Software Review and Security Analysis of Scytl Remote Voting Software

Michael Clarkson Brian Hay Meador Inge abhi shelat David Wagner
Alec Yasinsac

September 19, 2008

Contents

Chapter 1

Executive Summary

On May 20, 2008, the Florida Department of State (FLDoS) commissioned an independent expert review of the Pnyx.core ODBP 1.0 remote voting software, which was developed by Scytl. The review team was led by Florida State University's (FSU) Security and Assurance in Information Technology (SAIT) Laboratory. The review extended over a period of approximately three months and was performed by six independent technical experts. The review process included significant interaction between the vendor and the review team; the reviewers felt that this was positive and could serve as a helpful model for future reviews. This report is the culmination of that review.

Scytl's voting software is intended for use in the Okaloosa Distance Balloting Pilot (ODBP), which will test remote electronic voting for about one thousand overseas voters whose permanent residence is in Okaloosa County, Florida. This pilot replaces other absentee voting mechanisms for participating overseas voters. ODBP voters will enter their votes electronically, those votes will be transmitted over the Internet, and the votes will be tabulated electronically.

Electronic voting is highly controversial and known to present significant security risks and technical challenges. The computing systems on which electronic voting is based inherently increase the risk of large scale election fraud. Additionally, the Internet is a high-risk computing environment where malware is pervasive, hacking is a growing enterprise, and threats are difficult (or impossible) to quantify. The Pnyx.core ODBP 1.0 voting software is designed to mitigate these risks by employing (i) cryptographic techniques that are not present in traditional paper and mechanical voting systems, or even some modern electronic voting systems, and (ii) voter-verified paper records (called "Voter Choice Records"), as legally required by the state of Florida. This review identifies some strengths and weaknesses of this approach to electronic voting.

Although the review team had access to the Pnyx.core ODBP 1.0 voting software, the software is only one component of an entire voting system that includes people, procedures, and environments. We did not have access to many details of this system context—in part because such details had not yet been determined and documented during the review period, and in part because we were asked to focus our review on software. Such details include the procedures for election officials, the configuration of hardware and software components, and the chain of custody and audit procedures for the Voter Choice Records. We made our best effort to analyze the system given the available information, but we emphasize that our findings may require reinterpretation as such details become available.

In summary, our findings indicate:

- The system defends against outsider threats (such as Internet-connected hackers) as well as can reasonably be expected, given current technology. Cryptographic techniques and voter-verified records provide confidence that the system can be defended against outsider attacks. (See §4.2.)

- The system is vulnerable to attack by trusted insiders (such as election officials behaving maliciously). Defending against such attacks can be challenging in any voting system. In Scytl's system, Voter Choice Records are pivotal to this defense. Manual counts of the Voter Choice Records, as well as procedural controls on insider access to the system before and during an election, are the only way we have identified to secure the system against insider threats. (See §4.2.2.)

- The system relies on several trusted components that are critical to proper system functioning and security. These components must be strongly defended; otherwise, the system becomes more vulnerable to outsider and insider threats. We found that this weakness is not well documented. (See §4.2.5.)

- The system employs standard cryptographic algorithms in protecting ballot secrecy and integrity. However, the system does not employ advanced cryptographic techniques from the research literature (e.g., [18, 12, 3, 7, 22, 1, 8]) for minimizing trust assumptions about the hardware, software, people, and procedures that are part of the voting system.

Our findings identified vulnerabilities that, in the worst case, could result in (i) voters being unable to cast votes, (ii) an election result that does not accurately reflect the will of the voters, or (iii) disclosure of confidential information, such as the votes cast by a voter. The extent to which these vulnerabilities could actually be exploited in the ODBP is beyond the scope of this report given our lack of system context. Secure handling and audit of the Voter Choice Records may defend against some or all of these vulnerabilities, but these procedures were not available for review.

We identify three findings of particular significance:

- The use of supervised polling stations provides significantly better protection against voter coercion or vote-selling than is present in some other absentee voting systems, such as voting by mail. (See §4.2.1.)

- Two copies of each vote are stored: one electronically, and another on paper as a Voter Choice Record. This provides redundancy that is not present in existing vote-by-mail systems. If the electronic votes are well-protected, then they can enable audit of the paper records in ways that are not currently possible. (See §4.2.4.1.)

- After casting their ballot, each voter is given a receipt that is intended to give voters confidence that their votes were "Counted as Cast". These receipts do not achieve their stated goal of allowing voters to "independently verify that their ballots have been correctly accounted for." These receipts might indicate that a vote was received and decrypted by the county (a property not typically provided by current postal voting systems), but they do not provide assurance that the voter's vote was correctly recorded. (See §4.2.3.1.)

The findings in this report should not be assumed to apply beyond the context of the ODBP, including the software and hardware versions specified, the scale of the election, and procedures and processes as they were known to the review team at the time of the review. We emphasize that security reviews, especially of computing systems, are inherently limited in scope, as should be the interpretation of a review's findings.

It is important to note that this review is only one element of the review and testing process that will lead to the FLDoS certification decision. FLDoS conducts extensive in-house functional testing and software review for all voting system certification efforts. Thus, our review was not chartered, and is not intended, to provide comprehensive analysis; rather, it focuses on security issues from a software perspective as part of the overall certification process.

Chapter 2

Team Membership and Organization

2.1 Team Members

The team members come from a variety of schools and many have significant voting system software experience. Several members have been involved in state sponsored reviews for voting systems that were under consideration for state certification. Members have also created, or participated in creation, of novel voting systems that employ cryptographic techniques to accomplish strong electoral assurance. All members are familiar with cryptography, network security, secure coding, and software engineering to some extent and each member has strong skills in one or more of those areas.

2.2 Senior Investigators

1. David Wagner, Associate Professor of Computer Science, University of California, Berkeley (Lead Technical Investigator)
2. Brian Hay, Director, ASSERT Laboratory, University of Alaska, Fairbanks
3. abhi shelat, Assistant Professor of Computer Science, University of Virginia
4. Alec Yasinsac, Professor and Dean, School of Computer and Information Sciences, University of South Alabama (Project Administrator)

2.3 Investigators

1. Michael Clarkson, Doctoral Student, Department of Computer Science, Cornell University
2. Meador Inge, Graduate Student, School of Computer and Information Sciences, University of South Alabama

2.4 Withdrawn Members

Ryan Gardner and John Kerski graciously volunteered their time for this project and both participated in the planning and early investigative efforts, but were unable to continue due to scheduling conflicts and competing time demands.

2.5 Organization

Unlike previous SAIT Laboratory reviews for the Florida Department of State, this review was unfunded. Thus, most of the work occurred at night and on weekends, with the investigators volunteering their personal time in support of this effort. Correspondingly, the review was conducted in a distributed fashion and investigators did not physically

gather at any time during the review. Collaboration channels were established to facilitate team interaction and periodic conference calls promoted synergy among the members.

2.6 Review Process

The review process included several components:

- Documentation review
- System architecture review
- Hands-on source code review
- Static analysis using automated tools
- Direct vendor interaction and Q&A

Because of the large quantity of source code and the time limits of this review, our inspection of the source code was severely limited. Instead of reviewing large amounts of code by hand, team members spent the majority of their energy on documentation review, architectural risk analysis (where we examined the design decisions made by Scytl to ascertain their potential implications and identify systematic risks), and analysis of the cryptographic protocols.

The source code was delivered to the team in three phases. The Pnyx.core ODBP 1.0 software consists of a combination of the Pnyx.core libraries plus custom code written specifically for the Okaloosa Distance Balloting Pilot. The Pnyx.core libraries and documentation were delivered in mid May, followed about two weeks thereafter by the build environment. Finally, the custom code that was written specifically for ODBP-unique requirements was provided to the team in late June. The team members did not have the complete voting system source code until the third phase of delivery. The following is a more detailed project timeline of 2008.

- May 6: FLDoE signed the SoW and notified the Administrative Lead
- May 12: Administrative lead picked up first code CD from FLDoE. This CD included source code and documentation from Scytl's Pynx.core application
- May 13: Project Administrator began receiving security documents from team members
- May 19: Limited distribution of initial CD
- May 26: Administrative Lead received VM environment
- May 30: First conference call with Scytl. Okaloosa Associate Supervisor of Elections Paul Lux participated in this call
- June 3: Michael Clarkson joined the review team
- June 4: Joint Review Team/Scytl Bugzilla interaction database established
- June 5: Second conference call with Scytl
- June 17: Submitted review plan to FLDoE
- June 18: Received replacement disk for VM environment
- June 18: abhi shelat joined the review team
- June 19: Received TDP and final software distribution from FLDoE containing ODBP-specific code
- June 19: Meador Inge joined the review team
- June 19: Team conference call
- June 24: John Kerski withdrew from the review team

- July 14: Ryan Gardner withdrew from the review team
- July 23: Team conference call
- July 29: Team conference call

Scytl's submission of a full build environment for the source code was helpful to us in our review. Scytl provided VMWare virtual machine images containing a full build environment, scripts to drive the build process, and step-by-step documentation describing how to initiate the build process. This saved us significant time. Also, it made it possible for us to apply static analysis tools to the software.

We leveraged two different static analysis tools during our review. First, Fortify Software donated use of their source code scanning product, Fortify SCA, for our use during this review. We used Fortify SCA to scan for potential vulnerabilities in the code. Second, we used results from a scan of the source code using the Klockwork static analysis tool. This scan was performed by the Florida Division of Elections (FLDoE), and we thank the FLDoE for sharing with us the output of the Klockwork scan. Both tools produced detailed reports with many specific issues raised in the code. Unfortunately, due to time restrictions, we did not have time to analyze the reports in any depth. Instead, we were able to perform only a superficial examination of these reports.

Unlike previous reviews, the team participated with the vendor in an online question-and-answer exchange that proved invaluable to the study. The review participants felt that Scytl participated in the review in good faith, that their responses were prompt and informative, and that their participation significantly improved the effectiveness of the review process. We thank Scytl for the attentive, professional way that they engaged in this novel review process.

During the review, we focused on the software and documentation that was provided to us as part of the material submitted for certification. When the certification package did not include a particular kind of document that we expected to have been provided, we reported that omission. We emphasize that it is possible that those kinds of documents may actually exist even though they were not included in the certification package (e.g., because others did not anticipate that those particular documents might be relevant to this review). Similarly, where we attempted to conduct tests but were unable to do so, we report that fact. We report these instances as a record of our efforts: our intent is not to find fault but to document our analysis of the material submitted for certification.

In accordance with the Statement of Work, the vendor provided Chapter 6 as their perspective on our findings.

2.7 Project Scope

The project scope is carefully delineated in the Statement of Work. We repeat the pivotal paragraph here for the reader's convenience:

> "The project's purpose is to provide sufficient technical information in order for the Florida Department of State to render an informed decision on system certification. The project results, however, shall not include any recommendation for or against provisional certification of the target software."

Additionally, the statement of work narrows the project scope to three primary tasks related to software quality.

> "The project's aim is to assess the security posture of the target software by analyzing:
>
> 1. The overall quality of the code base,
> 2. The security architecture of the system, and
> 3. The security properties of the software."

Where we found vulnerabilities or potential risks, we attempt to list attack prerequisites and mitigations based on our analysis.

The project scope is also guided by questions posed by FLDoE in the Statement of Work. We list those questions in our findings section, where we provide our best effort to answer those questions based on our review.

The Statement of Work also requires project members to take care "not to include any sensitive information or otherwise confidential and exempt information" in the final report, and further requires:

> "Confidential and exempt information under the Florida Public Records law shall be identified in writing in the Final Report to DOS for purposes of redaction or exclusion in the version of the report released for the public."

This report represents our Final Report. In accordance with the Statement of Work, we have taken care to avoid including sensitive or otherwise confidential and exempt information in the report.

