users, the ActiveX control runs natively on the voter's machine.

Besides being restricted to overseas voters and military personnel, in 2004 SERVE will be limited to people who vote in one of 50 counties in the seven states (Arkansas, Florida, Hawaii, North Carolina, South Carolina, Utah, and Washington) that have agreed to participate. The 2004 trial is expected to handle up to 100,000 votes over the course of the year, including both the primaries and the general election. (By comparison, approximately 100 million votes were cast in the 2000 general election.) However, one goal of SERVE is to determine if a similar system might be suitable for expansion in the future to all overseas voters. The total number of eligible overseas citizens plus military and dependents is estimated at about 6 million voters. Furthermore, systems similar to SERVE might eventually be offered by Accenture or other vendors for certification in many more states, and with all voters eligible to use it, instead of just a limited population. For these reasons we analyze SERVE not as an experiment, but as a real voting system whose use could be significantly expanded in future years.

1.2 Brief History of Internet Voting

The SERVE system is a follow-on to an earlier FVAP voting system called VOI (Voting over the Internet). VOI was built by a different general contractor (Booz-Allen & Hamilton) and used a different architecture and codebase, so VOI and SERVE can be compared only in a general way. VOI was used only in the 2000 general election, handling a total of 84 votes in four states (Florida, South Carolina, Texas, and Utah). All were real votes, not test ballots.

The FVAP office issued a report on the VOI system in June, 2001 (*Voting Over the Internet Pilot Project Assessment Report*). Only a small part of the report is devoted to security issues, but most of the concerns we mention below as applying to SERVE were mentioned in the VOI report. One conclusion of the report is that the VOI experiment was so small that it was not a likely target of any attacks, and that even a successful attack would almost certainly have made no difference in the outcome of any election. (The fact that 50 votes were cast in Florida using VOI, and that a change of 269 votes in the official tally of that state would have resulted in Al Gore becoming President, shows how dangerous such an assumption can be. We note that Florida is participating in 2004 in the SERVE program.)

VOI also ignored key parts of the Internet voting security problem by taking the position that "In the VOI Pilot System the citizen's workstation was outside the security perimeter of the system". In other words, VOI made no attempt to defend against some of the most serious attacks to which it was vulnerable.

However, the VOI report expressed concern about security problems with "remote" Internet voting (i.e., voting from any Internet-connected computer, anywhere in the world, as permitted under SERVE), and it explicitly declined to recommend remote voting until such time in the future when the most serious threats have been resolved:

> "[Remote Internet voting] is subject to the same security concerns as the current VOI System. For this reason, we cannot recommend [it] as an immediate follow-on development to the VOI Pilot. …. Therefore, we recommend that research continue on these security issues so that this alternative could be implemented in the future when adequate security measures are available to counteract the malicious software (e.g., virus and Trojan Horse) threat and denial of service attempts." (Section 6.2.4)

In spite of this recommendation, the SERVE program is deploying remote Internet voting as the follow-on to VOI, even though the malicious software, denial of service, and other threats have not been resolved.

In 2000 there were several other experiments with Internet voting in U.S. public elections. In some cases the votes counted officially; in others they did not. The largest and most well-known was the Arizona Democratic presidential primary, conducted by election.com (whose assets were acquired in 2003 by

Accenture) in March of that year, in which approximately 85,000 votes were cast and counted. The Reform Party national primary was also conducted over the Internet that summer, as were various nonbinding Internet voting experiments in some counties of Washington, California, Arizona and elsewhere.

Several studies of Internet voting, including the security issues, were conducted in 1999-2000. The first was by the California Secretary of State's Task Force on Internet Voting, whose report was issued in January, 2000 and is online at http://www.ss.ca.gov/executive/ivote. That report was the first to clearly articulate most of the technical security issues regarding Internet voting in general, and it conspicuously failed to recommend that the state push ahead with "remote" Internet voting (e.g., from PCs at home, office, schools, libraries, and cybercafes), because of the numerous security problems it detailed. Those security problems had no good solutions at the time, and now, four years later, they remain unsolved.

Another study was conducted by the Internet Policy Institute with funding from the National Science Foundation. Their report (*Report of the National Workshop on Internet Voting: Issues and Research Agenda,* referred to as the NWI Report) was based on a conference held in October, 2000, and was published in March, 2001. The following is a key paragraph from the Executive Summary of that report:

> "*Remote Internet voting systems pose significant risk to the integrity of the voting process, and should not be fielded for use in public elections until substantial technical and social science issues are addressed* [italics in the original]. The security risks associated with these systems are both numerous and pervasive, and, in many cases, cannot be resolved using even today's most sophisticated technology. In addition, many of the social science concerns regarding the effects of remote voting on the electoral process would need to be addressed before any such system could be responsibly deployed. For this reason, it is imperative that public officials educate themselves about the dangers posed by remote Internet voting, and the ramifications of failure on the legitimacy of the electoral process."[1]

The "numerous and pervasive" security risks referred to in the NWI Report are similar to those described in the California task force report.

The Caltech/MIT Voting Technology Project published its report (*Voting: What Is, What Could Be*) in July, 2001. It is available online at http://www.vote.caltech.edu/Reports/2001report.html. That report was similarly pessimistic about the security of Internet voting, stating that "Remote Internet voting poses serious security risks. It is much too easy for one individual to disrupt an entire election and commit large-scale fraud."

As we see, all three major studies of Internet voting in 1999-2001 concluded that Internet voting is fraught with major risks that have no short-term solution. We know of no major studies anywhere, then or since, that have come to any different conclusion. Those risks remain as relevant and dangerous today as ever.

In sections 2, 3 and 4 of this paper we argue that all of the security vulnerabilities that were articulated in these 1999-2001 studies are present in the SERVE architecture, along with at least one additional risk that was not emphasized then, namely potential insider fraud. We claim that, given current technology, these vulnerabilities are inherent in any Internet voting architecture that allows people to vote from private computers, and that no technological innovation will change that in the foreseeable future.

1.3 Why security for Internet voting is far more difficult than for e-Commerce

Many people mistakenly assume that since they can safely conduct commercial transactions over the Internet, that they also can safely vote over the Internet. First, they usually underestimate the hazards of

---

[1] The NWI report added a footnote stating "However, remote Internet voting may be appropriate in the near-term for special populations, such as the military and government employees and their dependents based overseas. Such exceptions should be evaluated on a case-by-case basis." However, the viruses, worms, Trojan horses and spyware that we've seen in recent years are much worse than those prevalent at the time of that report, and many of the concerns over insider fraud and the need for voter-verified audit trails have only been understood and articulated since then.

online financial transactions, and are unaware of many of the risks they take even if they are careful to deal only with "secure" web sites through the SSL protocol. But they also assume that voting is comparable somehow to an online financial transaction, whereas in fact security for Internet voting is far more difficult than security for e-commerce. There are three reasons for this: the high stakes, the inability to recover from failures, and important structural differences between the requirements for elections and e-commerce.

First, high security is essential to elections. Democracy relies on broad confidence in the integrity of our elections, so the stakes areenormous. We simply cannot afford to get this wrong. Consequently, voting requires a higher level of security than e-commerce. Though we know how to build electronic commerce systems with acceptable security, *e-commerce grade security is not good enough for public elections*.

Second, securing Internet voting is structurally different from—and fundamentally more challenging than—securing e-commerce. For instance, it is not a security failure if your spouse uses your credit card with your consent; it is routine to delegate the authority to make financial transactions. But it *is* a security failure if your spouse can vote on your behalf, even with your consent; the right to vote is not transferable, and must not be delegated, sold, traded or given away. Another distinction between voting and e-commerce is that while a denial of service attack on e-commerce transactions may mean that business is lost or postponed, it does not de-legitimize the other transactions that were unaffected. However, in an election, a denial of service attack can result in irreversible voter disenfranchisement and, depending on the severity of the attack, the legitimacy of the entire election might be compromised.

Third, the special anonymity requirements of public elections make it hard to detect, let alone recover from, security failures of an Internet voting system, while in e-commerce detection and recovery is much easier because e-commerce is not anonymous. In a commercial setting, people can detect most errors and fraud by cross-checking bills, statements, and receipts; and when a problem is detected, it is possible to recover (at least partially) through refunds, insurance, tax deductions, or legal action. In contrast, voting systems must not provide receipts, because they would violate anonymity and would enable vote buying and vote coercion or intimidation. Yet, even though a voting system cannot issue receipts indicating how people voted, it is still vital for the system to be transparent enough that each voter has confidence that his or her individual vote is properly captured and counted, and more generally, that everyone else's is also. There are no such requirements for e-commerce systems. In general, designing an Internet voting system that can detect and correct any kind of vote fraud, without issuing voters receipts for how they voted, and without risking vote privacy by associating voters with their votes, is a deep and complex security problem that has no analog in the e-commerce world. For these reasons, the existence of technology to provide adequate security for Internet commerce does not imply that Internet voting can be made safe.

1.4 Criteria for assessing the security of SERVE: How much security is enough?

In evaluating the security of SERVE, we need a standard against which to compare it, i.e. an answer to the question "How much security is enough?" We recognize that no security system is perfect, and it would be irresponsible and naive to demand perfection. However, since we must not allow unacceptable risks of election fraud to taint our national elections, we must have some set of criteria for deciding what risks are acceptable.

On the one hand, election security has to be viewed as a component of national security, since the very legitimacy of democratic government depends on elections that are fair, open, trustworthy, and seen to be so. This would argue for the very highest security standards—ideally that not a single vote be lost, forged, spoiled, miscounted, bought, or sold, and that the voter not be coerced or have his/her privacy compromised under any credible threat scenario, even if the attacker has significant resources, full knowledge of the SERVE architecture, and an inside confederate.

On the other hand, the SERVE system is designed to be a form of absentee voting for UOCAVA citizens. Absentee voters vote from somewhere other than the precinct polling location, traditionally by marking or punching a paper ballot and mailing it back to the county officials, although faxing is sometimes permitted. In some western states 30% or more of all votes are absentee; Oregon in particular has eliminated its precinct polling places, so all votes there are absentee. Since the goal of SERVE is to facilitate absentee voting, arguably the security level of absentee voting should be the baseline against

which SERVE is compared. Our analysis is therefore premised on the following principle: *At the very least, any new form of absentee voting should be as secure as current absentee voting systems.*

While absentee voting procedures offer a fair degree of security and privacy, there are some inherent vulnerabilities of which everyone is aware and that we as a society have agreed to tolerate. There are many ways for an attacker to compromise a *small* number of absentee votes without detection, e.g. by spying on or coercing voters, especially in institutional situations like nursing homes; or by paying voters; or by interfering with the transport of ballots through the mail, etc. However, the key point that makes absentee voting tolerable in spite of its many vulnerabilities is that it is very difficult to mount any kind of large-scale or automated attack on the process without getting caught. There are no single points of vulnerability through which many absentee votes pass except at the offices of county election officials and their local post offices. In both cases, there are numerous procedural and legal controls that, if adhered to, place perpetrators at great risk of getting caught. Even if a vote fraud scheme were perpetrated in one of those places and somehow escaped detection, its scope would be limited to a single county, and in most states to only the relatively small percentage of absentee ballots cast in that county, so that the risk and penalties for getting caught outweigh the value of the small number of votes that might be compromised.