In the process of writing this report, we realized that we were not able to determine whether certain information is confidential and exempt under Florida Public Records law. The information in question provides details about the cryptographic protocol used by the Pnyx.core ODBP 1.0 voting system. To ensure that the body of our report does not include confidential and exempt information, we placed this information in a separate appendix to this report, labelled Appendix A. Our hope is that segregating the document in this way will make it easier for the FLDoE to make a determination about the status of this information.

2.8 Review Limitations

This review's purpose was to evaluate the target system's technical properties. Our analysis was not comprehensive: we focused primarily on the system's software and other technology, rather than election procedures and practices.

Operational system. One of our goals was to install the Pnyx.core ODBP 1.0 software on our own and run a mock election on the software, but in the end we did not succeed in this goal. The review team was provided with source code and virtual machines on which an appropriate build environment was preconfigured. While it was possible to easily build the majority of the voting system software using these virtual machines, the review team was unable to deploy a fully configured running election system from these components due to several factors. Documentation which may have made much of this work possible was not provided to, or explicitly requested by, the review team during the audit process, and in at least one case (the Voting Kiosk Live CD) specialized hardware would have been required in order to execute components of the voting system. As a result, the reviewers were unable to perform functional testing, load testing, fuzz testing, penetration testing, or other dynamic testing of the software in operation. We were also unable to confirm the attacks hypothesized in our findings against an operational version of the system.

Election procedures. Many aspects of election procedures were still under development at the time of the review and thus were not available to us. As a result, our analysis may fail to reflect how the system will be used in a real election, including procedural mitigations or other compensating controls. This may lead to an unduly negative—or positive—view of the system.

Cryptographic implementation. In our analysis, we focused primarily on the design of the system and the choice of cryptographic algorithms. We did not verify that these algorithms were implemented correctly, and it was generally beyond our ability to comprehensively check for implementation flaws. Also, we did not evaluate the possibility of configuration errors or other usability flaws. Problems in these areas could potentially compromise security or invalidate our conclusions.

Software testing. It is well-known that testing is an inherently incomplete process and that no testing process can guarantee absence of faults. Determining whether a complex software system will behave as required is notoriously difficult, expensive, and error-prone. Thus, we make no guarantees regarding comprehensive fault detection.

Because of our primary focus on the overall architecture of the system, rather than the code, this review was especially limited in its ability to detect bugs. The methods we used are especially unlikely to discover malicious logic intentionally introduced into the software, such as Trojan horses hidden in the source code.

2.9 Limited Interpretation of Results

We emphasize that our results do not extend beyond the scope of our investigation. We do not contend that this system is correct or secure beyond the specific results given here. Unless otherwise noted, the results of this investigation should be assumed to be relevant only to this particular election system, including the software and hardware versions

specified, the scale of the election, and procedures and processes as they were known to the review team at the time of this review. We assumed that the materials provided to us match what would be executed on election day; we have no way of checking this, and it was out of scope in any case.

We defer interpretation and application of our results to FLDoS. We made no attempt to determine whether the system complies with federal or state voting standards, regulations, or laws.

Chapter 3

System Overview

Scytl's Pnyx.core ODBP 1.0 remote voting software uses cryptographic mechanisms with the goal of providing security guarantees, such as the confidentiality of ballots and the accuracy of election results. This is a paradigm shift from election systems currently used in state elections.

The source code associated with the Pnyx.core ODBP 1.0 software is the proprietary, business-confidential property of Scytl. It is not generally available to the public absent nondisclosure agreements. The Scytl system has never been subject to federal certification testing or to examination by federally qualified testing labs to evaluate whether it complies with US voting standards.

The Pnyx.core ODBP 1.0 voting software is intended for use in the Okaloosa Distance Balloting Pilot (ODBP), which will test remote electronic voting for about one thousand overseas voters whose permanent residence is in Okaloosa County, Florida. This pilot replaces other absentee voting mechanisms for participating overseas voters. ODBP voters will enter their votes electronically, those votes will be transmitted over the Internet, and the votes will be tabulated electronically.

3.1 Architecture

The Pnyx.core ODBP 1.0 remote voting system involves the following participants:

- **Voters.**
- **Poll workers.** There will be at least two poll workers in each polling place, and they are responsible for running the polling place.[1] This includes the issuing point laptop, which runs the issuing point client software that is responsible for issuing voter credentials.
- **Canvassing board.** A set of people or organizations that are entrusted with overseeing the election, including validating and tabulating absentee ballots. They are given shares of the private key that allows the cast votes to be decrypted. In the ODBP, this set will contain three members.
- **County election officials.** These include individuals who are responsible for administering and operating the Mixing Service (see below).
- **Scytl employees.** It is envisioned that in the November 2008 election, some Scytl employees will be responsible for administering the Voting Proxy and Voting Service software (see below).

The system architecture includes several important components:

- **Voting kiosk.** A device,[2] located at the polling place, through which the voter expresses her preferences. In this system, the voting kiosk is an off-the-shelf laptop that is running Voting Client software. The laptop is booted using a bootable CD supplied by Scytl, called a Live CD, which downloads the Voting Client application from the Voting Proxy at the start of the election. The Live CD is based on an open source off-the-shelf Linux distribution. The polling place will have an Internet connection, and the voting kiosks will be Internet-connected.

[1] ODBP documentation [9] uses the term *kiosk worker* instead of *poll worker*, and *voting kiosk* instead of *polling place*.

[2] ODBP documentation [9] uses the term *voting laptop* to refer to this device.

- **Voting proxy.** An Internet-connected server, consisting of a Java web service that is executed on an Apache Tomcat web server, which is an open source, off-the-shelf web server supporting J2EE web services. The Voting Proxy interacts with the voting client during the voting protocol. During the November 2008 election, it is expected that the Voting Proxy software will execute on a machine managed by Scytl employees and Okaloosa County officials and physically located in space leased by Scytl at a secure third-party data center. The Voting Proxy is primarily responsible for relaying communications between the Voting Client and the Voting Service.

- **Voting service.** A C++ application that runs on a machine located in the same data center as the Voting Proxy. The Voting Service is not Internet-connected, but it is connected by a local-area network to the Voting Proxy. The Voting Service is responsible for storing encrypted votes. The Voting Service is connected to an Oracle database running on a machine located in the same data center; the database is used to store encrypted votes. The documentation specifies use of Oracle Database (RAC) version 10.1.0.4 or higher.

- **Mixing Service.** An application that receives the set of cast votes, shuffles the votes, decrypts them (with input from the Canvassing Board), and outputs the tallies. This software component runs on a dedicated computer in county headquarters that is never connected to the Internet and whose physical security is carefully guarded.

- **Bridging Laptop.** The bridging laptop is used access the Ballot Reconciliation System and to transfer the encrypted and signed votes from the Voting Server to the Mixing Server. This laptop is physically located at Okaloosa Elections office, a few meters away from the mixing server. The bridge laptop runs Red Hat Linux software (version 5) with the latest security patches applied, and its only network connection is through a VPN to the Voting Servers in the Data Center. The Canvassing Board will be provided a username/password to access a web interface to the Ballot Reconciliation running on the Voting Server.

- **Credential provisioning server.** Stores the password used to encrypt the voter signing key in an encrypted form. In November 2008, this server will run on the same machine (administered by Scytl and present in the data center) that is used to host the voting proxy and is Internet-connected.

- **Issuing point software.** The poll worker interacts with the issuing point software, running on the poll worker laptop (also referred to as the authentication laptop later in this document), to load the credential password onto the smartcard. The issuing point software interacts with the Credential Provisioning server to download an encrypted password, decrypt the password, and load it onto a smartcard. The smartcard is then inserted into the voting kiosk to activate the kiosk and enable the voter to cast a vote.

- **Voter registration software.** The voter registration software is the same voter registration software used elsewhere in Okaloosa County, and as such is not part of this voting system and is not provided by Scytl. It is used by poll workers to ascertain voter eligibility, and runs on the poll worker laptop (i.e., the authentication laptop).

3.1.1 Notation

In some parts of the protocol description, we use the following notation for these parties to the protocol:

V**:** The voter.

O**:** The poll workers.

C**:** The voting client.

S**:** This represents the voting proxy and voting service, collectively.

M**:** The mixing service.

P**:** The credential provisioning server.

E**:** The canvassing board.

G**:** The certificate generator process.

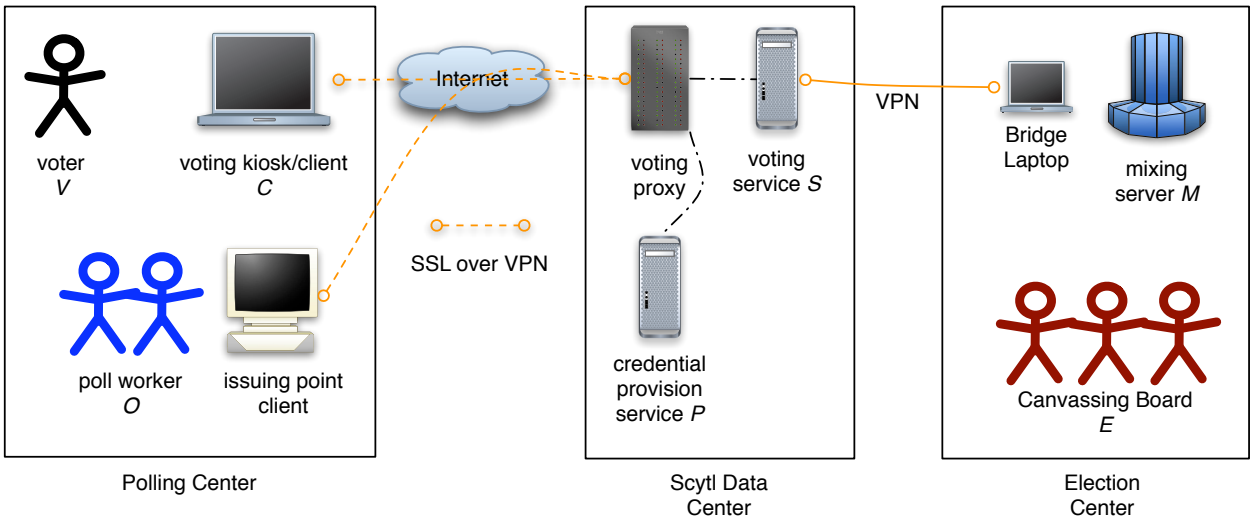

Figure 3.1: Simple System Diagram. This diagram was generated based on our understanding of the documentation we were provided. We compared this diagram with the ODBP diagram in [9, p. 9] and found some differences that we could not reconcile. We recommend that FlDoE compare and reconcile the differences.

3.2 Protocol Overview

The Pnyx.core ODBP 1.0 system uses a cryptographic protocol for encrypting ballots and protecting them from tampering while they are stored or in transit.

3.2.1 High-level View

The following is a highly simplified abstraction of the voting protocol:

1. A voter V prepares a ballot, encrypts and signs the ballot, and finally sends the resulting pair of ciphertext and signature to a voting server. The ballot is encrypted using the public key of the mixing service, and the ballot is signed using the voter's signing key S_V.

2. At the end of the election, the mixing service verifies the signatures of all of the votes, decrypts all of the ballots using the canvassing board private key, and then stores the decrypted ballots in a random order. The resulting list of decrypted ballots is signed using the canvassing board private key.

Cryptography is used to keep the ballot private, to make sure that ballots are not easily forged, and to ensure that each person votes at most once.

3.2.2 Key Management

The previous simplified protocol raises two questions about key management.

- How does each voter get a signing key?
- Who keeps the key used to decrypt all of the ballots?

These issues introduce complications for the protocol, and are handled as follows:

1. Because there is no public key infrastructure for all voters, each election requires the generation of signature keys for each voter. Therefore, a key generation step, which occurs on the Mixing Service before the election begins, generates random new keys (and certificates) for all of the voters. The secret portion of the signature key

is then encrypted with a randomly chosen password. Finally this password is encrypted under the encryption key of the poll workers.[3]

This process results in three files: (a) one file consisting of secret signature keys encrypted under a password, (b) one file consisting of corresponding passwords encrypted under the pollworker keys, and (c) a file consisting of the public parts of the signature keys. Part (a) is transferred manually to the Voting server, and (b) is transferred manually to the Credential Provisioning Service, and (c) is made public (in particular, transferred to the mixing server).