This security level—some vulnerability to undetected small-scale attacks, but little or none to large-scale attacks—seems a reasonable goal for a new voting system such as SERVE, one that would assure that its addition to the mix of other voting systems already in use would not reduce the overall security that elections currently possess, nor add new vulnerabilities of a different character or scale. What we must avoid at all costs is any system in which it is possible for a successful large-scale or automated (computerized) attack to compromise many votes. It must be essentially *impossible* that any such large-scale attacks go undetected; or that such an attack might be so easy and inexpensive that a single person acting alone could carry it out; or that the perpetrator(s) of such an attack might never be identified; or that such an attack might be carried out remotely, from foreign soil, possibly by an foreign agency outside the reach of U.S. law, so that the attackers face little or no risk. Any voting system with any of these vulnerabilities is, we believe, completely unsuitable for use in U.S. public elections. Unfortunately, the SERVE system has all of these vulnerabilities, as we document in the rest of this report.

1.5 Voting System Threats

Any truly democratic voting system must have ways of dealing with five important threats. A first serious threat is disenfranchisement, either of individuals or of classes of voters. A major concern is that particular classes of voters could be disenfranchised, based on the likelihood of their voting a particular way. Internet voting provides opportunities for selective disenfranchisement that can be difficult to detect, and even harder to ameliorate.

A second threat is that a voter's ballot could be modified by a third party. With conventional paper ballots, this could be done by, say, adding a vote for an office for which the voter had not voted or by invalidating the voter's ballot by adding too many additional votes. As we shall see, however, electronic voting creates new opportunities for automated, widespread ballot modification that had not previously existed.

A third threat is the loss of privacy—the undermining of the secret ballot. Voting at a properly-managed precinct polling station on paper ballots that are mixed with others in a physical ballot box is the best protection for privacy. The privacy of a voting station or booth is protected by not allowing two people to enter a polling booth together, even if they want to. (An exception may be made for people with disabilities who are unable to vote unaided). Since the ballot is immediately mixed with other ballots in the box, it is virtually impossible to reconstruct who cast which ballot within a precinct. It is much harder to protect the absentee voter's privacy when voting is done using paper absentee ballots filled out at home or at work and sent through the mail, and harder still when an electronic absentee ballot is processed by a large amount of software on several different computers.

A fourth threat, a well-known type of voter fraud, is that a voter may vote more than once. Quoting from the Florida Department of Law Enforcement: Those inclined to do so can capitalize on others' access to an absentee ballot by voting their ballot for them, often with the actual voter not knowing what has occurred. This offers tremendous opportunity for vote fraud, particularly to those who have access to the ill or infirm or those who do not have the ability to resist the influence of another as they are urged to vote in a

"required" manner. It also encourages those inclined to commit voter fraud to seek to utilize absentee ballots provided to those whose interest in voting is marginal or non-existent.[2]

As in the case of ballot modification, Internet voting provides new opportunities for multiple voting by allowing those willing to invest time in social engineering to determine those registered voters who are most unlikely to participate in an election. After obtaining that information, identity theft would allow Internet voting to be automated so as to achieve a significant amount of false participation in any given election.

A fifth threat, intrinsic to absentee voting, is vote buying, selling, and trading. As we show, this threat also is amplified by Internet voting.

Finally, a theme that all these threats have in common is the issue of scale. Computers are extremely good at automating repetitive tasks, but this cuts both ways: It is also easy to use computers to automate attacks. When computer security systems fail, typically they fail on a large scale. A major risk in any centralized Internet voting scheme like SERVE is that a single failure could affect hundreds of thousands of voters.

1.6 Vulnerabilities in SERVE

Lack of voter-verified audit trail and insider attacks. DRE (direct recording electronic) voting systems have been widely criticized because they are essentially unauditable. First, there is no way that a voter can verify that the vote recorded inside the machine is the same as the vote that he or she entered and saw displayed on the machine's touchscreen. And later, if serious problems occur in the canvass of the votes (which happens all to frequently with DREs), there is no independent audit trail of the votes to help resolve the problem.

These issues have been debated widely around the nation in the last couple of years, and the computer security community, including all of the authors of this report, nearly unanimously supports the position that no DRE voting system should be permitted that does not have some sort of voter-verified audit trail. Voter verification is the only readily available effective defense against programmed insider attacks. Since so much has been written about this subject we will not repeat the arguments here. For further information see, for example, Prof. David Dill's web site www.verifiedvoting.org, or Prof. Rebecca Mercuri's site www.notablesoftware.com/evote.html, or the California Secretary of State's report and directives on touchscreen voting at www.ss.ca.gov/elections/touchscreen.htm. What we wish to note here is that every argument about the need for voter verification and auditability that have been made about DRE systems also apply essentially unchanged to the SERVE system. Indeed, from a voter's point of view, SERVE can be said to act as one giant DRE machine.

Privacy. SERVE aims to provide at least the same levels of privacy and security as conventional absentee voting. Since different states process absentee mail ballots differently, we shall discuss how California attempts to protect the privacy of the mail absentee ballot. The voter inserts the ballot in an inner envelope, which in turn is inserted into an outer envelope. Information identifying the voter, such as the voter's name and signature, is written on the outer envelope. When a mail ballot is received, the name and signature on the outer envelope are checked against the voter list. Assuming they match those of a registered voter who has not yet voted, the outer envelope is opened in the presence of at least two people. The timing of the opening of the inner envelope is determined by state and county regulations. While privacy concerns would dictate that the inner envelope be opened such that it cannot be linked to the outer envelope, this may not always happen. And even if the best precautions are taken, the voter who forgets to put the ballot in the inner envelope will have his/her ballot made visible as soon as the outer envelope is opened.

SERVE attempts to separate the name of the voter from his/her vote through the use of public key cryptography. In public key systems, each participant has a pair of keys consisting of a private key and a public key. The private part is known only to the participant, and, as the name implies, the public portion

[2] http://www.fdle.state.fl.us/publications/voter_fraud.asp

is available to everyone. In each voting district that is participating in SERVE, a local election official (called the LEO by SERVE) generates such a key pair. The LEO's public key is used to encrypt (scramble) the ballots of SERVE voters from that district. Once a ballot has been encrypted, it can be read only if it is decrypted through the use of the unique private key known only to the LEO.

When the voter uses SERVE to cast a ballot, his/her web browser sends the completed ballot, along with identifying information such as the voter's name, to a SERVE web server. This information is transmitted to SERVE in encrypted form, sent in a way that only the SERVE web server can decrypt it. Because the ballot is encrypted before transmission, if someone were to intercept the encrypted ballot en route, it would be impossible to decipher the actual vote. Note that, at this point, the LEO's key pair has not yet been used.

When the ballot is received, SERVE verifies that the voter is registered and has not yet voted. SERVE decrypts the ballot using the SERVE private key, separates the ballot from the voter's name, and then encrypts the ballot (without the voter's name) using the LEO's public key. Therefore, only the appropriate LEO will be able to decrypt the encrypted ballot. The encrypted ballot is stored for later transmission to the LEO. SERVE retains the encrypted ballot, even after a copy has been sent to the LEO. SERVE also places the voter's name on a list of people who have already cast a vote, so that they will not be allowed to vote a second time.

This architecture introduces several privacy risks. First, the LEO could deduce how voters in his/her precinct have voted by downloading votes from SERVE so frequently that they get at most one new vote and voter name each time. Recall that the LEO can request from SERVE a list of names of voters from the LEO's district who have already voted via the Internet and the list (re-ordered randomly) of encrypted ballots for those voters. If a curious LEO makes the request sufficiently often, it should be possible to infer how each individual voter voted, an obvious privacy risk.

Second, the fact that ballots exist unencrypted on the server for a brief period before being encrypted with the LEO's key introduces other privacy risks. For instance, the SERVE system administrators could view how people have voted. Also, if the SERVE machines were compromised, the unencrypted ballots could be revealed to unauthorized third parties. See Appendix A for further discussion of this risk.

Third, the fact that the encrypted ballots together with the voters' names are stored in a SERVE database means that anyone with access to a LEO's private key and with access to the SERVE database could determine the votes of all SERVE voters from that voting district, another significant privacy risk.

Fourth, the encrypted ballots are stored for a long period of time on SERVE's computers, which increases the window of risk. We understand that, at a minimum, encrypted ballots will be retained until 18 months after the end of elections. We believe it is possible that encrypted ballots may remain accessible for an even longer period of time, and perhaps indefinitely, for instance on backup tapes or other computers, independent of the intention of the developers. Indeed, information often remains retrievable longer than developers intend; in modern systems, information is typically copied to so many locations that it can be challenging to find and erase them all. Consequently, it is conceivable that voter privacy could be compromised at a future date if the information were to land in the wrong hands and old system keys were exposed. Mathematical breakthroughs in the future, for example, could expose old keys.

In Sections 2 and 3, we describe in detail several further risks to voter privacy. While today's absentee voting systems have their own privacy risks, taking all these risks as a whole, we believe that SERVE heightens the risk of large-scale privacy compromise.

Vote Buying/Selling. Vote selling is a problem in all elections, but it is a special concern for Internet voting, since the Internet can facilitate large scale vote buying and selling by allowing vote buyers to automate the process. During the 2000 presidential election we saw the first Internet based attempt at vote swapping in a presidential election with the creation of a website to facilitate vote swapping between Gore and Nader voters. While the Gore/Nader swapping depended on the honor system and no money changed hands, what's new about SERVE is that a similar approach could be used to provide *enforced* vote swapping or vote bartering services, or to purchase votes from SERVE voters. Due to the ease of automating such attacks, the deployment of SERVE could potentially lead to vote buying or swapping on

a larger scale than has been seen before.

The most straightforward vote-buying scheme would involve the selling of voting credentials, namely personally identifiable information and the voter's password or private key. One possible defense we considered would be for SERVE to prohibit the submission of multiple votes from the same Internet address. Restricting the number of submissions from any particular web address is not a strong defense, however, because it is possible for a purchaser of votes to fool SERVE into thinking that the votes were coming from different addresses. Also, due to proxy servers, legitimate users often appear to come from the same IP address, so this hypothetical defense may not be deployable in practice. An extreme example of this is AOL, which uses the same IP address for all users coming from its domain. Finally, it is possible that many servicemembers would share the same computer, so obviously, those votes would legitimately come from the same IP address.

Another approach to vote-buying would be for the buyer to provide the seller with a modified version of the ActiveX component being used by SERVE. An appropriately modified version could ensure that the voting is done according to the wishes of the vote purchaser. There does not seem to be any way for SERVE to defend against this style of vote buying. In short, the possibilities for large-scale, automated vote buying, selling, and swapping in SERVE go beyond anything present in existing absentee systems.

Intimidation. Intimidation is a potential problem with all forms of absentee voting, since the voter is not guaranteed the privacy of the voting booth. But it can be even more of a problem with Internet voting if the voter is not using his/her personal machine, since the owner of the machine may have installed software that would record what the voter is doing.

Large-Scale Impact. Because SERVE is vulnerable to many different types of attacks, a significant percentage of votes cast over the Internet are also vulnerable, and a single successful attack might be able to affect a large fraction of all votes cast through SERVE. By contrast, when voting is conducted at physical precincts on mechanical devices or with paper ballots, vote manipulation, to the extent it occurs, happens on a far smaller scale: No single attack is likely to affect a large number of votes. To make the comparison more explicit, a single teenager, hacker, or other malicious party could potentially affect tens or hundreds of thousands of votes cast through SERVE, while it is extremely unlikely that any single person could conduct vote fraud on such a large scale in existing non-electronic elections. Consequently, vulnerabilities with SERVE could have a far more significant impact than was possible before the introduction of computers and the Internet to the voting process.

Too Many Potential Attacks. Because there are many different kinds of attacks that could be conducted against SERVE, as we discuss below, it is essentially impossible to protect against them all. While any particular attack taken in isolation might have a mitigation strategy, the cost of the defense could be high and would be added to the cost of defending against all the other attacks that have been anticipated. Worse yet, defenses created to inhibit one kind of attack may amplify the risks of another. And of course an attack that has not been anticipated remains a serious risk.

Many Sources of Attacks. The Internet knows no national boundaries. Consequently, an election held over the Internet is vulnerable to attacks from anywhere in the world. Not only could a political party attempt to manipulate an election by attacking SERVE, but so could individual hackers, criminals, terrorists, organizations such as the Mafia, and *even other countries.* There is no need to postulate a large conspiracy or highly sophisticated adversaries; many of the attacks we describe could be mounted by lone individuals with college-level training in computer programming.

Undetectable Attacks. Election fraud has occurred in many different types of elections. A recent example involved boxes of paper ballots that were found floating in San Francisco Bay in November, 2001. There also have been elections in which deceased people have voted. Undoubtedly, many instances of election fraud have gone undetected. But, when there is a physical ballot, there is a chance that fraud, or even unintentional mistakes, can be corrected – or at least uncovered.

With Internet voting, however, there is no way to verify that the vote that has been received by SERVE accurately represents the intent of the voter, nor can the voter verify that his/her vote was received by SERVE and accurately recorded by the LEO. The mere presence of a confirmation screen does not prove

that the vote was recorded correctly. These concerns are analogous to those that have been expressed about DRE's without a voter-verified audit trail.

Detected fraud could be almost as worrisome as undetected fraud. If fraud that impacts the SERVE votes were to be detected, it's not obvious what would happen. A judge can order a new election within a jurisdiction, and states can decide how to deal with election related issues. But there is no provision for re-holding a federal election. If the 2004 election is as close as the 2000 election was, it is possible that SERVE votes might swing the election to one of the candidates. If there were reason to believe that those votes were unreliable or possibly manipulated, this could have an adverse impact on an already cynical public.

On-screen Electioneering. Many states have electioneering laws that prohibit any form of campaigning within some distance of a polling place. In California, for example, that distance is 100 feet. Yet there are no laws yet to prevent the worst kinds of electioneering inside a web browser window while someone is voting. For example, an ISP (or browser company, etc.) may derive revenue from advertising, like AOL, and would thus have the ability to target ads based on the IP address a user is connected to at the moment. These could take the form of pop-up ads or even ads within the browser window. The problem is that the moment when a voter connects to the SERVE vote server address (or a voter information site) he/she could be bombarded with all sorts of political ads. It is even possible that at least some forms of ads will end up being protected by the First Amendment, and then there will be no escaping them.

1.7 Organization of This Report

The remainder of this report is organized as follows. The technical core of the report analyzes three significant threats to the security of SERVE: attacks made possible by lack of control over the voting environment (Section 2), web spoofing and man-in-the-middle attacks (Section 3), and denial of service attacks (Section 4). Finally, we present our conclusions and recommendations (Section 5). Also, several appendices discuss additional issues, including other security risks of SERVE (Appendix A), a prior experience with Internet voting (Appendix B), a possible alternative to SERVE (Appendix C), and fundamental problems with writing secure and bug-free software (Appendix D).

The security threats to SERVE are summarized in Table 1, where we characterize the threats in terms of the skill level required to mount attacks, the consequences of successful attacks, whether or not the attacks are realistic, and finally the countermeasures that might be used to thwart them.

2. Lack of Control of the Voting Environment

Perhaps the greatest challenge with Internet voting arises from the fact that, in contrast to conventional elections, electoral authorities no longer have control over all the equipment used by voters. With SERVE's Internet voting system, voters can vote from home or elsewhere on their own computers or vote from other locations using computers controlled by third parties. As a result, hackers and other third parties might be able to gain control of a large number of computers used for voting, and election officials would be powerless to protect the integrity of the election. This facet of SERVE's Internet voting architecture poses significant risks to the security of elections. The lack of control of the voting machinery opens up three classes of attack: Compromise of the privacy of votes, disenfranchisement, and vote alteration. The next two sections describe how an attacker could gain control of the voting environment, and what he or she could do once that control is gained.

2.1 How an attacker could control the voting environment

There are two basic scenarios in which an attacker could control the voting environment: When a voter uses someone else's computer and when a voter's own computer contains malicious software. The latter could occur either because of pre-installed applications designed to attack the election, or because of malicious remote code, such as a worm or virus designed to exploit flaws in the Windows operating system or other applications.

If one votes at a cybercafe, the owners or system administrators of the cybercafe control the computer. In addition, a prior visitor to the cybercafe could have taken control of the computer and installed remote spying or subversion software. There are similar risks to voting from any shared computer such as those at

public libraries.

Threat	Skill needed	Consequences	Realistic?	Countermeasures
denial of service attack (various kinds)	low	disenfranchisement (possibly selective disenfranchisement)	common on the Internet	no simple tools; requires hours of work by network engineers; launchable from anywhere in the world
Trojan horse attack on PC to prevent voting	low	disenfranchisement	There are a million ways to make a complex transaction such as voting fail.	can mitigate risk with careful control of PC software; reason for failure may never be diagnosed
on-screen electioneering	low	voter annoyance, frustration, distraction, improper influence	trivial with today's web	nothing voter can do to prevent it; requires new law
spoofing of SERVE (various kinds)	low	vote theft, privacy compromise, disenfranchised voters	Web spoofing is common and relatively easy	none exist; likely to go undetected; launchable by anyone in the world
client tampering	low	disenfranchisement	one example: change permissions on cookie file. Many other trivial examples	none exist for all possible mechanisms. Too difficult to anticipate all attacks; likely never diagnosed.
insider attack on system servers	medium	complete compromise of election	Insider attacks are the most common, dangerous, and difficult to detect of all security violations	none within SERVE architecture; voter verified ballots needed, e.g. Appendix C; likely undetected
automated vote buying/selling	medium	disruption of democracy	very realistic, since voter willingly participates	none exist; buyers may be out of reach of U.S. law
coercion	medium	disruption of democracy	harder to deploy than vote buying/selling, but man in the middle attacks make it achievable with average skill	none exist; likely to go undetected.
SERVE-specific virus	medium or high	vote theft, privacy compromise, disenfranchised voters	Some attacks require only experimentation with SERVE; others require leak of SERVE specs or code and resourceful attacker	virus checking software can catch known viruses, but not new ones; likely to go undetected
Trojan horse attack on PC to change votes or spy on them	high	vote theft, privacy compromise	widely available spyware would be a good starting point	can mitigate risk with careful control of PC software; harder to control at cybercafe, or other institutionally managed networks; attack likely to go undetected

Table 1 This table describes, for each potential threat to SERVE, what skill is required by the attacker, the consequences of a successful attack, how realistic the attack is, and what countermeasures might be used to thwart the attack.

If one votes at work, the employer controls the computer. A study found that 62% of major US corporations monitor employee's Internet connections, and more than one-third store and review files on

employees' computers.[3] While monitoring Internet connections by passively sniffing would not impact the security of SERVE, since the system uses the Secure Socket Layer (SSL) protocol which encrypts traffic, any monitoring that is based on software running on the employee's computer could be used for malicious purposes. An employer is also in a position to coerce employees who vote at work into voting a certain way.

Software running on a voter's computer also poses risks. Backdoors, placed in software and activated when a user tries to vote, can invisibly monitor or subvert the voting process. The prevalence of so-called *Easter Eggs* in many popular software packages demonstrates that this is a real possibility. (Easter Eggs are cute extras that a software developer adds to the application without authorization, for fun. One well-known example: Microsoft's Excel 97 spreadsheet application contains a full-fledged flight simulator that can be launched using a secret sequence of keystrokes.)

Today's computers come loaded with software developed by many different entities; any employee at any of those companies could conceivably leave a backdoor that attacks SERVE. Operating systems, games, productivity applications, device drivers, multimedia applications, browser plug-ins, screen savers, and Microsoft Office macros are all possible carriers. Every time someone downloads new software, the risk is increased.

In addition to the threat from pre-installed applications, there is a threat from remote attackers. Such an attacker might gain control of a computer without being detected. For example, an attacker could exploit a security vulnerability in the software on a voter's computer. The attacker could then take remote control of the machine using any number of products. Examples of remote control software are PCAnywhere and BackOrifice. It is an unavoidable fact that today's computing systems are inadequate to protect against this threat. Successful penetration of even well-defended computers is routine and common.

Voters' home computers are unlikely to be as carefully defended as corporate ones, and hence voters' machines are especially susceptible to attack. Attacks can be easily automated; hackers routinely scan thousands or even millions of computers in search of those that are easiest to compromise. We can envision scenarios in which the computers of SERVE voters have been compromised on a large scale, calling into question all votes cast over the Internet. Regrettably, such a scenario is all too possible.

Remote attacks might be spread using any of a number of attack vectors. Perhaps most fearsome is a virus or worm that spreads itself and contains a malicious payload designed to take control of machines and wreak havoc with a future election. Since virus checking software programs defend against only previously known viruses, virus checkers often are unable to keep up with the spread of new viruses and worms. Consequently, malicious worms are widespread among Internet-connected computers today. For instance, in 2001, the Code Red worm infected 360,000 computers in 14 hours, and in 2003 the Slammer worm brought down many ATM machines and compromised many Internet hosts.[4] Modern worms are even more virulent, are often spread by multiple methods, are able to bypass firewalls and other defenses, and can be difficult to analyze. For example, it took quite a while to realize that SoBig.F was a large-scale Trojan horse designed to plant spam engines.[5]

The threat of SERVE-specific viruses should not be discounted. The first question that comes to mind is: "Can virus checking software prevent this threat?" The answer, we believe, is "No." New viruses almost certainly will not be detected by most current virus checking software. Moreover, it is not too difficult for attackers to build new viruses, or to modify existing viruses sufficiently that they will avoid detection. One can even find virus construction kits on the Internet. In addition, the attacker has the advantage that he/she can test new versions of viruses using the publicly available virus checkers that potential victims use to confirm that the virus will not be detected before its release. In our experience, new viruses usually spread rapidly until their signature is known and the anti-virus companies update their definition files, but this may be too late: the damage to the election may already have been done.