2. There is still an issue of how to get the secret signature key to the voter at the voting kiosk.

   When the voter arrives at the polling place and has been confirmed as a voter, the poll worker downloads the encrypted password for that voter from the Credential Provisioning Service, decrypts it and stores it onto a smartcard. The pollworker then accompanies the voter to the voting laptop, inserts the smartcard, and leaves. The voting client then downloads the voter's signing key from the Voting server and uses the password on the smartcard to decrypt the secret part of the signature key.

3. In order to distribute trust across many parties, the mixing service's decryption key is split into several parts and distributed to different canvassing board officials. During the tally process, these officials need to enter their own parts of the key into the mixing server to decrypt the ballots.

3.2.3 Software, Hardware, and Network Security

A major challenge for any computer-based voting system, including Scytl's, is to ensure that the voting hardware actually runs the approved voting software, and not some malicious software that has been designed to look and act like the voting software. This is a very difficult problem to handle in general. The Pnyx.core ODBP 1.0 voting system attempts to deal with this problem by taking the following steps:

1. The voting client runs on a physically-secured laptop that is under the physical control of poll workers and election officials.

2. This laptop runs a specially hardened version of the Linux operating system that contains a specially configured version of Java that executes only signed applications. Moreover, the signed applications must be signed by the signing key of the Florida Division of Elections.

3. Before using this laptop in an election, the hard drive is physically removed from the laptop, in an attempt to remove most sources of non-volatile storage available on the laptop. (Of course, the BIOS may still provide non-volatile storage, as may other cards.) The laptop's wireless card will not be removed but will be disabled in software by the Live CD.

4. The issuing point laptop (also referred to as the authentication laptop) runs a Windows-based operating system and is connected to two USB card readers. This laptop uses a VPN connection to connect to the Credential Provisioning Service. This laptop runs a fully patched version of Windows XP SP2, with Windows firewall and antivirus software provided by the Okaloosa County Elections Office; the specific configuration was not yet fully defined at the time of this review.

5. The Mixing Service computer is stored in an isolated environment that is not accessible from the Internet (i.e., it is an air-gapped machine). Information is transferred to-and-from this machine in a manual way.

6. All of the machines used in the protocol are configured with software firewalls to limit unauthorized access.

When the polls are open, the voting client and the voting proxy are in communication via an Internet connection. Several steps are used to protect the system against network threats:

1. All ballots are encrypted by the voting client before being transmitted to the voting proxy.

[3] The use of this encryption key is documented in [27].

2. The OpenVPN virtual private networking software is used to secure the public Internet channel between the voting client and voting proxy, as well as the Internet channel between the issuing point software and the credential provisioning service. The OpenVPN software is open source, off-the-shelf software for establishing a secure channel over an insecure network.

3. The voting client and voting proxy communicate via a HTTPS (SSL-encrypted HTTP) connection that is tunneled over the OpenVPN tunnel. The voting proxy machine also establishes a Linux iptables firewall to limit incoming packets so that it can only be contacted by authorized voting clients. The credential provisioning service can only be contacted by authorized issuing point laptops.

4. The voting client and voting proxy communicate using a proprietary cryptographic protocol designed by Scytl, which is tunneled over the HTTPS connection (which is in turn tunneled over the OpenVPN tunnel).

The voting kiosk software is loaded via a two-step process. Before the opening of polls, poll workers insert a special Live CD prepared by Scytl into the voting kiosk laptop. The Live CD is a bootable CD that contains a hardened Linux distribution, a Java virtual machine, and certificate chains for many of the parties involved in the voting protocol. When the Live CD boots, it executes scripts written by Scytl that establish an OpenVPN tunnel with the voting proxy and then download the voting client Java application. This Java application is received as a signed JAR. The JVM checks that the Java application contains a valid signature and then executes the Java application that is used for the election. The scripts also set up a software firewall using the Linux iptables software to limit incoming connections: in particular, incoming packets are allowed only if they are related to the OpenVPN connection between the voting kiosk and the voting proxy. The software is configured using hard-coded IP addresses to avoid any use of DNS.

3.2.4 Trusting Officials

Certain parts of the protocol are intended to limit the amount of trust needed in the poll workers.

1. There are at least two poll workers at every polling center.

2. The poll worker never has direct access to the voter's signing certificate. The poll worker executes the issuing point software, which downloads a login and (encrypted) password from the credential provisioning service and loads it onto a smart card. The poll worker physically transfers the smartcard from the issuing point laptop to the voting kiosk, but the poll worker never learns this login and password. The voting client then downloads the voter's encrypted signing key and certificate from the voting proxy and uses the password on the smartcard to decrypt the voter's signing key.

3. The communication between the issuing point client and the credential provisioning service are encrypted, under certain keys. These keys are also stored on smart cards and are never disclosed to poll workers. Thus, it is more difficult for a conspiracy of officials to copy or exchange keys.

3.2.5 Voter Choice Records

Before the voter casts her ballot, the voting kiosk prints a copy of her selections onto a piece of paper, called the Voter Choice Record (VCR), using a printer connected to the voting kiosk laptop. The voter is asked to inspect this VCR. If she does not approve of its contents, she change her selections and print a new VCR. At most three VCRs may be printed. If the voter does approve, then she may cast her ballot, at which point an electronic record of her vote is encrypted and transmitted to the voting proxy, and the voter is instructed to carry the VCR to poll workers and deposit it in a VCR receptacle for safekeeping.

After the election is concluded, the poll workers secure the VCR receptacle and carry it with them as they fly back to Florida, making sure to preserve the chain of custody of these paper records. The VCRs can then be used in audits or recounts. The documentation provided to us contained no discussion of audits or recounts. We were informed that auditing procedures were under discussion at the time of the review.

A sample VCR, provided to us by Scytl to illustrate the general format of these records, is reproduced in Figure 3.2. We warn that the contest names are not intended to be representative of a real election, and the format of the VCR might change slightly before November 2008. The VCR contains:

- In each contest, the name of that contest, and a list of all selections made in that contest.
- Instructions to the voter.
- Information that identifies the election in which this was cast.
- A single letter (A, B, or C) that identifies whether this was the first, second, or third VCR printed by the voter. Normally, this letter will be A, but if the voter changes her selections and prints a new VCR, then the second VCR will contain the letter B, and the third VCR the letter C.
- A unique session ID, composed of a meaningless string of numbers and digits.
- A horizontal line for folding.

The procedures that a voter will follow to submit their VCR(s) was yet to be determined at this time of this review, but one possible procedure might be as follows. The voter folds the VCR(s) along the horizontal line, then shows her folded VCR(s) along with her folded Voter Receipt to the poll worker. The poll worker validates the visible information and then asks the voter to deposit all of the VCR(s) into the VCR receptacle.

3.3 Protocol Details

Here we describe the voting protocol steps in more detail.

3.3.1 Key Generation Step

Each voter is associated with a unique digital signature key. These keys are generated by the ODBP Credential Generator application, which is executed in the Mixing Server before the election starts via the following steps:

1. The canvassing board provides a comma-separated values file containing the voter data required for the electronic voting process.
2. This file is fed into the ODBP credential generator application which does the following:
   (a) for each voter V in the input file, generate a (private) digital signing key S_V and a corresponding (public) verification key P_V,
   (b) generate a certificate $Cert_V$ for P_V using the election Certificate Authority key,
   (c) store S_V and $Cert_V$ inside a PKCS#12 container $Cont_V$ protected by a randomly-generated password p_V,
   (d) encrypt p_V using P_O, the public key of the poll workers, yielding $EncPass_V$,
   (e) store $Cont_V$ and $Cert_V$ together with the assigned voter ID in a database that will eventually be transferred to the voting service via removable media,
   (f) generate a random voter login id $Login_V$, and
   (g) store $Login_V$ and $EncPass_V$ with the assigned voter ID in another database that will eventually be transferred to the Credential Provisioning Service via removable media.

3.3.2 Voter Sign-in Step

When a voter arrives at the polling place, she checks in with the poll workers and they perform the following steps to verify that the voter is eligible to vote:

1. The voter authenticates to the poll worker by presenting a photo ID.
2. The poll worker verifies that the voter is authorized for the election. This verification is done using the the Voter Registration system already employed by Okaloosa County.

A

Voter's Choice Record

Okaloosa County – General Election – November 4th, 2008

K8dT-fKa3-KMaH-A

Show only this part to Kiosk official

This part must be kept hidden

Instructions

Review that the printed options match your selections on the screen and press "Cast Ballot".
If these printed options do not match your selections, or if you change your mind, please, press the "Review" button.

Selected Options:

PRESIDENT
BADNARIK/CAMPAGNA

SENATE

CONG 1

FL HOUSE 1

SHERIFF

SUP OF ELECTIONS

SUP CRT BELL

SUP CRT CANTERO

CRT APP ALLEN

CRT APP BARFIELD

CRT APP HAWKES

CRT APP KAHN

CRT APP PADOVANO

CRT APP WOLF

CNTY JUDGE1

SCHL BRD 2

AMENDMENT 1

AMENDMENT 2

AMENDMENT 3
NO

AMENDMENT 4

AMENDMENT 5

AMENDMENT 6

AMENDMENT 7

AMENDMENT 8

Race 180

Race 220
Candidate 10

Race 250

Race 350

Race 800

Race 820

Figure 3.2: A sample of a Voter Choice Record. Credits: Scytl.

3. If the previous step is successful, the official retrieves the login and password for the voter. The login and password are the voter credentials required for voting on the voting laptop. These credentials are stored encrypted on the Credential Provisioning Service and only accessible by the ODBP issuing point through a VPN. They are retrieved as follows:

   (a) The poll worker has a PIN-protected smartcard. This smartcard stores a secret key S_O corresponding to poll worker public key P_O. The card can be accessed only after entering a PIN known to the poll worker. The poll worker inserts this smartcard into the poll worker laptop, where the issuing point client runs.

   (b) The poll worker creates a paper document called a Voter Certificate, using the Voter Registration software on the authentication laptop. The Voter Certificate lists the election id, the precinct number, the ballot style, the voter name, the voter address, the date of birth, and the voter registration number. The voter physically signs this certificate, with pen to paper. (The term Voter Certificate refers not to a cryptographic certificate, but rather to a paper document that is produced to keep a record that the voter has voted.)

   (c) The poll worker scans the Voter Certificate for later upload to the Okaloosa registration server.

   (d) The poll worker executes the ODBP issuing point client, which connects to the Credential Provisioning Service and requests the voter login and encrypted password associated with this voter's Voter ID.

   (e) The issuing point application receives $Login_V$ and $EncPass_V$ from the Credential Provisioning Service, along with the precinct ID and ballot style assigned to this voter. The issuing point client decrypts $EncPass_V$ using key S_O stored on the poll worker's smartcard, obtaining password p_V. Then, the precinct ID, ballot style and voter ID are shown to the poll worker.[4]

   (f) The poll worker verifies or corrects the precinct ID and/or ballot style of the voter if necessary.

   (g) The poll worker creates a smartcard for the voter. This PIN-protected smartcard stores the precinct ID, ballot style, $Login_V$, and p_V. The PIN for this smartcard is created during the election configuration phase and entered by the poll worker into the voting and authentication laptops when they are booted. Consequently, the PIN is known to the issuing point and the voting client software, but the PIN is not known to the voter. Therefore, voters cannot themselves access the credentials stored in the smartcard.

4. The voter and poll worker proceed to the voting booth where the voting kiosk laptop is located. The poll worker inserts the voter's smartcard into this laptop. The poll worker leaves.

3.3.3 Voting Protocol

This section contains a summary of the steps of the protocol associated with casting a vote. We provide a more detailed version in Appendix A.