The worm threat is also quite real. It is easy for any competent programmer to write a crude worm; the

[3] http://www.amanet.org/research/pdfs/ems_short2001.pdf
[4] http://securityresponse.symantec.com/avcenter/venc/data/w32.sqlexp.worm.html
[5] http://securityresponse.symantec.com/avcenter/venc/data/w32.sobig.f@mm.html

source code to previous worms can be obtained and modified to create new worms. Writing a sophisticated worm is substantially harder. One set of experts estimates that a small team of experienced programmers could, after months of work, develop a worm that might compromise the majority of all Internet-connected computers within a few hours [SPW02]. We don't know if such an ambitious project would succeed on the first attempt, and there seems to be no clear consensus within the security community on how long such a worst-case worm could remain undetected. Some argue that it would be detected within hours or days, while others argue that it may be possible to conceal its existence for weeks or even longer. In any case, worms remain a significant risk. A smaller-scale worm that more selectively targeted a smaller population would be much harder to detect, and possibly could evade detection indefinitely. Even an unsuccessful widespread attack could damage voter confidence.

There are other ways that attacks might spread. One possible attack involves scanning a large number of computers and attacking them directly. This technique is widely used by hackers. We might also see attacks that are spread by the inclusion of malicious worms in email in order to influence a SERVE election. Elections also could be undermined through the use of web sites that are altered to contain malicious content. Any user visiting such a web site would have his or her computer invisibly hijacked. Because the SERVE voting system requires that voters enable certain dangerous features on their computers, the risk of web site attacks may be heightened. For instance, SERVE supports only Microsoft Windows, a platform that has suffered from many security problems; also, SERVE requires that voters enable ActiveX scripting, cookies, Java, and JavaScript—web technologies that pose significant risk to the security of voters' computers.

ActiveX scripting is a Microsoft technology that allows code from the Internet to run natively on client machines. There is a security architecture for ActiveX that is outside the scope of this report. As stated earlier, SERVE requires ActiveX because some of the functionality needed in the system cannot be achieved inside a browser. However, the use of ActiveX introduces additional vulnerabilities into the system, as shown below.

A dangerous hybrid attack involves placing malicious content on specially chosen websites. For instance, an attacker with a vendetta against one candidate might booby-trap the website of that candidate, so that those who visit the candidate's website are unable to vote using SERVE. Such selective disenfranchisement might eliminate several hundred votes for a candidate, thereby throwing the election to his/her opponent.

There are several ways to booby-trap a website or email that do not pose any major technical difficulties to the attacker. One simple method is to place on the website or in the email a malicious ActiveX control that, when viewed, changes the voter's machine so that it will not work with the SERVE voting system. (We give some examples in the next section.) For a malicious ActiveX control to run, it must be marked as *trusted*.[6] Any programmer who becomes a valid publisher, i.e., whose public key is signed by Microsoft, Verisign, GTE, Thawte or a corporate signing authority, can produce code that is implicitly trusted by the Windows operating system. There have been documented cases of people tricking Microsoft into signing a malicious ActiveX control.

Targeted attacks could be applied either on a large scale or on a small scale. There could be email-driven or web-driven attacks that affect hundreds of thousands or even millions of users: the extraordinary prevalence of spam is testament to the leverage available through email-driven attacks. For instance, these techniques might allow one party's partisan to prevent a large fraction of the other party from voting, while leaving most or their own party unaffected. Although it may be possible to build sophisticated email- or web-driven attacks that evade detection, it seems likely that such stratagems would be noticed if employed on a large scale. Nonetheless, even if such an attack were to be detected, there may be little one could do beyond invalidating the entire election, hardly a desirable outcome.

2.2 What the attacker can do with control of the voting environment

An attacker with control over the voter's computer is in position to observe how someone votes,

[6] Code is marked as *trusted* by applying a digital signature with a private key whose public counterpart is part of the Windows system, or whose public component has been certified by Microsoft.

compromising the secrecy of the vote. For instance, such an attacker might place spy software on the computer to silently record all actions taken by the voter. Today, spy software is readily available on the commercial market to record all keystrokes typed, websites visited, and actions taken by the user. What is surprising is not that such software exists, but that it is readily affordable to all; one can find such software for under $50, as well as dozens of free versions. Moreover, if an attacker wants to monitor many voters at once, the software could be customized without too much difficulty to target SERVE elections specifically. Worse, the average computer user would have no way to detect whether third parties have observed his/her vote.

An attacker with control of the voting environment could disable ActiveX or Web cookies, for example, by changing the permissions on the cookie file to disallow write access, so that the user could no longer vote through SERVE. Such an attack is easy to mount. A clever attack could be designed in such a way that the user could not re-enable the necessary permissions. In this case the voter would realize that he/she had been disenfranchised. More sophisticated attacks might cause disenfranchisement in a way that the average voter would not detect.

Targeted voter disenfranchisement poses a serious threat to the integrity of the election. It is possible to imagine widespread attacks that targeted all voters in a particular party for disenfranchisement, leaving the other party unaffected. Such an attack would have serious consequences.

While the ease of selective disenfranchisement is a serious concern, another risk is that a malicious third party with control of a registered voter's computer could use that control to cast an unauthorized ballot, thereby violating the integrity of the election. The fraudulent ballot could appear to come from the authorized voter but in reality be filled out by the attacker. Or, an attacker could wait to see how the voter votes and then change the ballot cast by the voter before it is processed by the ActiveX control from SERVE. Such an attack would require some sophistication, but is not impossible. There are several safeguards in the SERVE design that would hinder this attack. But once the attacker has control of the client, these safeguards could be overcome. The easiest technique might be to modify the ActiveX control running on the machine so that the SERVE safeguards are intercepted and not seen by the user. The *patched* ActiveX control would act as a man in the middle (see below), giving the voter the experience he/she expects when voting legitimately, but modifying the vote. In one possible scenario, the attacker might allow the vote to proceed unmodified if it is to the attacker's liking, modifying or discarding it otherwise.

Privacy compromise, disenfranchisement, and vote fraud potentially could be perpetrated without anyone noticing. The voter might not be aware that his/her vote has been monitored or subverted by a malicious third party. Likewise, election officials would likely not have any way to detect the behavior of such malicious third parties.

3. Spoofing and Man in the Middle Attacks

Despite all of the safeguards incorporated in the SERVE system, there are certain attacks that cannot be prevented. In this section, we describe the vulnerability of SERVE to the compromise of voter privacy and ultimately to the subversion of the election. We describe the attacks in increasing order of severity.

The first attack we describe is a man in the middle attack that compromises voter privacy. A man in the middle attack is one in which the adversary interposes itself between the legitimate communicating parties and simulates each party to the other party. To simplify the discussion, we focus on vote privacy, where an attacker seeks to learn how various people voted. The ability of an arbitrary outsider to learn on a wide scale how voters voted is enough of a threat to democracy that we think this alone justifies canceling the SERVE project. The fact that the attack is relatively easy to mount only strengthens our claim.

There are several ways that an adversary could become a *man in the middle*:

- **Control the client machine:** As described in the previous section, if an adversary can control the voting machine, then the adversary can act as a man in the middle to control the vote, even on an encrypted session.
- **Control the local network:** If the attacker has control over the local network environment, such as an employer in a workplace or anyone who shares a wireless network, then the attacker can

interpose him/herself as a man in the middle of any network communications.

- **Control an upstream network:** An ISP or foreign government that controls network access from the voter to the voting server could masquerade as a man in the middle.
- **Spoof the voting server:** Even without physical access to the network path between a voter and the voting server and without access to the machine, there are social engineering attacks where a voter can be fooled into thinking that he or she is communicating with the voting server. This attack could be implemented, for example, by posting or emailing a link that appears to go to the voting server, but in fact does not.
- **Attack the Domain Name Service (DNS):** Attacks against the DNS could route traffic to an attacker instead of to the legitimate vote service.

Clearly, there are many ways that an attacker could become a man in the middle. The use of SSL does little to mitigate the man in the middle attack if the only goal is to compromise privacy. Any man in the middle could act as an SSL gateway, forwarding application data between the voter and the vote server unaltered. The attacker would be able to see all of the traffic by decrypting and re-encrypting it as it passes between the two. In effect, the attacker would communicate using two SSL sessions, one between itself and the voter, and the other between itself and the vote server, and neither would know that there was a problem.

One attempt to prevent a man in the middle attack would be for the ActiveX control from the voting server to sign the IP address of its SSL endpoint along with the completed ballot. However, this is not a good defense: It would be easy for a man in the middle to defeat this countermeasure by applying a simple patch to the ActiveX control as it traveled to the voter from the server. The patch would hard code the correct IP address into the right spot for the signature. Of course, the attacker would also have to re-sign the patched ActiveX control; becoming an ActiveX signer requires fooling one of the certifying authorities, or simply purchasing a key. Browsers come with over one hundred default keys that are already completely trusted, whether the users know it or not.

We carefully analyzed the SERVE system against this attack and conclude that the attack would be relatively straightforward to mount, and that it could be successful.

In the process of analyzing privacy, we discovered another vulnerability of SERVE, where the system could be used for vote selling as follows. The vote seller sets up a proxy server, as described above, and draws voters to his site. In this case the voters willingly connect to the attacker, with the result that the voter's privacy could be compromised. Since the attacker could see how people voted, the vote selling scheme could be verified. Similarly, attackers intent on coercing voters could use the same techniques to learn how the victims had voted, increasing the capability to coerce voters.

Man-in-the-middle attacks could also be used to disenfranchise voters, further raising the level of seriousness of this vulnerability. Once an attacker can act as a man in the middle, the attacker can completely shut out the vote server and spoof the entire interaction with the voter. Although SERVE has some safeguards in place, they assume that the voter knows exactly what to expect from the voting experience; it is probably safe to assume that an attacker could create a voting experience that the voter would believe was real. Once the attacker can spoof the election service, voters are completely disenfranchised. The attacker could make them think that they voted, and the voters will not know that their communications never reached the vote server. One safeguard in SERVE is that voters can check to see if their votes were recorded. It is not clear what the procedure would be if after the election, a large percentage of absentee voters said that they had voted but did not see their names. More likely is the possibility that few voters would bother to check. In either case, the election would be greatly disrupted.

Similar attacks could work against the registration process. Voters could be led to believe that they registered successfully, when in fact they were communicating directly with the adversary and not interacting with the legitimate registration server. The voters would discover when attempting to vote that they were not registered, which could be very disruptive.

Perhaps the most serious consequence of man in the middle attacks is that attackers could engage in election fraud by spoofing the voting server and observing how the voter votes. If a vote is to the attacker's liking, an error message is given and the voter is redirected to SERVE's legitimate voting site; in this

case, the vote will be counted. If the attacker does not like the vote, then the entire voting session is spoofed; in this case, the user thinks he/she has voted, but in fact the vote was never received by SERVE and will not be counted. For instance, an attacker could arrange that votes for one candidate will be received and counted by SERVE, while votes for other candidates will never be seen or counted by SERVE. Thus, the attacker could use the privacy compromise described above to actually subvert the outcome of the election.

While the designers of SERVE are very talented and while they attempted to mitigate many of the threats posed by Internet voting, man in the middle and spoofing attacks remain threats that are not overcome in the system. It is not clear to us how one would avoid such threats in the current Internet environment. This forms part of the basis for our conclusion that Internet voting cannot be made secure for use in real elections for the foreseeable future.

4. Denial of service attacks

If a hacker could overload the election web server and prevent citizens from voting, the integrity and meaningfulness of the election would be compromised. Such attacks, where legitimate users are prevented from using the system by malicious activity, are known as *denial of service attacks*. We believe that denial of service attacks are a serious risk for SERVE.

Denial of service attacks are possible in everyday life. For instance, flooding a victim's telephone number with a deluge of unwanted phone calls can pose enough of a disruption that the victim disconnects his/her telephone—thereby becoming unreachable to legitimate callers. On the Internet, denial of service attacks are often much more devastating, because Internet denial of service attacks can be automated with a computer, and because such attacks can often be mounted untraceably over the Internet.

A particularly nasty variant of denial of service attack is the *distributed denial of service* (DDoS) attack. In a DDoS attack, many attacking machines collaborate to mount a joint attack on the target. In this scenario, an attacker could take control of many computers in advance by spreading a custom-crafted virus or worm. In computer security jargon, the compromised machines are often known as "zombies" or "slaves," because the attacker leaves behind hidden software that causes infected machines to blindly obey subsequent commands from the attacker. "Zombie networks" are widely used by hackers today to mount denial of service attacks (and to send spam).

Denial of service is not a theoretical threat; the risk is all too real. Denial of service attacks have become a steadily growing nuisance over the past several years. Automated tools for mounting DDoS attacks have been circulating among the hacker community since at least 1999 [HW01], and hackers routinely amass large "zombie networks" of compromised machines. In February 2000, major DDoS attacks were mounted against several high-profile web sites, including CNN, Yahoo and eBay.[7] It was later discovered that these damaging attacks had been perpetrated by a lone teenager not on US soil.[8]

Since then, DDoS attacks have become routine. One study recorded over 10,000 denial-of-service attacks during a three-week period in 2001 [MVS01]. In 2001, the Code Red worm infected 360,000 computers in 14 hours; it contained code to mount a DDoS attack on the White House website. (Fortunately, the DDoS attack was deflected at the last minute).[9] In 2003, an Internet election in Canada was disrupted by a denial of service attack on Election Day.[10] These are not isolated examples; it is all too easy to mount DDoS attacks, and the culprits are rarely caught.

4.1 How an attacker could mount a denial-of-service attack

Broadly speaking, there are two major forms that Internet denial of service attacks can take. In the first category are attacks in which an adversary is able to swamp the network connection of a targeted web server with junk data that clogs up the network and prevents other, legitimate traffic from getting through. The second category includes attacks in which the adversary is able to overload the web server's computational

[7] http://www.nipc.gov/investigations/mafiaboy.htm
[8] http://www.cnn.com/2000/TECH/computing/04/18/hacker.arrest.01/
[9] http://www.symantec.com/avcenter/venc/data/codered.worm.html
[10] http://cbc.ca/stories/2003/01/25/ndp_delay030125

resources with useless tasks that keep it busy; if the web server becomes too busy, it may be unable to respond to connections from legitimate users. We will discuss both categories.

In a network flooding attack, the adversary sends huge volumes of data in the direction of the victim, saturating the victim's network connection and making it impossible for legitimate users to contact the victim. SERVE's websites are at risk for this kind of attack; if their network connection is overwhelmed by a denial-of-service attack, then eligible voters will not be able to vote using SERVE.

As a rule of thumb, the robustness of a website against network flooding attacks is determined largely by the network capacity available to that website. For instance, a website with a 1 Gbps link to the Internet would be hard-pressed to withstand a 1 Gbps DDoS attack. Large e-commerce sites typically have a 10 Gbps link at most. In comparison, researchers have observed DDoS attacks with peak traffic rates in excess of 150 Gbps [MVS01,MPSSSW03]. It seems unlikely to us that SERVE could withstand such a high-volume DDoS attack.

In the second category of denial-of-service attack, the adversary sends many valid-looking requests to the victim in an attempt to overload the victim's computer and keep it busy with useless work. There are many opportunities for attacks of this sort, and it would be hard to anticipate them all. Instead, we will describe an example of one attack in this category, to give the general idea. Similar principles may apply to many other aspects of the SERVE architecture.

SERVE uses an SSL-protected website. However, SSL is susceptible to a denial-of-service attack. An adversary could send many requests to initiate new SSL connections, and the SSL protocol requires the recipient to perform a slow cryptographic operation (typically an RSA private-key computation) when responding to each such request. The exact performance depends on the security level provided, but with the fastest and lowest-security key sizes that are today considered acceptable (i.e., 1024-bit RSA), modern computers can handle about 100 new connections per second; hardware accelerators raise this number to thousands of new connections per second. Today's largest e-commerce sites can handle up to 15,000 new SSL connections per second. In comparison, an attacker might be able to initiate about 500,000 new SSL connections per second, based on the following assumption: It is plausible that an attacker could gather a "zombie network" of 10,000 slave computers, and each computer could initiate about 50 new SSL connections per second. Consequently, an attacker could generate 10 to 100 times more SSL traffic than the SERVE website is likely to be able to handle. Thus, a DDoS attack against SERVE's SSL web servers could render SERVE unreachable to voters and disrupt an election in progress.

Unfortunately, mitigating or responding to denial-of-service attacks is very difficult. Today's technology is not up to the task. For example, no good defenses against network flooding attacks are known on today's Internet. It may be possible to defend against the particular SSL attack we describe; however, defending against all variants of this scheme is difficult. As an attacker will attack the weakest link in any system, SERVE must protect against all possible denial-of-service attacks—a very difficult task.

In summary, we are concerned that, no matter how much energy is invested into defensive countermeasures, adequate protection against denial-of-service attacks is unattainable with the technology available today. No matter how careful the designers are, SERVE is unavoidably at risk.

4.2 The implications of denial-of-service attacks for SERVE

An attacker could mount a large-scale denial of service attack that renders SERVE's voting service unavailable on the day of an election. Those voting on Election Day would be unable to vote, calling into question the validity of the election.

Alternatively, network services could be knocked out or degraded for areas where a particular demographic is known to vote for a particular party. The outcome of the election could be swayed by such an attack. Detection of such a selective disenfranchisement attack would be possible, but it is not clear how to respond; once polls close, there may be no good choices.

Though today's absentee voting process already disenfranchises some voters, we fear that SERVE could make the problem worse, not better. We recognize that UOCAVA voters are having a harder time voting than they ought to, and there are reasons to believe that a significant number (perhaps 20-30%, according to

some estimates) of military voters fail in their attempt to vote absentee. However, SERVE runs the risk of exacerbating these problems. With SERVE, there is the possibility that the disenfranchisement rate could rise to close to 100%, if a denial of service attack is successfully mounted against SERVE. In addition, SERVE creates the risk of large-scale selective disenfranchisement, which is not present in today's absentee system. Large-scale selective disenfranchisement is especially problematic because it could be used to influence the outcome of an election.

One important difference between SERVE and in person voting is that eligible voters can vote at any time during a 30-day window starting 30 days before Election Day and extending until the close of polls on Election Day. If voters could be persuaded to vote early in this time window, the impact of denial-of-service attacks might be reduced: In the past, most denial-of-service attacks have lasted only for a few days, and when the attack subsides, affected voters could then vote, if the polls had not yet closed.

However, SERVE's 30-day window cannot be relied upon to defend against denial-of-service attacks. There are reasons to believe that a large proportion of the voting population will want to vote on Election Day. (See Appendix B for an example). This introduces the threat of *last-day denial-of-service attacks* in which the attacker mounts a denial-of-service attack starting on the morning of Election Day and lasting until polls close. Because responding to denial-of-service attacks takes time, it is likely that an attacker would be able to maintain a last-day denial-of-service attack all day long, so that the SERVE systems would remain unreachable for the entire Election Day. In such a scenario, any overseas citizen who had intended to vote on Election Day would be unable to vote through SERVE, probably would not be able to find any alternative way to vote before the close of polls, and thus would be disenfranchised.

We expect that last-day denial-of-service attacks would disenfranchise a substantial fraction of the SERVE population. There seems to be little that SERVE can do to defend against such attacks. For these reasons, we consider last-day denial-of-service attacks a significant threat to the security of SERVE's elections.

5. Conclusions

Our conclusions, based on the arguments in this report are summarized as follows:

a) DRE (direct recording electronic) voting systems have been widely criticized elsewhere for various deficiencies and security vulnerabilities: that their software is totally closed and proprietary; that the software undergoes insufficient scrutiny during qualification and certification; that it is especially vulnerable to various forms of insider (programmer) attacks; and that DREs have no voter-verified audit trails (paper or otherwise) that could largely circumvent these problems and improve voter confidence. All of these criticisms, which we endorse, apply directly to SERVE as well.

b) But in addition, because SERVE is an Internet- and PC-based system, it has numerous other fundamental security problems that leave it vulnerable to a variety of well-known cyber attacks, (insider attacks, denial of service attacks, spoofing, automated vote buying, viral attacks on voter PCs, etc.), any one of which could be catastrophic.

c) Such attacks could occur on a large scale, and could be launched by anyone from a disaffected lone individual to a well-financed enemy agency outside the reach of U.S. law. These attacks could result in large-scale, selective voter disenfranchisement, and/or privacy violation, and/or vote buying and selling, and/or vote switching even to the extent of reversing the outcome of many elections at once, including the presidential election. With care in the design, some of the attacks could succeed and yet go completely undetected. Even if detected and neutralized, such attacks could have a devastating effect on public confidence in elections.

d) It is impossible to estimate the probability of a successful cyber-attack (or multiple successful attacks) on any one election. But we show that the attacks we are most concerned about are quite easy to perpetrate. In some cases there are kits readily available on the Internet that could be modified or used directly for attacking an election. And we must consider the obvious fact that a U.S. general election offers one of the most tempting targets for cyber-attack in the history of the Internet, whether the attacker's motive is overtly political or simply self-aggrandizement.

e) The vulnerabilities we describe cannot be fixed by design changes or bug fixes to SERVE. These vulnerabilities are fundamental in the architecture of the Internet and of the PC hardware and software that is ubiquitous today. They cannot all be eliminated for the foreseeable future without some unforeseen breakthrough. It is quite possible that they will not be eliminated without a wholesale redesign and replacement of much of the hardware and software security systems that are part of, or connected to, today's Internet.

f) We have examined numerous variations on SERVE in an attempt to recommend an alternative system that may deliver somewhat less voter convenience in exchange for fewer or milder security vulnerabilities. However, all such variations suffer from the same kinds of fundamental vulnerabilities that SERVE does; regrettably, we cannot recommend any of them. We do suggest a kiosk architecture as a starting point for designing an alternative voting system with similar aims to SERVE, but that does *not* rely on the Internet or on unsecured PC software (Appendix C).

g) The SERVE system might appear to work flawlessly in 2004, with no successful attacks detected. It is as unfortunate as it is inevitable that a seemingly successful voting experiment in a U.S. presidential election involving seven states would be viewed by most people as strong evidence that SERVE is a reliable, robust, and secure voting system. Such an outcome would encourage expansion of the program by FVAP in future elections, or the marketing of the same voting system by vendors to jurisdictions all over the United States, and other countries as well. (The existence of SERVE has already been cited as justification for Internet voting in the Michigan Democratic caucuses.)