As described above, immediately before the polling period begins, the voting client and the voting proxy establish an SSL connection tunnelled through an OpenVPN connection. The client downloads a signed voting application and begins executing it. (The voting client operating system contains a version of Java that only runs signed Java applications; the operating system also contains the public key certificate of the Division of Elections so that the received Java application is automatically verified.)

Then, for each voter, the following steps are performed:

1. The voting client reads $Login_V$ and p_V from the smartcard which has been inserted into the voting client by the poll worker.

2. The client sends $Login_V$ to the voting proxy, which responds with PKCS#12 container $Cont_V$. The client uses password p_V to decrypt $Cont_V$ and recover signing key S_V.

3. The client retrieves a ballot design from the voting proxy and displays it to the voter. The voter makes choices on the ballot. After voting in every race, the voter is given an opportunity to review his/her selections.

4. Once the voter confirms his/her selections, the voting laptop prints a Voter Choice Record (VCR). The VCR includes the voter's choices, in cleartext, and a random *session id*. The voter either confirms that the VCR accurately reflects their intent and moves to the next step in the process, or chooses to change their selections by returning to the previous step in the process.

[4] The fact that this is shown to the poll worker is documented in [27].

5. The voting client chooses a ballot identifier B_{id} (a random number) and encrypts a pair consisting of the voter's choices and B_{id}, yielding encrypted ballot B. The encryption method is a a hybrid encryption under the public key of the canvassing board. The voting client also signs, using voter signing key S_V, a hash of a tuple derived from B, B_{id}, $Login_V$, and the election id. This yields receipt request R_r. The pair (B,R_r) is sent to the voting service.

6. The voting service acknowledges the vote by sending back receipt R, which is the voting service's digital signature on the hash of a tuple derived from B_{id}, the election id, and $Login_V$.

7. The client prints the Voter Receipt (which is also referred to as the "Counted as Cast" receipt). This includes the election id, $Login_V$, B_{id}, a timestamp, and receipt R.

3.3.4 Ballot Reconciliation Step

The ballot reconciliation step allows the Canvassing Board to review each voter's eligibility and signature and determine whether a ballot cast by that voter will be accepted as valid. If a ballot is rejected, a reason must be recorded. For example, if the voter was not eligible to vote, or the signature on the voter certificate does not match the voter's signature on the file, the ballot may be rejected. We did not have time to study the ballot reconciliation step during this review.

3.3.5 Mixing Protocol

The mixing service is responsible for decrypting and anonymizing the ballots. After the end of the election, the list of encrypted ballots (as collected by the voting service) are transferred to the mixing service. This is done in two steps: the encrypted and signed ballots are downloaded from the Voting Server to the Bridge Laptop. They are then moved onto removable media (such as a USB stick) and the removable media is then inserted into the mixing server. The mixing service then reconstructs the canvassing board's private key from the canvassing board members' shares, checks the voter's digital signature on each encrypted ballot, ensures that at most one ballot is present from each voter (by discarding duplicates), decrypts all of the encrypted ballots, randomizes the order of the decrypted ballots, and then signs the list of decrypted ballot under the canvassing board's private key.

Election officials can request a 'results' report from the mixing service, which contains a list of the cleartext ballots (after shuffled into random order) signed under the canvassing board's private key; this can be used to tally the votes. Election officials can also produce a list of the ballot ids of the ballots that were included in the tally, in random order and signed under the canvassing board's private key. This signed list of ballot ids can be posted upon the county's web page, so that voters can check whether their vote was included in the tally by checking whether the ballot id printed on their receipt is listed on the county web site.

Further details can be found in Appendix A.

3.3.6 Post Election Protocol

At the end of the election and after results have been tabulated and reported, the following steps take place:

1. Scytl copies all of the data collected by the Voting Service onto DVDs or other appropriate backup media and delivers this data to the Okaloosa Elections Office.

2. Scytl thens erases all of the information from its servers and databases, without maintaining any backup copy. Scytl has informed us that they are working with the Okaloosa Elections Office to write a more formal procedure for this step, but that it is not expected to be ready until September 2008.

   The use of a secure erasure product has been specified. In particular, Scytl indicates that it will use the DBAN tool (http://www.dban.org/) configured to use a US DoD-recommended wipe method. DBAN performs a low-level secure erase of the entire hard drive, and in our opinion, it appears to be a good choice for the purpose for which it will be used.

3. It has not been determined how the canvassing board members will dispose of their secret key information.

4. Per law, the county elections office must keep the ballot information for 22 months. It is our understanding that this information will be erased after that period. The county elections office is responsible for maintaining the custody and security of this information during the 22 month retention period, and for erasing the information afterwards. It is not yet clear how the 22 month legal retention requirement will be fulfilled in this context. Written instructions for this procedure were still under development at the time of this review.

3.3.7 Addendum

A first draft of this report was submitted to FLDoE and Scytl, per the Statement of Work. Scytl's comments on the draft revealed a new fact about the voting protocol: The session id from the printed Voter Choice Record is also included in the encrypted vote sent to the voting service. (See §4.2.1.3.) This is not documented in the protocol specification Scytl provided [25]. Furthermore, during the review period, neither Scytl nor the review team discovered the absence of this in the protocol specification that the review team constructed and Scytl inspected [27].

Since the review team had no knowledge of this part of the protocol design, or the code implementing it, this report contains no evaluation of its implications for the security of the voting system in general, or for ballot secrecy in particular.

Chapter 4

Findings

As noted earlier, the Statement of Work detailed several specific questions that are relevant to the FLDoE certification process. We provide our findings relative to those questions in this section. We note that, because of this format, some findings are redundant because they apply to more than one question.

4.1 Software Quality Analysis

4.1.1 Is accurate complete life-cycle documentation available?

We did not find complete voting system lifecycle documentation. There were significant gaps in the documentation that we were provided and we found documentation inaccuracies relative to the ODBP documentation [9].

In addition, we did not find documentation of the software engineering processes that were used to develop this software. Several competing models for the software life-cycle exist, e.g. waterfall, spiral, iterative, etc. It was not possible to determine which of these models was used from the given documentation and source code. However, these models generally contain at least the following steps: requirements, architecture, design, implementation, testing, integration, deployment, and maintenance. Documentation regarding several of these steps was lacking.

We found no documentation of accepted practices for assuring code quality, such as testing plans, requirements for peer code review of all source code that is checked in to the software revision control software, documentation of design decisions and design reviews, and use of static analysis tools.

Action: We strongly contend that voting systems vendors should practice rigorous software engineering practices. All voting systems submitted for certification should completely document the development life cycle. Vendors should ensure that the documentation accurately reflect the voting system submitted for certification.

Action: FLDoE should consider requiring thorough, accurate documentation of these software engineering aspects.

In response to our inquiry, Scytl informed us that they have never had any bugs in any version of their software released outside Scytl that could cause votes to be recorded or counted inaccurately. This is positive, but not conclusive. We again emphasize that rigorous software engineering processes are necessary for critical applications such as voting systems.

Finally, as we detail in Section 2.6 above, we only reviewed documentation that was presented to us and it is possible that we did not receive documentation relative to this finding due to miscommunication regarding project scope or other misunderstandings. Thus, we do not find that the documents do not exist, rather only that they were not part of the package that was provided to us by FLDoE.

4.1.1.1 Finding: The cryptographic protocol specification needs improvement

The cryptographic protocols implemented by the Pnyx.core ODBP 1.0 voting system are specified and documented in [25, §5]. Whereas these provided specifications are substantially complete, they also are sometimes incomplete or ambiguous. This was particularly the case with the initial steps in the voting protocol: ODBP requirements [9] here supercede the provided specification, and no detailed cryptographic specification implementing these requirements

was provided. Also, the inclusion of the Voter Choice Record's session id in the voting protocol (see §3.3.7) was not documented in the provided specifications. In general, we believe that the provided specification fell short of the standard expected in the cryptographic literature.

Since this system relies on these cryptographic protocols for its security, and since every detail in a cryptographic protocol is potentially crucial, we see it as essential that the protocols be fully and unambiguously specified in the documentation. Only then can code reviewers and maintainers effectively understand the requirements for their correct implementation.

The review team, assisted by Scytl, spent a substantial amount of review time completing a detailed specification. Since this was time not spent on actual analysis of the protocols, the effectiveness of this review process was reduced. This also negatively impacted our ability to inspect the code and determine whether the code correctly implements the intended protocols.

Action: For voting systems that employ proprietary, novel, or non-standard cryptographic protocols, election officials should require that a full specification of these cryptographic protocols be submitted as part of the application for certification.

Note: By *specification*, we mean a complete description of the cryptographic protocol, with sufficient detail to enable an independent software developer to fully implement the protocol without being forced to guess at the designer's intent. Specifications should use notation that is standard in the literature (e.g., [17, 24]), for example:

1. $A \longrightarrow B$: n
2. $B \longrightarrow A$: $\mathsf{Sign}(n; k_B)$
3. ...

The minimum essential features are that the specification identifies all messages, their order, the sender and receiver, the complete message contents (at the symbolic level of cryptography), as well as a description of what checks recipients are supposed to perform on the messages they receive.

4.1.1.2 Finding: Test plan and results documents were not included in the material submitted for evaluation

Explicit documentation of system, unit, and integration test plans was not included in the material to us. Some unit tests were found with the provided source code, however, we did not perform a detailed analysis of the coverage provided by these tests.

Action: For system certification, FLDoE should require detailed test plans, exercised test sets, and test output from the vendor.

4.1.1.3 Finding: Documentation of the threat model and security architecture was not included in the material submitted for evaluation

We found no clear documentation of:

- the threat model that the voting system was designed to resist,
- the security goals that the system is designed to achieve,
- the system architecture, or
- the design rationale.

By the system architecture, we refer to a clear and accurate description of all system components and how they relate; such a description is found in Chapter 3 of this report. The system architecture should include a list of trusted components and what purpose they are trusted for. It should not include inaccuracies or irrelevant elements such as those described in finding 4.1.1.5.

By the design rationale, we refer to a description of the design choices that were made and an explanation of how the design responds to the security goals and threat model, with justification and thoughtful analysis of these design choices[1].

As we noted earlier, important elements of the system, such as the cryptographic protocols, also were not clearly documented.

4.1.1.4 Finding: No assurance document providing convincing evidence that the voting system meets the requirements, under specified conditions, was included in the material submitted for evaluation

As we noted earlier, cryptographic voting systems represent a paradigm change in voting systems. Because of the pivotal use of cryptography, the software implementing these systems is more nuanced and difficult to understand. Thus, cryptographic voting systems demand rigorous attention to documentation, correctness proofs, test set retention, etc.

As part of this documentation, we would expect to see an assurance document, i.e., a document that provides a clear and convincing argument, supported by concrete evidence, that the system meets its requirements, under specified conditions and in a specified environment. In other words, we would expect the designers to provide positive evidence that the system meets its requirements, to design the system in such a way that this evidence is available, and to document this evidence and the argument that the system meets its requirements. We would also expect the designers to document the conditions or assumptions under which the system is claimed to meet its requirements. We would expect these claims to be supported by appropriate analysis.

However, we found no evidence of any such document, no evidence that any such analysis has been performed, and no evidence that the system was designed to facilitate construction of such an argument. Based upon this, it is our conclusion that the voting system was not designed and implemented following practices that are customary for high-assurance systems. As a consequence, independent examiners, such as ourselves, have diminished ability to be confident in the voting system software. Thus, certification decisions are more difficult to reach and security confidence is generally weakened.

4.1.1.5 Finding: System documentation contained occasional inaccuracies

The Pnyx.core ODBP 1.0 voting system is built upon Pnyx.Core, a library that provides a platform for building voting systems. We understand that Scytl has built multiple voting system products upon this common core, and the system analyzed in this report is just one of those products. We found a number of places where the documentation submitted for certification contained statements that appeared to refer to other Scytl products, rather than to this system.