   However, the fact that no successful attack is detected does not mean that none occurred. Many attacks, especially if cleverly hidden, would be extremely difficult to detect, even in cases when they change the outcome of a major election. Furthermore, the lack of a successful attack in 2004 does not mean that successful attacks would be less likely to happen in the future; quite the contrary, future attacks would be more likely, both because there is more time to prepare the attack, and because expanded use of SERVE or similar systems would make the prize more valuable. In other words, a "successful" trial of SERVE in 2004 is the top of a slippery slope toward even more vulnerable systems in the future.

h) Like the proponents of SERVE, we believe that there should be better support for voting for our military overseas. Still, we regret that we are forced to conclude that the best course is not to field the SERVE system at all. Because the danger of successful, large-scale attacks is so great, we reluctantly recommend shutting down the development of SERVE immediately and not attempting anything like it in the future until both the Internet and the world's home computer infrastructure have been fundamentally redesigned, or some other unforeseen security breakthroughs appear.

We want to make clear that in recommending that SERVE be shut down, we mean no criticism of the FVAP, or of Accenture, or any of its personnel or subcontractors. They have been completely aware all along of the security problems we have described here, and we have been impressed with the engineering sophistication and skill they have devoted to attempts to ameliorate or eliminate them. We do not believe that a differently-constituted project could do any better job than the current team. The real barrier to success is not a lack of vision, skill, resources, or dedication; it is the fact that, given the current Internet and PC security technology, and the goal of a secure, all-electronic remote voting system, the FVAP has taken on an essentially impossible task. There really is no good way to build such a voting system without a radical change in overall architecture of the Internet and the PC, or some unforeseen security breakthrough. The SERVE project is thus too far ahead of its time, and should not be reconsidered until there is a much improved security infrastructure to build upon.

Acknowledgements

We thank Kim Alexander, Dr. Steve Bellovin, Lillie Coney, Prof. David Dill, Prof. Doug Jones, Yoshi Kohno, Prof. Deirdre Mulligan, Prof. Ron Rivest, Prof. Gene Spafford and Adam Stubblefield for helpful comments.

References

[AK96] Ross Anderson and Markus Kuhn, "Tamper Resistance - a Cautionary Note," *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1-11, November, 1996.

[Garman81] John R. Gamran, "The "bug" heard round the world," *ACM Software Engineering notes*, 6(5):3, October, 1981.

[HW01] Kevin J. Houle, George M. Weaver, "Trends in Denial of Service Attack Technology", October 2001.

[Kohno03] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach, "Analysis of an Electronic Voting Machine", Johns Hopkins Information Security Institute Technical Report TR-2003-19, July 23, 2003.

[MVS01] David Moore, Geoffrey M. Voelker, Stefan Savage, "Inferring Internet Denial-of-Service Activity", Usenix Security 2001.

[MPSSSW03] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, Nicholas Weaver, "Inside the Slammer Worm", IEEE Security & Privacy 2003.

[Pfleeger03] Charles P. Pfleeger and Shari Lawrence Pfleeger, *Security in Computing third edition,* Prentice Hall, 2003.

[SPW02] Stuart Staniford, Vern Paxson, Nicholas Weaver, "How to Own the Internet in Your Spare Time", Usenix Security 2002.

Appendix A

The main body of the report covers some of the most serious security risks in great detail, but we did not attempt to give an exhaustive list. This appendix is intended to supplement that list. In this appendix, we briefly discuss several other security issues that also pose serious risks for SERVE.

Vulnerabilities in Servers Could Breach Election Security

SERVE uses centralized computers (servers) to record and forward votes. If those servers are compromised, every ballot cast through SERVE could be modified or replaced, and the integrity of the entire election would be irreparably damaged. Since the servers are a central single point of failure, it is absolutely vital that they resist attack.

The risk of intrusion into SERVE's centralized computers is, unfortunately, significant. SERVE has deployed a careful and well-designed firewalling architecture designed to prevent many kinds of direct attacks; however, there remain possible vulnerabilities in the software exposed to the outside world that could enable attackers anywhere on the Internet to penetrate SERVE's defenses and gain control of the servers. Some examples of the kinds of vulnerabilities are buffer overruns, format string vulnerabilities, directory traversal bugs, race conditions, cross-site scripting bugs, SQL injection bugs, cryptographic failures, and session authentication weaknesses. It is important to understand that there are no comprehensive methods for avoiding, or testing for, these vulnerabilities and bugs. As a result, even the most widely used, exhaustively tested software in the world usually has many such vulnerabilities. SERVE has deployed several mitigation strategies that we will not enumerate here, but these strategies are not sufficient, in our view, to reduce the risk to acceptable levels.

High-confidence software can be found in safety-critical systems such as nuclear reactor control subsystems, fly-by-wire passenger airplanes, the FAA's air-traffic control system, the Space Shuttle, and certain military applications. Designers of safety-critical systems typically avoid the use of commercial software, because it is widely accepted that standard commercial programming practices pose an unacceptable risk for such applications. Designers of safety-critical software employ known techniques for building highly reliable software. These elaborate and costly techniques have not been used in the development of SERVE. Therefore, we must recognize that the standard commercial practices used in SERVE, and in the commercial-off-the-shelf (COTS) software it builds upon, come with an unavoidable risk of failure.

Existing Processes are Inadequate for Certifying Voting Software

Election law in most states requires that all voting systems—whether electronic or not—be qualified by an authorized federally licensed laboratory known as an Independent Testing Authority (ITA), and then submitted to the state for certification. The ostensible purpose of these procedures it to make sure that the voting system meets the voluntary federal voting system standards promulgated by the FEC (and in the future, by NIST), and that they conform to the state's election laws. It is tempting to place a lot of faith in certification procedures as a means for preventing security failures. We believe such faith is unwarranted. We argue that even a lengthy, conscientious testing and examination program by the most qualified people cannot give us the necessary security guarantees. In fact, in general, no process can, since in most cases the problem of establishing that a program meets any particular security requirement is known to be fundamentally unsolvable [Pfleeger03].

With electronic and Internet voting systems the most important part of the ITA qualification process is review and testing of the software. We argue, however, that this review and testing does not, and cannot, guarantee that the software in voting systems actually does the job it is supposed to do. Ideally one would like the testing to be designed to verify that the software is at least *correct*, i.e. that it captures and counts votes properly under all normal conditions, in any possible election, and with any legal voter behavior. It should also verify that the software is *robust* and *reliable*, i.e. that it works reasonably even in the presence of various bug and failure scenarios, including aberrant behavior by users (such as clerks and voters, etc.); and that the software is *secure,* i.e. that it contains no internal malicious logic (Trojan horse code) and is not vulnerable to any of a vast range of potential external attack scenarios. Unfortunately, however, neither

the ITAs nor anyone else could perform review and testing even remotely comprehensive enough to establish any of these properties because the time and labor involved would be truly astronomical. There are fundamental limits to what testing can accomplish; it is a truism of the software world that *while testing can be used to verify that bugs and security vulnerabilities are present, it can never prove that they are absent*.

What the ITAs actually do to qualify voting system software is considerably less than the ideal. They run mock elections with test votes to verify that the software appears to work as required under normal conditions. This amount of testing is probably sufficient to detect simple program errors and obvious defects, although those will likely already have been found by the developers. (We do not really know because, as we note later, results of this testing in SERVE are secret.) But generally speaking such a testing process cannot be expected to do more than filter out the most obvious bugs, leaving intact more subtle bugs that get triggered less frequently. And it is safe to say that the ITAs do very little testing at all to look for software reliability, security, or malicious code problems, since testing is generally ineffective for those purposes.

Besides testing, the ITAs also examine the source code of the software. Most of the examination is in the form of automated scripts that run over the source and look for simple properties required by the FEC standards, e.g. that there are sufficient comments in the code, that "modules" in the programs are not too long, and that each has a single entrance and exit. These are syntactic and stylistic requirements that are only crude indicators of good engineering practices; they indicate nothing at all about the code's correctness, reliability, or security.

We have been told that the ITA engineers also examine the source code by eye, searching for suspicious-looking code fragments that might indicate the presence of a bug, a security vulnerability, or malicious logic. If so, it may give ITA engineers a better idea of the software engineering skill that went into the program development. But it is unlikely that someone not on the development team will spot many subtle bugs in a large codebase, because he or she can never take the time to fully understand its overall code structure in the necessary detail. And, contrary to many people's intuition, it is unlikely in the extreme that anyone, whether on the development team or not, would detect malicious logic that was deliberately disguised by a clever programmer, no matter how much effort was put into the search.[11] It is much easier to hide a needle in a haystack than to find it.

However, even if the ITAs were to do an excellent job of examining the software used in election systems, we would still have major concerns. Regrettably, the ITAs operate under the same cloak of secrecy as voting machine vendors. Both the tests conducted by the ITAs and the results of those tests are secret. Still, there are some things that we know about how ITAs have evaluated computerized touch screen machines (DREs), and there are other things that we can deduce from some glaring failures in the DREs that were not detected by the ITAs. It is reasonable to assume that the shortcomings of ITAs with respect to DREs will carry over to their certification of Internet voting.

For example, there are fundamental differences between modern software systems and previous voting systems. In particular, today's software-based systems are orders of magnitude more complex than mechanical and paper voting machines, and this complexity poses tremendous challenges for verification. In a mechanical voting machine, there are only so many moving parts and only so many ways that the system can misbehave. In contrast, modern software systems are exceptionally complex, with the electronic equivalent of hundreds of thousands or millions of moving parts; such machines can fail in convoluted and unpredictable ways. The very fact that Microsoft and other software vendors are forced to issue frequent software patches (see below) demonstrates that the complexity of modern software systems makes it impossible to detect all software errors, let alone malicious code, with testing and code reviews. Because current methods are good at exposing only simple failures, these methods are not a reliable way of certifying software.

Even if current practices were effective at checking that the voting system behaves as it should during normal operation, security is by definition about the behavior of the system when it is under attack by a malicious entity, a (hopefully) abnormal situation. Because it is hard to anticipate how attackers might

[11] For an example of software problems that were undetected by ITA testing, see [Kohno03].

behave, it is hard to test for security flaws. Existing practices almost certainly can be improved, but there are limitations on what can be achieved given our current state of knowledge.

It is even harder to detect intentional security flaws that have been purposefully planted by insiders. After careful consideration, we have concluded that if a malicious insider with access to SERVE software wanted to insert a back door in SERVE's software systems, it is likely that the attacker could camouflage the attack well enough to avoid detection by the ITA's during the certification process, especially in the absence of a meaningful code review.

As if these problems with the certification process were not enough, there is a giant loophole in the FEC requirements: commercial-off-the-shelf software does not have to be tested at all by the ITAs. It is simply assumed to be bug-free, Trojan free, and invulnerable to external attack.

And finally, we note that for systems where security is critical, testing should be done in a truly hostile environment. As Anderson and Kuhn point out in [AK96], "although it is necessary to design commercial security systems with much more care, it is not sufficient. They must also be subjected to hostile testing." The SERVE system has undergone ITA testing but has not existed in a truly hostile environment. There are well known examples of code that underwent extreme testing and then failed when deployed in the field. One example that illustrates this is that the first attempt to launch a space shuttle failed due to a software synchronization error [Garman81]. Very few software systems are tested as rigorously as that which runs on the space shuttle, and yet a serious bug was not found until after the failure in the field.