For instance, Section 6.3 of the "Election Project Quick Guide" document states that the Voting Client software is delivered as a Java applet (implying that it is downloaded by a web browser running on the voting client, and the Java applet is executed by the browser). In fact, this is not accurate for the system evaluated in this report. Instead, in the system we examined, the Voting Client software is a Java application that is executed directly by the voting kiosk laptop. No web browser is involved. The difference between a Java applet and a Java application has important security consequences, so this kind of misstatement made it more challenging for us to review the security of the system.

As another example, Section 5.1 of the "Election Project Quick Guide" states that the Voting Proxy and Voting Service would normally be installed and running on the same computer. However in a conference call Scytl informed us that in fact they intend to install these two software applications on two separate computers. Likewise, Section 6.1 of the "Election Project Quick Guide" describes a default configuration where the Voting Proxy can be accessed by `http` (unencrypted) connections. However Scytl informed us that in fact they intend to configure the Voting Proxy so that it will only accept `https` (encrypted) connections.

Interestingly, in both cases Scytl's intended configurations reflect better practice than what is documented, but it is nonetheless critical that documentation accurately reflect the actual deployment scenario.

[1] For instance, a design rationale might answer questions such as: In what ways does the chosen architecture respond to the threat model and prevent the most likely risks? Why is the system designed to require an always-on Internet connection, rather than using an alternate architecture where votes are transmitted once at the end of election day to reduce the risk of Internet-based attack? Why does the system use DREs, rather than manually marked paper ballots? Do the design choices increase or decrease vulnerability to insider threats, and why?

4.1.2 Is the architecture of the software well-suited to the requirements of the voting system?

The software architecture is essentially a client-server system, in which the voting clients are connected through the Internet to the voting server. To authenticate voters, poll workers have an additional laptop at each voting site that also connects to the voting server through the Internet. The voting server collects cast electronic ballots in a database, and at the end of the voting period the electronic ballots are transferred to the mixing server through the use of removable media (as the mixing server is not network connected).

For voters to vote, the network connection between the polling place and the voting server must be active. Network outages due to non-malicious activity (e.g., equipment failure or downtime for system upgrades) are certainly not unheard of. Such an outage might be unlikely for the voting server's network connection, since the voting server will be located in a data center (which is likely to provide redundancy to reduce the risk of such failures). However, it is not uncommon to experience outages or intermittent connectivity on consumer network connections—i.e., at the voting client's network connection. It is also possible that network outages could be caused deliberately. A successful denial of service attack (e.g., see the discussion in sections 4.2.2.4 and 4.3.9.2 of this report) against either end of the network connection between the voting client and the voting server would result in voters not being able to vote using this electronic system for the duration of the outage.

The requirement for continual network connectivity during the entire time that the polling place is open may also be problematic for another reason: There will always be a network path through which the voting client and server may be reached. Remote systems (i.e., outside systems not intended to be part of the voting system) may therefore attempt to communicate with the voting client or server at any time. While traffic on such network pathways might be filtered or blocked (e.g., through the use of a firewall or VPN), such filtering may fail (e.g., a new vulnerability in the filtering or VPN software may be discovered prior to or during the voting period). As a result, the risk of compromise for both the voting client and the voting server is increased.

The risk associated with these issues could be reduced if the system were not dependent on continual network connectivity whenever the polling place is open.

4.1.2.1 Finding: The Voting Client relies on software that is downloaded across the network at run-time, and only stored in volatile memory

As described in finding 4.2.2.3, the voting kiosk downloads a Java application from the Voting Server at the start of the election, and it is this software that is then executed to implement the voting functionality on the voting kiosk. As this software is only stored in volatile memory on the voting kiosk, it is difficult to determine what software was actually used during an election. For instance, it is difficult to verify whether the software that was executed matches the version that was certified by the State of Florida, and there may be no way to check whether malicious code was introduced. The ability to perform such an audit is a desireable property for an election system, and the current software architecture does not easily enable such an audit to be performed.

At best, the current process ensures that some application digitally signed by the Florida Division of Elections is executed on the Voting Client. However, it is possible that multiple applications could be digitally signed by the FLDoE key. For example, an updated version of the Java application could be produced and properly signed to address some bug fix, or an insider with access to the signing key could sign a malicious application of their choice. In either of these cases it would not be possible to determine from an audit of the Voting Client which Java application was downloaded and executed. In contrast, a Live CD which included a static version of the Java application would allow an auditor to determine which Java application was executed by conducting an examination of the CD at any point during or after the election. This approach would also not require trust to be placed in the FLDoE cryptographic key, the people with legitimate access to it, and any processes designed to protect it from outsiders.

Placing the Java application on the voting server does not provide any added protection for the voting client, as anyone with the ability to modify the Live CD could simply configure it to ignore the Java application retrieved from the voting server, and run an alternate malicious application of their choice.

Action: Include the Java Application on the Live CD and use that static version to run the election, rather than relying on code downloaded over the network.

4.1.3 Is the target software build environment complete and properly documented?

The target software build environment is complete and accompanied by proper documentation. The software components can easily be built for Windows or Linux. For Windows, Visual Studio Solution files were provided to build the software. Similarly, for Linux the appropriate shell scripts were provided. In addition, options for building debug and release variants of the software are available for each platform. We commend Scytl for providing a full build environment, with documentation; it aided the review.

4.1.3.1 Finding: The resources necessary to deploy a functional voting system were not available to the review team during the audit period

The source code was provided to us (excluding the cardlet component), and the virtual machines provided included a complete build environment (including appropriate documentation). This made it easy for us to build the software.

However, the additional step of producing a fully functional voting system from the compiled software required resources that were not provided to, or explicitly requested by, the review team (see 2.6). Such resources include software deployment and configuration documentation, and in at least one case (the Voting Client) specific hardware. While the deployment of a fully functional voting system is not specifically required by the statement of work for this audit, it would have provided us with the opportunity to perform several types of additional testing that could have directed our focus, validated our results, and allowed us to view the system in a somewhat more realistic context.

We note that FLDoE conducted functional testing and we offered them some suggestions relative to that process, but we were not able to conduct functional testing, security testing, or other testing on our own, nor were we provided with the results of the FLDoE testing.

Action: FLDoE should consider requiring that the vendor make a fully functional election system available to reviewers during future audits.

4.1.4 Is the target software structure well-defined and evident from its presentation?

The target software seems to be structured in a reasonable way. The abstractions provided by both of the main implementation languages, C++ and Java, are used in an effective manner. More specifically the C++ code makes effective use of namespaces, classes, and functions. Similarly, the Java code makes effective use of packages, classes, and functions. The cogent use of these abstractions leads to high cohesion and reusability, both of which are considered a desirable property of software [16].

The software contains many wrapper classes and levels of indirection. This forced the team to chase references to defined classes and functions through multiple layers when we were reviewing the code. This appears to be at least partially due to the fact that the Pnyx.core software is intended as a general-purpose library that can be used as a basis for different kinds of voting products; this leads to a level of generality and complexity that is not necessary for the voting system we reviewed and that partially obscures understanding of the code. The tradeoff is that this allows the code to be used in more circumstances, but introduces complexity not necessary for the ODBP project.

4.1.5 Is the target software internally, accurately documented?

The level to which the code was internally documented (i.e., commented) was inconsistent. In some files the code was heavily documented, with comments at least at the level of every function definition, whereas others only included sparse documentation. However, this inconsistency did not present a significant challenge for the investigators during the review.

Many of the function level comments were in a format consistent with the format used by automated code documentation tools (such as Javadoc).

4.1.6 Are the programming language(s) that are employed well-suited to their use?

The software was written using two programming languages, namely Java and C++. While Java does appear to be well suited to the tasks for which it was used, the same cannot be said of C++. While it is an extremely widely used programming language, C++ presents several inherent challenges to writing high assurance software. For example:

- C++ relies on the software programmer to correctly implement memory management in order to safely allocate, deallocate, and reference memory during program execution. The failure to implement such operations correctly has been the cause of, or a contributing factor in, a large number of software vulnerabilities in many software products in recent decades.
- C++ provides no inherent support for concurrency in multi-threaded programs, and as such the programmer is again left to implement thread-safe code correctly.
- Producing C++ code which uses exceptions consistently and correctly is challenging, and relies on the programmer's in-depth understanding of the nuances of exception handling [5].

The Pnyx.core ODBP 1.0 C++ code made heavy use of concurrency, exceptions, and APIs that require careful handling of memory management, so all three of these challenges apply to the Scytl C++ code. While it does appear that in many places Scytl programmers have used C++ APIs that reduce the risks associated with using C++, and while no clear vulnerabilities were identified in the Scytl C++ code during this audit, the risk associated with using C++ is this context remains. In the absence of a compelling reason to use C++ an alternate language (such as Java) which includes features such as automatic memory management, array bounds checking, and concurrency between multiple threads would be more appropriate.

In some cases C++ offers a performance advantage over Java, and in such circumstances the choice of C++ over Java may be reasonable. However, this does not appear to be one of those cases, as the need to limit the risk of compromise would seem to be more important than raw performance for election system programs. In addition, while not all of the C++ code was reviewed in the timeframe available to the auditors, the sections that were reviewed did not appear to have been written with high-performance as a goal, making it less likely that C++ was chosen for that reason.

4.1.7 Does the target software follow software engineering practices that are well-suited to avoid or minimize the risk of security vulnerability?

Software engineering is a process that involves all phases of the development of a software product, from requirements gathering and the initial design to support of the deployed product. As such it is almost certainly not possible to make a definitive statement about the software engineering practices of Scytl based on the review of a static snapshot of this software. A complete analysis of the software engineering practices used by Scytl was outside the scope of this audit.

However, during the audit it was possible to observe some indicators of the suitability of the software engineering practices used by Scytl. While these indicators certainly only show a small part of the picture, they do suggest that some areas of Scytl's software engineering practices could be improved to reduce the risk of security vulnerabilities.

As noted earlier in Finding 4.1.1.1, while definitions for the cryptographic protocols were provided to the review team, these definitions were not sufficiently detailed. Producing complete documentation is a vital component of software engineering, and the lack of detailed cryptographic specifications is an indicator that Scytl's software engineering practices could be improved.

The following two findings also provide an indication of areas where software engineering practices may need improvement.

4.1.7.1 Finding: Insufficient and incomplete use of static analysis tools

Scytl makes insufficient and incomplete use of commonly used techniques that can identify likely bugs, including those that may be the cause of security vulnerabilities. Such tools include static analysis and, to a lesser extent, fuzzing. While some static analysis tools do appear to have been employed by Scytl for some of the source code specific to the ODBP, the tools used (Checkpoint and Findbugs) are limited to the Java programming language and perform a fairly shallow, "lint"-like analysis (checking for stylistic issues and a specific class of easy-to-detect bugs). As such, much of the code (i.e., all of the C++ code, and all of the Java code with the exception of the ODBP-specific modules) has not been subjected to static analysis during the development process.

Fuzzing is a technique commonly used to identify bugs by essentially providing malformed input to a program. In this case, such testing would likely have been useful for testing the resilience of the various communication protocols used. While this testing technique is less commonly used by developers than static analysis, its use is growing for projects that contain network-facing applications, such as Scytl's Pnyx.core ODBP 1.0 system.

Action: Voting systems submitted for certification should be required to integrate static analysis into their development cycle and to provide static analysis results as part of the Technical Data Package.

4.1.7.2 Finding: Changes to the source code do not appear to result in changes to the external software version number

Rigorous, effective version control is critical in software development efforts, particularly those involving mission critical applications such as voting systems. During the analysis phase, Scytl updated a portion of their code in response to a vulnerability identified during the review, and submitted that change to the Florida Department of Elections. While is is commendable that the vendor responded promptly to this detected vulnerability, it does not appear that this source code modification resulted in a change to the external software version number (it remained at 1.7.1c). This means that there are two versions of the software, both of which use the external version number 1.7.1c, where one of these versions contains the potentially vulnerable code, while the other does not. Scytl reports that they did change their internal version number in response to this change (updating it from 1.7.1c to 1.7.1c1), but such a change is apparently not reflected in the version number reported by the software, and as such does not allow a user to determine if the software running on a particular system is the updated version or not. This practice is dangerous and can result in incorrect versions being inadvertently installed into operational systems.

Action: All voting systems software submitted for certification should follow rigorous, effective version management practices, to include the update of external version numbers in response to changes in the software.

Action: Scytl, in cooperation with FLDoE, should immediately update the reported version number for the code modifications that we report here.

4.1.8 To what extent does the software follow defensive software engineering practices?

Defensive programming is a technique that is often recommended for software where security or reliability is critical. It involves writing code to check assumptions near where they are relied upon, even if it seems that those checks are unnecessary or redundant. Defensive programming is generally considered to be good practice and beneficial to software robustness.

Unfortunately, we did not have sufficient time to evaluate to what extent the software follows defensive software engineering practices.

4.1.9 Is the structure and implementation of the software well-suited to the security requirements?

Unfortunately, we did not have sufficient time to evaluate the structure of the software in a careful or systematic way. We generally found the structure of the software to be clear and in accordance with our understanding of the system architecture. See also Sections 4.1.4 and 4.2 for further analysis related to this topic.

We did notice possible issues in a few places, detailed below.

4.1.9.1 Finding: We identified two bugs in certificate validation code

During manual source code analysis, we identified two bugs in the code that performed certificate validation. Each bug causes validation to performed improperly, in two different ways, on certificate chains of length 2 or longer. Apparently, these bugs have no impact on the November 2008 ODBP because all certificates in that pilot will be directly signed by the CA.

On the positive side, the structure of the code was such that the bugs were immediately clear upon reading the corresponding sequence of code. The bugs were accidental logic flaws that caused the code sequence to "make no sense." On the negative side, the bugs were readily apparent enough that it raised concerns in our eyes about the software engineering and quality assurance processes in use by the vendor.

Scytl informed us that these two bugs were already known to them and had been classified as a low priority because they were not expected to impact the November 2008 pilot. We are concerned that unfixed bugs can be dangerous, because it is not always clear to be certain what the impact of these bugs may be. Occasionally bugs that are believed

to be harmless turn out to have unexpected negative consequences, and it is not easy to rule out this possibility. In addition, unfixed bugs make code review harder and may present a hazard for code maintenance. For these reason we suggest that, in mission-critical systems such as election software, it would be safer to fix all known bugs.

4.1.9.2 Finding: We identified several logic errors in the retry logic in the voting client

During manual source code analysis, we identified several logic errors in the voting client code that attempted to recover from connection errors by retrying the connection. In particular, if the client attempts to connect to the voting proxy but encounters an error, then the client will retry this connection. However, due to bugs in the exceptional code path for handling these errors, the client could retry a second time, ignore the second response from the server, and then use the first response from the server (taken from the first connection attempt) in subsequent communications, even though this first response might be invalid.

Scytl informed us that they consider this a low priority issue, because OpenVPN is responsible for ensuring that lost packets are retransmitted, and consequently this code path should not be used in the ODBP. Accordingly, Scytl stated that they do not expect this to have any impact on the November 2008 ODBP. Unfortunately, we found it difficult to be certain that this code path will never be executed, and it seems like bad practice to allow known bugs to remain in the code.

Action: Voting system developers should be required to repair all known bugs in software submitted for certification.

4.2 Security Architecture Analysis

4.2.1 Does the target software contain effective controls or safeguards to protect ballot secrecy?

The software presented to us contains a number of mechanisms and safeguards designed to protect ballot secrecy. These mechanisms seem likely to be effective against malicious outsiders, though they are not structured in a way that would protect against misuse of authorized access by insiders. On the whole, the level of ballot secrecy provided by the Pnyx.core ODBP 1.0 system appears to be comparable to, if not significantly better than, the level of ballot secrecy provided by the voting methods it is intended to replace, which include absentee voting by mail, fax, and unencrypted email.

In more detail, we analyze several different aspects of ballot secrecy:

- *Ballot secrecy against outsiders:* To ensure that voters can vote according to their conscience, most voting systems are designed to prevent third parties from attributing votes to voters. The Pnyx.core ODBP 1.0 system attempts to prevent outsiders from learning how voters voted by encrypting votes and randomizing the order in which they are reported. These mechanisms appear to be largely effective at protecting against outsiders, with one exception.

  We discovered a flaw in one relevant protection mechanism. Though the vendor submitted changes to fix this flaw, as detected, this flaw may pose risks to ballot secrecy in some scenarios. Recall that votes are encrypted at the polling place and decrypted by the mixing service. Election officials may optionally cause the mixing service to output a list of decrypted votes, e.g., to double-check the mixing service's tally of the votes. To protect voter anonymity, the mixing service shuffles the votes into a pseudorandom order before including them in this report. Unfortunately, as discussed in Section 4.3.4.1, due to defects in the code, the pseudorandom order used by the mixing service is predictable. As a result, an individual who obtains access to this report may be able to learn how specific voters have voted.

  Scytl informed us that they have prepared a fix for this flaw. Once this defect is fixed, we expect that the voting system will effectively protect ballot secrecy from outsider threats. Even if it is not possible to deploy the fixed software immediately, we expect it should be possible to introduce procedural controls to address this defect, e.g., by ensuring that no election worker ever invokes this reporting functionality.

- *Ballot secrecy against insiders:* Malicious insiders may be able to attribute votes to voters.

  There are several ways that insiders might be able to compromise ballot secrecy. Most importantly, the software stores electronic records that could be used to reconstruct how individual voters have voted. In particular, the mixing service (and other system components) maintain a list of encrypted votes linked to each voter's identity; and at certain points in the election lifecycle the decryption keys for decrypting these votes briefly exist in non-volatile RAM on the mixing server[2]. A malicious individual who receives both of these pieces of information could recover how voters who voted on that voting system voted. Though the software does contain technical mechanisms that partially mitigate this risk, the software does not contain technical mechanisms that would prevent a malicious individual with sufficient foresight, technical sophistication, and access to the mixing service from obtaining this information[3]. Consequently, voters who wish to keep their vote a secret must rely upon the honesty of individuals with access to the mixing service. Fortunately, the system appears to make it difficult for even an authorized insider to compromise ballot secrecy after normal election operations have concluded without the cooperation of canvassing board members.

  We did not find clear procedures in the system documentation for ensuring that all copies of these electronic records will be securely erased after every election.

[2] Scytl has explained to us that the mixing service reconstructs the Electoral Board private key from the shares of the board members, uses it to decrypt votes, and then deletes its in-memory copy of this key. Assuming that this is implemented correctly, the time period during which the private key is stored in RAM should be short, reducing the risk accordingly. We are not aware of any provisions to prevent the private key from being stored in swap during this time. If that occurred, then the private key might be written to the hard disk and retained on the hard disk, possibly for a lengthy period of time.

[3] For instance, a malicious individual with authorized access to the mixing service might be able to introduce malware onto the mixing service machine. If the malicious individual introduces malware before the shares are entered into the machine, the malware could be programmed to retain a copy of the private key. The malicious individual might later be able to obtain a copy of the retained private key. This attack would require advance planning, technical knowledge, and ability to access the mixing service both before and after the private key is reconstructed.

- *Coercion and improper influence:* The Pnyx.core ODBP 1.0 system does appear to protect voters from coercion, pressure, improper influence, and vote-buying. As far as we can tell, voters cannot prove how they voted to third parties. Even if a voter wants to demonstrate how they voted, the voting system appears to prevent them from doing so, if poll workers and election officials follow proper procedures. Voters do receive a receipt after voting, but this receipt does not reveal how they voted; third parties without special insider access learn nothing about the voter's vote from the contents of the receipt. Because voters vote from a polling place staffed by poll workers, it is possible to ensure that voters vote in a environment where they are free of improper influence. This enables voters to vote their conscience in the privacy of the voting booth without fear of reprisal.

It is also instructive to compare the Pnyx.core ODBP 1.0 system to other voting systems that it might replace. The Pnyx.core ODBP 1.0 system is intended as an alternative for overseas voters who otherwise would normally vote by mail, or perhaps by fax or cleartext email.

We can compare the Pnyx.core ODBP 1.0 system to postal voting (vote-by-mail), according to the level of ballot secrecy these two systems provide:

- *Protection against outsiders:* The Pnyx.core ODBP 1.0 system seems to be roughly comparable to vote-by-mail, in the level of ballot secrecy provided against malicious outsiders, though there are some differences and the ODBP system has some advantages over postal voting. Because Scytl's Pnyx.core ODBP 1.0 system encrypts votes while they are in transit, it appears to be more secure than postal voting against attempts to eavesdrop on votes while they are returned back to county election officials for counting.

- *Protection against insiders:* None of the systems available to overseas voters provides strong protection against malicious insiders who exceed their authority. Instead, in all systems, ballot secrecy relies upon the integrity and honesty of election officials and other insiders.

  It is difficult to compare the ballot secrecy of the Pnyx.core ODBP 1.0 system vs. postal voting. Both systems rely upon procedural controls and the honesty of trusted election officials. Though it is possible there may be differences in the level of ballot secrecy provided by these two systems, it seems likely that any such differences (if they exist) would depend upon procedures and typical practice. As such, they are necessarily beyond the scope of a software review such as this one.

- *Protection against coercion:* The Pnyx.core ODBP 1.0 system provides markedly better protection against coercion than postal voting. Postal ballots provide little protection against coercion and improper influence, since a coercer can insist on watching as the voter marks and mails his/her ballot.

We can also compare the Pnyx.core ODBP 1.0 system to voting by fax, according to the level of ballot secrecy these two systems provide:

- *Protection against outsiders:* The Pnyx.core ODBP 1.0 system provides better protection against outsiders for ballot secrecy than voting by fax. Faxed ballots are not encrypted and thus are subject to interception while they are transit. Faxes may be transmitted over communications infrastructure that is controlled by unfriendly governments or subject to hacking, and thus the secrecy of faxed ballots cannot be assured.

- *Protection against insiders:* None of the systems available to overseas voters provides strong protection against malicious insiders who exceed their authority. Instead, in all systems, ballot secrecy relies upon the integrity and honesty of election officials and other insiders.

  The Pnyx.core ODBP 1.0 system seems to provide better protection against malicious insiders than voting by fax, for a simple reason: faxes are directly identified with the voter's identity, so it would be easy for election workers to unintentionally notice how a fax voter voted, even if they didn't want to know. In contrast, even though it would be possible for a malicious insider to learn how a voter voted in the Pnyx.core ODBP 1.0 system, this would require malicious intent and some degree of technical sophistication—insiders cannot accidentally notice how voters have voted.

- *Protection against coercion:* The reviewed system provides markedly better protection against coercion than voting by fax. Like postal voting, vote-by-fax provides little protection against coercion and improper influence, since a coercer can insist on watching as the voter marks and faxes his/her ballot.

Finally, we compare the reviewed system to voting by unencrypted email, according to the level of ballot secrecy these two systems provide:

- *Protection against outsiders:* The Pnyx.core ODBP 1.0 system provides significantly better protection against outsiders for ballot secrecy than voting by unencrypted email. Unencrypted email is subject to interception while it is in transit, since it is normally transmitted via the Internet, whose associated infrastructure and servers are subject to hacking. It may even be transmitted over communications infrastructure that is controlled by unfriendly governments. Thus the secrecy of unencrypted emailed ballots cannot be assured.

- *Protection against insiders:* None of the systems available to overseas voters provides strong protection against malicious insiders who exceed their authority. Instead, in all systems, ballot secrecy relies upon the integrity and honesty of election officials and other insiders.