Given these difficulties, it is no surprise that recent experiences with certification of software voting systems have not been encouraging. Regrettably, there are no good solutions in sight. The computer security community has been struggling with these problems for decades, and they are unlikely to be resolved any time soon.

Commercial-Off-The-Shelf Software Poses a Major Risk to Election Security

SERVE relies heavily on commercial-off-the-shelf (COTS) software. Voters will vote from computers running a Microsoft operating system, and SERVE's infrastructure is built on top of commodity operating systems and applications. We are concerned that heavy reliance on COTS software introduces significant risks.

One of the fundamental problems with using COTS software in a voting system is that it is exempt under FEC guidelines from evaluation by the ITA's during federal qualification. It does not have to conform to FEC coding standards and its source code need not be inspected at all. It does not have to be tested at all except in the context of the whole voting system it is part of. Most of the software comprising SERVE—millions of lines, including the cryptographic core (i.e. the software performing the cryptographic operations) —is COTS, and thus exempt from close scrutiny.

Some advocates of the COTS exemption argue that COTS software is everywhere—that it is the most widely used software in the world, and thus the most thoroughly tested and reliable. We believe that this argument is fundamentally mistaken. While we agree that COTS software is widely used, we disagree that widespread usage is reason for confidence in its security. Indeed, broad usage experience has only underscored the fact that the most widely used COTS operating systems and applications are riddled with bugs and plagued by security defects. New security vulnerabilities in COTS software are discovered every day. For instance, over 4,000 new COTS vulnerabilities were reported in 2002 alone. Such statistics make it likely that the COTS software that SERVE relies upon will have many unknown vulnerabilities, some of which could compromise election security if discovered and exploited.

This raises the question: "What do we do if a new security vulnerability is discovered in COTS software upon which SERVE relies?" Currently, when a security hole is discovered, the software vendor issues a security patch, which happens on a fairly frequent basis. Since security patches modify the voting system, SERVE is required to resubmit the system to the ITA and to the states for certification. This procedure normally takes about two weeks from patch issuance to installation.

Today, it is routine to see hackers exploiting new security vulnerabilities only days after they are initially

announced or after the patch is first made available. Thus, on the one hand a two-week delay may be too long to prevent exploitation of vulnerabilities. On the other hand, two weeks may be insufficient to install, *adequately* test, and certify a new patch. Therefore, we believe that the heavy use of COTS software in SERVE, its known security risks, and its exemption from examination during qualification represent significant risks for SERVE.

Appendix B

The ICANN experience

Internet voting has been occurring for several years, most typically with private elections, such as stock holder elections or the election of officers of private organizations.

Internet elections might possibly be acceptable if the stakes are not high. For example, if the issues on which stockholders are voting are not controversial, then there is little incentive to subvert a stockholder election. By contrast, there is a lot of incentive to subvert national presidential and congressional elections.

In 2000 the Internet Corporation for Assigned Names and Numbers (ICANN)[12] held an election over the Internet. Anyone age 16 or older with an Internet address was eligible to vote. The world was divided into five regions, and candidates were nominated from each region by a nominating committee. There was also a Member Nomination process that involved the "endorsement" of a potential candidate via the Internet.

Problems with accessing the official website occurred in every phase of the ICANN election, including voter registration, endorsement, and voting. ICANN significantly underestimated the computing resources that would be required, especially since people tended to wait until near the various deadlines to attempt to access the ICANN website. Also, some of the processes that ICANN had put in place to try to minimize fraud, such as requiring "activation" of membership after a member had registered, were confusing to voters. Many voters were unable to vote because they did not activate their membership by the deadline.

Even though individual passwords were mailed to physical addresses, some people claim to have voted multiple times. There were also numerous reports of people being disenfranchised. Disenfranchisement occurred because the ICANN website was overwhelmed by the demand, because some voters never received their passwords or subsequently lost them, and because some voters did not realize that registration was not sufficient for voting. In spite of difficulties with registration, about 158,000 people registered. While 76,183 of the registered voters activated their membership, the number who managed to vote in the election was only 34,035.[13] These numbers suggest large-scale disenfranchisement of voters at every stage of the process, with fewer than 1/4 of the initially registered voters actually voting. According to the Markle Foundation report, "the technical weaknesses in the registration system made it virtually impossible to assess the integrity of the voters' list, the security of the PINs, and secrecy of vote."

There were also rumors of country or corporate competition to pack the voter list during the later phases of the registration period. Technical problems experienced during the first day of voting, which took place over a ten day period, created, according to the Markle report, "a credibility problem…with voters and interested parties watching the process."

---

[12] One of the authors of this paper, Barbara Simons, was nominated by the endorsement process; she was the runner-up in North America, losing to Karl Auerbach.

[13] "Report on the Global, On-Line, Direct Elections for Five Seats Representing At-Large Members on the Board of Directors, a report by the Markle Foundation, http://www.markle.org/News/Icann2_Report.Pdf

Appendix C

An Alternative to SERVE

Although we are concerned that the SERVE architecture has too many serious vulnerabilities to be used in public elections, we agree nonetheless that there is a need to lower barriers to voting for overseas Americans and the military. Here we offer an alternative architecture that we believe has most of the benefits that SERVE offers, with far less vulnerability. We are not proposing that this scheme be adopted, but we do suggest that it is a much better starting point for a voting system for that population.

The troubles with SERVE derive from three fundamental design choices: It uses the Internet heavily, with all of the vulnerabilities that implies (e.g., denial of service, spoofing, and man-in-the-middle attacks). It relies on voters using private, unsecured PCs with proprietary, commercial software configured to accept mobile code, with all of the vulnerabilities that implies (e.g., virus attacks, various kinds of privacy violations). And SERVE itself is proprietary software, with all of the vulnerabilities that implies (e.g., security holes, bugs, insider fraud).

We suggest building a system that avoids the dangers of these architectural features. Here is how it might be done:

a) The system would be based on kiosks, to be located at consulates and military bases at major sites around the world. Voters would come to the kiosk to vote, rather than vote from their own PCs.

b) The software on the kiosk would be booted from a clean copy, maintained and configured by trained elections officials so that the software environment on the voting machine is known and controlled.

c) The kiosk is never connected to the Internet; hence, no Internet-related attacks are possible. It receives software and databases via disks (such as DVD Write Once Read Many (WORM)) delivered by certified mail in advance of the election. It does not transmit ballots back to the counties by Internet; rather, it prints them, and they are mailed back to the counties, just like any other absentee ballot.

d) The kiosk has three databases that allow it to perform the voting functions. All three together could fit on one DVD-WORM disk.

   i) An identification database to authenticate all voters who have registered to vote via the SERVE system. This would have at most 6 million records, since that is the size of the eligible population.

   ii) A voter registration database to indicate which ballot image (style) each voter is supposed to receive. This data is compiled from information contributed by any or all of the 3000+ county jurisdictions in the U.S. that wish to participate. There are approximately 200 million registered voters in the U.S., but only at most the 6 million records corresponding to those eligible to vote via SERVE would be needed.

   iii) A ballot image database, containing a PDF or XML representation of each ballot style used in every county in the U.S. that is participating in SERVE. There are on the order of 100,000 such ballot styles. These also would be provided by the LEO in the counties participating in SERVE.

e) The voter at the kiosk identifies himself or herself using military ID or other authentication information provided by SERVE. The kiosk verifies that the voter is registered to vote, and looks up which ballot style to issue. Then the voter indicates his or her choices on a touch screen, and prints out a completed ballot. The voter examines the printed ballot to make sure it correctly indicates his or her choices. (If not, the voter approaches the election official for help in voiding the ballot and re-voting.) This last step provides for a *voter-verified paper ballot*, and assures that no potential bugs or Trojan logic anywhere in the system's code can incorrectly record the voter's intent. Finally, the voter

deposits the completed ballot in an envelope addressed to his home county for mailing back, again like a traditional absentee ballot.

The system we have described is essentially a remote ballot printer located near most Americans throughout the world. Its design would need to be fleshed out with more detail and additional security procedures. These procedures would be charged with preventing multiple voting, managing encryption keys for the databases, and determining what kinds of electronic record of the voting transactions the machines keep and what they do with these records.

We stress that this architecture is not a complete system; it is only a starting point. The details would need to be worked out, and that process could always introduce unforeseen issues. For this reason, we believe it is crucial that any such system undergo hostile review, and that all relevant communities be involved in the design and evaluation of the system. However, with appropriate care, we believe such a system could deliver significant benefits. Such a system would reduce the voting transaction from three trips through the mail down to one, and with proper development we expect it could be made far less vulnerable than SERVE to any kind of remote or programmed attack.

Appendix D

In this appendix, we elaborate on some of the issues that are often misunderstood about software and the difficulty of finding hidden code or flaws in programs. We also discuss why the Internet and the current personal computers are not appropriate platforms for voting applications.

Determining that software is free of bugs and security vulnerabilities is generally impossible

In mathematics there is a long history of profound impossibility results, i.e. theorems stating that certain problems are fundamentally impossible to solve regardless of how much effort or intelligence is applied to them. For example, it is impossible to trisect an angle (divide it into three equal parts) by classical means of ruler and straight edge construction; it is impossible to solve a 5th-degree (x^5) polynomial equation using only addition, subtraction, multiplication, division and extraction of roots; it is impossible to construct a consistent and complete axiomatization (set of rules) for arithmetic.

Computability theory has a many such results, referred to as unsolvable or incomputable problems. They are typically of the form "there exists no computer program that can do X". The first and most famous such result, due to Alan Turing, is known as the Halting Problem: there exists no program H that can determine (in a finite number of steps) whether or not an arbitrary other program P ever halts. It does not matter what programming language is used, or what computer is used, or how fast it is, or how much memory is available, or how long the program is, etc.; there simply is no program H that can solve the Halting Problem.

If a problem is unsolvable in this sense, then humans cannot solve it either. This may seem contrary to the common observation that there are things humans can do easily that no computer program can do, e.g. understand the English language; but these are examples of things no program yet written can do, rather than examples of things no program at all can do. In general, if humans can perform an information processing task, then they do so by some method in the brain. We may not know the method; but if we did, we could write a (complex) program that emulates that same method and performs the task just as well. There does exist a program that understands English, even if no one knows at the moment how to write it, so understanding English is not an unsolvable problem.

Unfortunately many of the important questions one might ask about the behavior of software--its correctness, reliability, or security--turn out to be unsolvable. Here is a partial list of such problems that are relevant to program correctness (absence of bugs), program security, and the privacy required by election software. They are stated informally because the mathematical apparatus to state them precisely is beyond the scope of this report; but each could be suitably formalized and proved mathematically.

It is impossible to determine by any finite means, for an arbitrary program, whether or not it:

- halts (a compressed phrasing of the Halting Problem);
- has a particular type of bug (e.g. array bounds error, fencepost error, dangling pointer);
- is correct (i.e. meets its specifications);
- has hidden functionality (i.e. does additional things not mentioned in its specifications, especially malicious actions);
- preserves privacy (does not store or transmit information that it should not);
- is reliable (i.e. does something reasonable in spite of various kinds of failures, e.g. memory, communication, or software failures);
- is secure (i.e. performs its job in spite of various kinds of attacks).