  Like voting by fax, the Pnyx.core ODBP 1.0 system seems to provide better protection against malicious insiders than voting by unencrypted email: emails are directly identified with a voter's identity and so election workers who fail to avert their gaze might inadvertently notice how an email voter voted. In comparison, the Pnyx.core ODBP 1.0 voting system protects election workers from accidentally noticing how individual voters voted.

- *Protection against coercion:* The Pnyx.core ODBP 1.0 system provides significantly better protection against coercion than voting by unencrypted email. Email provides little protection against coercion and improper influence, since a coercer can insist on watching as the voter marks and emails his/her ballot. In addition, for unencrypted email ballots, it appears that it would be easy for a coercer to insist that the voter Cc: the coercer on the email containing the voted ballot, revealing the voter's vote; this attack is not possible against the Pnyx.core ODBP 1.0 system.

In short, the level of ballot secrecy provided by the Pnyx.core ODBP 1.0 system appears to be comparable to, if not significantly better than, the level of ballot secrecy provided by the voting methods it is intended to replace.

Finally, the secrecy of the ballot in the reviewed system relies upon the correctness of the software that is used during the election, especially the voting client and mixing service code. As explained below in depth, if that code were malicious, it could subvert ballot secrecy. Malicious software can be injected into applications such as the reviewed system by either compromising the source before it is delivered or by injecting malware into the code base after it is delivered, for example, during testing or during actual operation.

As explained in Section 4.3.3, we cannot rule out the possibility of malicious code in the certified source code, as it is beyond the state of the art to verify that any software is free of malicious code. At the same time, we have no evidence or reason to believe that the reviewed software contains malicious code that could compromise ballot secrecy. To prevent subsequent introduction of malware, it is important that we prevent individuals from introducing malicious code into the system, because such malicious code could have severe consequences that extend beyond ballot secrecy—malicious code could also tamper with votes.

It is not clear what motive an attacker might have to violate ballot secrecy by introducing malicious code: presumably, an attacker with the ability to introduce malicious code would be more likely to try to change votes directly, rather than violate ballot secrecy and then indirectly buy votes or coerce voters. As a result, this threat seems secondary, though it illustrates that it is important to have robust protections against the introduction of malicious code.

4.2.1.1 Finding: If the software used during the election contained malicious logic, ballot secrecy could be compromised

The secrecy of the ballot in Scytl's Pnyx.core ODBP 1.0 voting system relies upon the correctness of the Scytl software, especially the voting client and mixing service code. If that code were malicious, it could subvert ballot secrecy. We emphasize that this risk only applies if the code executed during the election is malicious, e.g., if Scytl's code contains malicious logic or if the code that is executed during the election differs from the legitimate, certified code written by Scytl.

For instance, though we have not verified this, it appears that malicious code on the voting client might be able to leak votes to an unauthorized third party over the Internet. In particular, the voting kiosk has a connection to the Internet, so it could leak information directly, by simply transmitting that information to a drop site elsewhere on the Internet. In principle one can imagine mechanisms that could detect or prevent this but we are not aware of any

such mechanism in the Pnyx.core ODBP 1.0 system. Similarly, the voting client can leak information indirectly using subliminal channels. We expand on the subliminal channel issue next. The ability to leak confidential information about voters' votes seems to be a general issue with any voting architecture that uses a network-connected voting client.

4.2.1.2 Finding: If the voting client software contained malicious code, it could use voter receipts as a subliminal channel to make the receipts salable

In the reviewed voting system, each voter receipt contains a *ballot identifier*, a number chosen by the voter client that is supposed to be random (unique and unpredictable). However, a corrupted voting client could exploit this ballot identifier as a subliminal channel. Instead of generating a random number, the corrupted client could generate a number that encodes the voter's identity and choice (e.g., using a hash function). The voter could then sell this receipt to the adversary who corrupted the voting client.[4]

In general, any architecture where the software chooses random numbers that are included in a public channel is likely to share a similar property: If the client were malicious, then it could leak information that violates the secrecy of the ballot. We foresee no mitigation of this architectural property without significant redesign of the Scytl protocols.

Nonetheless, exploiting this vulnerability requires replacing the Pnyx.core ODBP 1.0 voting client software with malicious software. Such malicious software could mount far more damaging attacks on the integrity of an election. Thus, we do not consider the subliminal channel to be a serious risk in Scytl's voting system.

Action: Election officials must prevent the introduction of malicious code into the system. They must ensure that deployed voting clients use the certified code on election day.

4.2.1.3 Finding: Voter choice records and encrypted ballots contain a component whose implications for ballot secrecy are unclear

A *voter choice record* (VCR) is a piece of paper printed by the voting laptop. The VCR is supposed to be a copy of the voter's electronic vote. This piece of paper also contains a *session id*, which is supposed to be a random (unique and unpredictable) number. The cryptographic purpose of the session id is not clear to us, nor is its presence clearly documented in the materials given to us by Scytl.

If the voting client were corrupted, then this session id could be chosen maliciously by the corrupted voting client (just as discussed above in Finding 4.2.1.2 with respect to the subliminal channel in ballot ids). This would enable an agent with access to the VCRs generated by that corrupted voting client to infer from the voter choice record how each voter voted. As above, we do not consider this a serious risk.

However, the session id introduces another potential vulnerability. Each session id is associated with a particular voter, and the session id appears on the same piece of paper that records (in plaintext) that voter's vote. This raises the question of whether the session id could be used to violate ballot secrecy. We emphasize that we have not constructed an attack that would do so, and that such an attack would require either a malicious voting client or an error in Scytl's client code.

The review team experienced some confusion at the last minute, when Scytl informed the team that the session id is also included in the encrypted vote sent on the network. (See §3.3.7.) This was not documented in the protocol specification and was omitted in earlier communications from Scytl about the protocol. Because this use of the session id was discovered at the last minute, the review team was not able to evaluate its effect on the cryptographic protocol or its security implications.

Action: Scytl should document the cryptographic purpose of the session id. Future reviews should investigate the purpose and security implications of the session id.

[4] As we noted earlier, the voting client is a trusted component. Thus, we know that malicious software could cause a variety of security problems for the voting system. For example, a corrupted voting client could violate voters' privacy by sending a network message to the adversary after the voter votes. However, note that the subliminal channel enables the corrupted voting client to behave in every observable way as an honest client: The corrupted client need not send any extra messages, write any extra values to storage, or exhibit any other behavior that would reveal it to be corrupted. The corrupted client simply chooses one number, which was meant to be random, in a way that to a human will still appear random. Thus, it would be impossible for an audit of the client's behavior to reveal whether this attack has occurred.

4.2.1.4 Finding: The 'results' report from the Mixing Service could potentially enable coercion, vote-buying, and improper influence, if released to untrusted individuals

The Mixing Service contains functionality to output a 'results' report. It appears that this report could endanger voter anonymity and the secrecy of the ballot, if it were revealed to untrusted individuals. This report contains a list of all of the cast vote records, one per voter. Each cast vote record contains the list of all selections made by that voter. These are not separated by contest.

This could potentially allow a voter to "mark" their ballot in a way that allows the voter to prove how they voted. A simple example would be for the voter to vote a write-in vote for themselves in some inconsequential contest; this would enable them to pick out their own ballot from the 'results' report file and thus to prove to others with access to the report file how they voted.

As another example, suppose that a vote-buyer or coercer knew in advance that he would be able to obtain access to the 'results' report. The vote-buyer might pick a random, unique, unpredictable code and instruct the voter to enter that code as a write-in in an inconsequential contest and might instruct the voter how to vote for the rest of the contests. Once the vote-buyer obtains access to the 'results' report after the election, the vote-buyer will be able to determine whether there was any ballot cast using this unique code, and if so, whether the rest of the votes were cast as the vote-buyer instructed. If the voter followed instructions, then the vote-buyer could pay the voter whatever price was agreed upon in advance. If the voter failed to follow instructions, the voter-buyer could take revenge. A similar attack could be mounted without relying upon write-in votes, by using so-called "pattern voting." If the vote-buyer uses a different code for each voter, then the voter-buyer will be able to identify which voters have cooperated and which ones have not.

The take-away point is that any individual with access to the 'results' file may have the ability to exert improper influence over voters: a voter who colludes with such an individual can prove how they voted to that individual. As a result, we suggest that only trusted individuals—e.g., election officials—should be allowed to have access to the 'results' file. Election officials should exert control over who is allowed access to the 'results' file: the 'results' file should not routinely be released to the public, to the candidates, or other interested individuals.

We note that this functionality is not unusual: many other voting systems also provide a similar reporting functionality. As such, this issue is not unique to the Pnyx.core ODBP 1.0 voting system, and we expect that procedures used to protect other voting systems can also be applied to the Pnyx.core ODBP 1.0 system as well.

Action: Election officials should control access to the 'results' report. It should not be routinely released to others.

Action: Scytl should document the issues surrounding the 'results' report so that election officials are aware of these issues and can make informed decisions. The use procedures for the voting system should describe under what conditions the 'results' report should be shared with others.

4.2.1.5 Finding: Under some conditions, the 'results' report from the Mixing Service could potentially violate ballot secrecy and enable reconstruction of how individual voters have voted, against their knowledge

When the Mixing Service produces the 'results' report, it shuffles the order of the cast vote records in a pseudorandom fashion. However, this shuffling is imperfect. In particular, the Mixing Service groups the cast vote records (in the order they are received) into batches of 1000 votes. Then, for each batch, it shuffles all votes within that batch[5]. However there is no shuffling between batches, so after shuffling a vote remains in the same batch it was in originally.

Scytl informed us that the batch size was chosen so that, if the number of voters in the November 2008 pilot is as expected, there will be only a single batch. In this case the batching issue will not affect the 2008 pilot. On the other hand, if more than 2000 voters vote on the ODBP system, then the batching could potentially have implications for ballot secrecy.

In addition, the Mixing Service keeps each cast vote record (with all the votes cast by a single voter) together throughout the process; contests are not shuffled independently. In many elections, it is not uncommon for there to be many ballot types, where each ballot type contains a different list of contests on the ballot, and voters in different districts or different precincts receive different ballot types. Note that the cast vote record identifies the ballot type provided to that voter, because the cast vote record reveals the list of contests that the voter was permitted to vote on.

[5] Scytl informed us that, if the total number of votes is not a multiple of 1000, the leftover votes are included in the last batch, so the last batch may contain from 1000 to 1999 votes. For instance, if there are at most 1999 votes cast, then they will all be included in a single batch.

In some situations these two properties could potentially have negative consequences for voter anonymity: individuals with access to the 'results' report and other information could potentially reconstruct how some voters voted. We give an illustrative example. Suppose that at least 2000 voters vote on the Pnyx.core ODBP 1.0 voting system, and Alice is one of the first 1000 voters to vote in this way; then we know that her cast vote record will be found in the first batch. Suppose there are many ballot types, and we know which ballot type is assigned to Alice; this would normally be inferrable from public information, such as Alice's voter registration information and the list of ballot types. Suppose that among all the first 1000 voters, Alice is the only voter with this ballot type—none of the other 999 voters is assigned the same ballot type that she was. Suppose that the adversary has access to the 'results' report output by the Mixing Service. In this case, the adversary can examine the first batch of 1000 cast vote records in this 'results' report, find the unique cast vote record whose ballot type matches the ballot type assigned to Alice, and then the adversary will have identified Alice's votes. In certain circumstances, this could enable an adversary with access to the 'results' report to identify how specific voters voted.

This is only one example scenario, and it is not intended to be exhaustive. There are many variations on this basic attack. For instance, if we know that Alice is one of exactly 3 voters among her batch who were all provided a particular ballot type, then we can narrow down Alice's ballot to one of 3 possibilities. If we additionally know (through some independent means) who Alice was likely to vote for in one or two races, we might be able to identify which of those 3 is likely to have been Alice's cast vote record and thus identify how Alice voted in the remaining races.

The risk of this attack depends upon many factors. In general, the risk increases as the number of ballot types increases. For example, primary elections may be especially at risk, if each voter's ballot type depends not only upon where they live but also on their registered party affiliation. The risk may be significantly mitigated if there are strong controls to prevent untrusted individuals from gaining access to the 'results' report.