Reading this list one should get the impression that just about anything you really want to know about the behavior of important software, including election software, is impossible to determine for certain. And indeed that is the case. If there were a surefire way to detect bugs in programs, then commercial software would be bug-free. And if there were a foolproof way to find or avoid security vulnerabilities, then

operating systems companies would use those methods, and would not have to issue security patches on a regular basis.

If all of these important software problems are unsolvable, then how does working, secure software get built at all? This is the subject of software engineering. The general answer is that, with enough careful mathematical and algorithmic analysis ahead of time, with enough methodological care and discipline on the part of programmers, with ferocious drive for simplicity of design, with enough systematic and randomized testing of the software, with formal proofs that key parts of the program have key properties, and with enough openness so that the software can be scrutinized by many experts, then the prevalence of defects such as bugs, or security vulnerabilities, can often (but not always) be made tolerably low. But there is no method for writing totally bug free software, or totally secure software; and if by a miracle such software were to be created, there is no general way of proving it to be bug-free or secure. This is a mathematical fact of life. Anyone who claims that some nontrivial program is bug-free or secure simply because it has been widely used or thoroughly tested is misinformed. No amount of use or testing is sufficient to prove that.

Though there has been substantial research on new methods for software engineering—for instance, proof carrying code and static program analysis, among others, these technologies are currently immature, and they are a long way from being applicable to the complex type of software needed for elections.

Malicious code

Software is generally designed to serve a clear, documented purpose, and people rely on it doing what its specifications and documentation say. But software can be written to do other things in addition to, or instead of, what it is supposed to do, i.e. it may have secret or hidden functionality that is not documented. Sometimes this is legitimate, but sometimes the hidden functionality is malicious, i.e. it does something that the programmer or vendor wants, but the user does not want, and does it without the knowledge of the user. Such malicious code might spy on the user by surreptitiously sending private data to an Internet site; it might throw ads in the face of the user; it might disable security protections on the computer to allow later break-ins by unauthorized people; it might install other unwanted programs; it might do random violence by deleting data or files; or it might do any of a thousand other malicious things.

The terms *Trojan horse*, *virus*, and *worm* all refer to types of malicious code, differing only in the means by which they get transported to the computer and get executed. Most people are aware of email viruses, which is malicious code in the form of an email attachment, but there are many other infection routes, including, for example, malicious scripts (ActiveX controls, JavaScript programs, or Java applets) that can enter your computer as an invisible side-effect of visiting a web page. Malicious code is one of the most serious security threats in any application, because it is so easy to install, and so difficult to detect.

In the context of the SERVE voting system, malicious code is a threat in two separate ways. First, it is possible that an insider (one of the developers who builds SERVE) might insert malicious code into the SERVE software itself, perhaps to spy on votes or to selectively throw some away. (We don't suggest this is at all likely; but the fact is that it cannot be excluded.) The other threat is malicious code infecting voters' computers, in such a way that it spies on the voting process, or prevents voting, or even changes votes without detection.

It is important to understand, as discussed above, that there is no foolproof test for whether or not malicious code is installed. Virus checkers, for example, can detect the presence of viruses that have been seen before, studied by experts, and for which a signature has been extracted; but it cannot reliably detect new viruses. Even experts with access to the source code of a program may not be able to tell if there is malicious code in it, since it is relatively easy to disguise malicious code so that it is extraordinarily difficult to find.

The most powerful defense against malicious logic is for the security of the election not to depend on detecting it at all, but to structure the voting system so that we can guarantee that it works correctly even if malicious logic is present. Voter verified audit trails are among the simplest and strongest features that can provide such guarantees (though there may be others). But in our opinion some form of protection against malicious logic is imperative for any software-based voting system.

Security weaknesses of the Internet

The Internet is revolutionizing communication, commerce, and entertainment, but in ways that were never envisioned by its designers. The basic data transmission, naming, and routing protocols still in use on the Internet today, called collectively TCP/IP, were designed in the late 1970's and early 1980's, a time when all network users formed a single community, when the network engineers all knew each other personally, and there was universal trust among the users. There was little need for, security because the community of users was small and everyone was cooperative. The key goals of the Internet's original protocols were scalability, ease of use, communication performance, and reliability; security was not a priority, and the issues were not well understood at the time anyway.

But today, after a million-fold growth in size, the Internet is a voluntary worldwide federation of networks of great diversity, with no central authority and no common culture or goals. It links hundreds of millions of people and organizations that are mostly strangers to one another, but sometimes competitors or even enemies. In such an environment, security is of profound importance, and many security protocols have been layered on top of the basic ones with the intent to secure certain kinds of applications, e.g. email or financial transactions, against certain specific threats.

Yet, for all of the importance of security today, the Internet has no general security architecture; in fact, it is well known to be full of very general vulnerabilities. It relies on the voluntary cooperation of thousands of corporations around the world to keep its naming and routing infrastructure consistent, and it uses open, forgeable source addresses on every packet it transmits. As a result, it is not possible to guarantee exactly where an outgoing data packet will be routed, or where an incoming packet came from. These limitations are the ultimate reason why spoofing and man-in-the-middle attacks are so easy to perpetrate and difficult to defend against in any Internet application, not just elections. Likewise the routers and servers distributed all over the world that form the Internet's higher-level infrastructure are themselves just computers with their own array of security vulnerabilities, known and unknown; they are often easily tampered with, by either insiders or outsiders.

It is extremely hard, to build a secure, all-electronic Internet-based application, e.g. a banking system, a legal system, or an election system, on top of such a fundamentally weak security foundation. In our opinion, no one should attempt it without voter-verified audit trails and out of band (i.e. non-Internet) channels of communication between the LEO and the voter, or else an as-yet unknown security breakthrough.

Security weaknesses of the PC

The PC platform that is used as a voting machine under SERVE is another dangerously weak element of the security architecture. A PC running Windows software is very easily compromised, and never more so than when it is connected to the Internet.

The PC was originally designed as a personally owned and maintained, single-user system (which is why it was called a "personal" computer). This fact seemed to minimize the need for sophisticated security since the owner/user could always trust himself/herself. The Internet was not originally a security consideration because the PC family of architectures was designed in the early 1980's, long before there was any thought that PCs would connect directly to the Internet. In any case, it was generally assumed that whatever light security might be useful, such as password protection, could be satisfactorily implemented entirely in software.

As a result, PC hardware has evolved to this day with essentially no concern for security at all. Most PC motherboards, for example, contain no tamperproof, cryptographic keys that can be used as the basis for identification, and no source of true random numbers for cryptographic key generation. Furthermore, the key human interface devices for PCs (keyboard, pointer, screen, or speakers) are not designed to perform cryptographic operations. This has the effect of forcing critical data to be manipulated in the clear (unencrypted) on the main part of the PC, where it is vulnerable to malicious software.

The Microsoft software suite for the PC was also originally designed without security in mind. The primary design goals of the Windows and Internet Explorer have been maximal functionality and ease of use, not security. Only recently has Microsoft made security an important priority, but their operating

systems are still well known to be thoroughly riddled with security vulnerabilities. Indeed, for years hardly a week went by when Microsoft did not issue one or more security patches. Indeed, to help manage the problem Microsoft has started batching the patches so that groups of them are released monthly.

Windows is especially vulnerable to malicious software attacks. There are so many forms of software, and so many vectors of transmission, that it is impossible to control them all. Users are constantly encouraged to download and install software, sometimes without knowing it, including OS and browser updates and patches, device drivers, plug-ins for browsers and other applications, scripts associated with web pages and office documents, shareware programs, and of course, full applications. Any of these types of programs can contain malicious logic that might completely undermine the security of the PC, and of any votes cast on it, without the user/voter ever finding out.

These security limitations of PC hardware and software are widely acknowledged. There is an industry-wide consortium known as the Trusted Computing Platform Alliance (http://www.trustedcomputing.org) that has specified hardware additions to the PC architecture that are intended to remedy some of its security deficiencies. So far there are no hardware implementations of the TCPA spec and no operating system support for it. It is possible that within a few years TCPA-equipped PCs will be begin to be sold, and a few years after that they will largely replace all of the current generation of PCs. But there is as yet no public specification of what security features and tools Microsoft will implement using the TCPA hardware, or whether they will suffice to permit safe Internet voting from a PC.

Appendix E

Author Biographies

David Jefferson is a computer scientist at Lawrence Livermore National Laboratory where he does research in scalable parallel computation (supercomputing). He has also worked for many years in the field of election technology and security, serving as chair of the Technical Committee of the California Secretary of State's Task Force on Internet Voting in 1999-2000; as a member of the executive committee of the NSF panel on Internet voting in 2000-2001; and a member of the California Secretary of State's Task Force on Touchscreen Voting in 2003. He has been active nationally as a speaker and advisor on computerized voting system issues, and is a current member and past chair of the board of directors of the California Voter Foundation. Before joining LLNL, he was a computer scientist at DEC/Compaq Labs in Palo Alto, and before that had been Associate Prof. of Computer Science at UCLA, where he was best known as the co-inventor of the Time Warp method of parallel discrete event simulation.

Aviel D. Rubin is an Associate Professor of Computer Science and Technical Director of the Information Security Institute at Johns Hopkins University. Prior to joining Johns Hopkins Rubin was a research scientist at AT&T Labs. Rubin is author of several books including Firewalls and Internet Security, second edition (with Bill Cheswick and Steve Bellovin, Addison Wesley, 2003), White-Hat Security Arsenal (Addison Wesley, 2001), and Web Security Sourcebook (with Dan Geer and Marcus Ranum, John Wiley & Sons, 1997). He is Associate Editor of ACM Transactions on Internet Technology, Associate Editor of IEEE Security & Privacy, and an Advisory Board member of Springer's Information Security and Cryptography Book Series. Rubin serves on the board of directors of the USENIX Association and on the DARPA Information Science and Technology Study Group.

Barbara Simons is a technology policy consultant. She earned her Ph.D. from U.C. Berkeley, and was a computer science researcher at IBM Research, where she worked on compiler optimization, algorithm analysis, and scheduling theory. A former President of the Association for Computing Machinery (ACM), Simons co-chairs the ACM's US Public Policy Committee (USACM). She served on the NSF panel on Internet Voting, the President's Export Council's Subcommittee on Encryption, and the President's Council on the Year 2000 Conversion. She is on several Boards of Directors, including the U.C. Berkeley Engineering Fund and the Electronic Privacy Information Center, as well as the Advisory Board of the Oxford Internet Institute and the Public Interest Registry's .ORG Advisory Council. She has testified before both the U.S. and the California legislatures. She is a Fellow of ACM and the American Association for the Advancement of Science. She received the Alumnus of the Year Award from the Berkeley Computer Science Department, the Norbert Wiener Award from CPSR, the Outstanding Contribution Award from ACM, and the Pioneer Award from EFF.

David Wagner is an Assistant Professor in the Computer Science Division at the University of California at Berkeley. He has extensive experience in computer security and cryptography, and over 50 technical publications. He and his Berkeley colleagues are known for discovering a wide variety of security vulnerabilities in various cell phone standards, 802.11 wireless networks, and other widely deployed systems. In addition, he was a co-designer of one of the Advanced Encryption Standard finalists, and he remains active in the areas of systems security, cryptography, and privacy. Wagner is an Alfred P. Sloan Research Fellow and a CRA Digital Government Fellow.