This issue may be of only minor relevance to the November 2008 election, because that election will involve only overseas voters registered within a single county. As a result, the number of different ballot types is limited to the number of ballot types in use within that county. However, if use of the Pnyx.core ODBP 1.0 system became more widespread, this issue could become more relevant.

Action: Scytl should repair this shortcoming of the software: rather than shuffling only within batches, the Mixing Service should perform a single shuffle of all votes.

Action: Until Scytl fixes this shortcoming, to protect the secrecy of the ballot, election officials should control access to the 'results' report. It may be appropriate to restrict access so that this report is not made public or shared with untrusted individuals. The level of protection necessary is inversely proportional to the average number of overseas voters per ballot type.

Technical details: We are not aware of any reason why it would be necessary to shuffle votes in batches of 1000. We do not see any barrier to performing a single shuffle of all the votes. If there are n votes cast in total, then in principle all $n!$ permutations should be possible—the shuffling algorithm should not rule any of these out (though of course the pseudorandom number generator is likely to restrict this number). There are well-known algorithms in the computer science literature for performing efficient and scalable shuffles: for instance, one can shuffle n votes in $O(n \log n)$ time using standard techniques. These algorithms should easily scale to shuffling of millions or tens of millions of votes, which is far more than are expected to be handled by the Pnyx.core ODBP 1.0 voting system in the foreseeable future. This improvement to the software would significantly mitigate the issue.

4.2.1.6 Finding: Data retention issues are in tension with ballot secrecy

In the Pnyx.core ODBP 1.0 system, the electronic records that are preserved after the election contain enough information to trace votes back to the voter who cast those votes. In particular, given the encrypted votes and the decryption key used by the mixing service, it is possible to link the identity of each voter to how they have voted. This poses several challenges for security:

1. It is envisioned that at least one copy of these records would be retained for at least 22 months. This means that if there is an extended window of vulnerability, where if the records are disclosed at any point during those 22 months, voter anonymity could be compromised. Of course, the longer the window of the vulnerability, the greater the risk of inadvertent disclosure. It also means that it would be technically possible to reconstruct how voters have voted long after the election occurred, for instance in the event of a court investigation.

We suggest that it might be worthwhile to examine whether it is possible to securely erase part of this information—e.g., the private keys—before 22 months is up, or whether there is some way to permanently and irreversibly anonymize the information that is stored.

2. The retained copy must be securely erased after 22 months have past. We understand that procedures are being developed for securely erasing this information, and it is envisioned that this process will involve use of the DBAN software. The DBAN software appears to be a very good choice for this purpose.

3. All other copies of this information must be erased, preferably as soon as possible. One challenge with electronic records is that copies can proliferate. During the routine operation of the election, copies are stored on the voting proxy and voting service computers, which are operated by Scytl and located in the Scytl data center. In addition, copies are stored on the mixing service computer. If system operators use removable media to transport these records, a copy of this information may be retained on that media. And any backups of these systems may cause further copies of the information to be made. It will be necessary to develop a procedure to identify all copies of this information and securely erase all such copies. It will be necessary to devise some way to provide positive control over the information at the Scytl data center and enable election officials to verify that all copies have been deleted. And it may be necessary to provide a way for election observers to confirm for themselves that the data erasure procedures have been followed.

Action: Data retention issues need to be worked out. Procedures need to be developed to ensure that all copies of this information are erased as soon as possible, and to ensure positive control over the retained copy of the data.

4.2.1.7 Finding: The ballot secrecy property (from the scientific literature) provided by the Pnyx.core ODBP 1.0 voting system is receipt freeness

In the scientific voting literature, *receipt freeness* [3] means that the process of voting does not permit the voter to acquire any piece of information that could later be used to convince an adversary of how the voter voted. Thus, the voter has no receipt that could be used, e.g., to sell his vote. We believe that Scytl's Pnyx.core ODBP 1.0 voting system provides receipt freeness (although we have no mathematical proof of this claim, nor has Scytl provided one).

We note that the cryptographic voting literature defines another property called *coercion resistance* [13]. This has a specific technical meaning in the context of Internet-based voting schemes, in which voters may vote from anywhere. But since the ODBP allows voters to vote only from supervised polling places, coercion resistance is not relevant to ODBP. It would appear from Scytl's documentation that they have envisioned implementing additional defenses against coercion, perhaps designed to provide this kind of coercion resistance. However, this defense is not implemented in the software being used in ODBP [25, p. 15].

Nonetheless, the reviewed voting system does provide substantial and effective defense against voter coercion. The Pnyx.core ODBP 1.0 system (like most polling-place based voting systems) relies upon procedural protections—poll workers are entrusted to prevent improper influence and ensure that each voter is able to vote in secret. By isolating voters while they cast their votes, most of the threat of coercion is mitigated by the Pnyx.core ODBP 1.0 system.

Action: Scytl should consider constructing a mathematical proof that its voting scheme satisfies receipt freeness; this would yield increased assurance in the scheme.

4.2.2 To what extent does the software architecture, implementation, and documentation address the threats which the target software might face?

We focus first on the integrity of the election. Several classes of threats to the integrity of the election were identified; we consider each threat class individually.

- *Insiders:* The software architecture alone does not address many insider threats.

  By insiders, we refer mean anyone with legitimate access to the voting system other than the voters and poll workers. This includes state and county election officials (e.g., the canvassing board), Scytl employees, and system administrators of the various computer systems used. There are various ways in which these insiders could manipulate the results of the election. In many cases these attacks require only one corrupt insider, while in other cases it may require some collaborative effort.

- *Poll workers:* The software architecture partially addresses the threat of malicious poll workers.

  We identify some ways in which a corrupt poll worker could potentially manipulate the votes cast in that polling location, possibly without the knowledge or cooperation of other poll worker(s). However, in all of the attack scenarios we identified, the damage that a malicious poll worker could do appears to be limited to the one polling location that the poll worker is entrusted with overseeing, so the effect of such attacks is limited.

- *Outsiders:* The software architecture partially addresses the outsider threat.

  For the purposes of this review, outsiders were considered to be anyone with no legitimate access to the voting system. The primary example of such an attacker is an Internet connected "hacker" whose only ability to interact with the voting system is across the network. To a large extent, this threat is mitigated by technical controls, such as firewalls on the voting client and voting server and the use of a VPN and SSL connections for network communication. However, as discussed in section 4.1.2, the risk of compromise by an outsider is (probably unnecessarily) increased as a result of the requirement that the voting systems be network connected for the entire duration of the voting period.

- *Malicious software:* The software architecture partially addresses the threat of untrusted or malicious client software.

  The software architecture and election procedures address the untrusted client threat to some extent by placing the voting client hardware under the control of election officials during the voting period. In addition, the software does verify that the Java application downloaded and executed by the voting client was signed by a trusted cryptographic key, and there is a process to ensure that the Live CD used to boot the voting client is also trusted through the use of a cryptographic hash function.

  However, as discussed in finding 4.2.2.3 the voting client relies on code which is downloaded across the network, and this code is not stored in persistent storage on the client (as it has none), which introduces the risk of introduction of malicious client software. While this would likely require at least one corrupt insider [6] to place and correctly sign malicious Java code on the voting server, this should also be considered to be relevant to the untrusted client threat, as there does not seem to be a mechanism by which the voting client can be audited to determine what code it executed.

  The software architecture only partially addresses the threat of malicious software running on trusted system components, such as the voting server and the mixing service. It partially addresses these threats, for instance by running the mixing service on an air-gapped machine that is not connected to any network. It does not completely address these threats, because malicious software on the mixing service could tamper with the unofficial results of the election.

The Voter Choice Records (VCR) provide a defense against many or all of these attacks, assuming that sufficient controls exist in the VCR chain of custody, and depending on the audit mechanisms employed, and assuming that voters carefully check their VCR for accuracy. This is not to suggest that other procedures should not be carefully designed and implemented throughout the election system as well, but it is the opinion of the review team that the procedures surrounding the use of the VCRs are vital to protecting the integrity of the election, even if all other procedures are carefully designed and followed. Designing and implementing procedures necessary to provide protection for paper records (the VCRs in this case) in an election system is a problem that is more clearly understood, and for which election administrators have had significantly more experience, than is the case for procedures related to complex, distributed software and hardware based electronic systems. In addition, the VCRs are verified by voters and thereafter not affected by electronic manipulation or electronic failures.

[6] While it seems reasonable to assume that one corrupt insider would be required to perform this attack, it is not clear that collusion between 2 or more insiders would be required. For example, if a single corrupt insider at FLDoE signed malicious code, they may be able to convince Okaloosa County or Scytl staff to load that signed code onto the servers as part of an election setup process or through some social engineering effort, thus achieving the goal of introducing malicious code into the Voting Clients without the voting server administrators or the poll workers knowingly cooperating. In addition, as FLDoE staff are involved in the initial installation and configuration of the voting servers, it may be possible to introduce the signed application during that process, or alternatively to introduce malware at that stage which allows subsequent access to the servers without Okaloosa or Scytl administrator involvement. We were unable in the time provided to perform a complete analysis to determine which individuals or groups could perform such an attack, but in the limited analysis that was performed it was not clear that collusion between 2 or more parties was required.

The bottom line is that the ability of the Pnyx.core ODBP 1.0 voting system to defend against threats to its integrity seems to be heavily reliant upon how the Voter Choice Records are handled and audited. However, the VCR handling procedures were not available to the review team during the review.

To place Scytl's Pnyx.core ODBP 1.0 remote voting system in context, it may be informative to compare the Scytl system to other voting systems that it might replace, such as voting by mail, by fax, or by unencrypted email. We can compare how effectively these systems addresses threats to integrity of the election.

- *Insiders:* It appears that all these systems are vulnerable to insiders.

  One potential advantage of the Pnyx.core ODBP 1.0 system and postal voting over voting by fax or unencrypted email is that the former two systems retain a voter-verified paper record of the voter's vote that can be used for auditing election results. This may help better protection against, for instance, corrupt vendor employees or certain other insiders, if those records are audited appropriately, if sufficient controls exist to protect the chain of custody of these records, and (in the case of the Pnyx.core ODBP 1.0 system) if voters check these records carefully for accuracy.

- *Outsiders:* The Pnyx.core ODBP 1.0 system appears to provide better protection against outsiders than voting by fax or unencrypted email, because it does not transmit votes in unencrypted form over unsecured communications channels.

  Comparing the Pnyx.core ODBP 1.0 system vs. postal voting in the level of protection they provide against outsiders is more difficult. Assuming that poll workers are honest and follow procedures and that election procedures are well-designed, the Pnyx.core ODBP 1.0 system may provide better protection for the integrity of the election, because voters can be authenticated in person by trusted poll workers (hence to conduct vote fraud an attacker would need to show up in person), because ballots are not subject to tampering in the mail, and because election officials retain both an electronic and a paper copy of the ballot.

Overall, the Pnyx.core ODBP 1.0 system may be at somewhat greater risk of high-tech attacks that require considerable sophistication but affect many votes, whereas it may be more secure against low-tech attacks that require no sophisticated technical knowledge but affect relatively few votes. We emphasize that these comparisons are approximate and are predicated upon what we know about the Pnyx.core ODBP 1.0 system at this point in time.

Some specific findings related to these issues are given below.

We also examined the security of the Pnyx.core ODBP 1.0 system against threats to its availability, e.g., denial-of-service attacks that may prevent voters from voting. We identified some potential areas of concern that leave us uncertain whether the architecture adequately addresses the threat of denial-of-service threat. We were not able to confirm or refute these concerns definitively. We refer the reader to Finding 4.2.2.4.

Threats to ballot secrecy are analyzed in section 4.2.1 and are not considered further here. Because threats of voter coercion and improper influence can also affect the integrity of the election, we repeat our conclusion that the software architecture does address the voter coercion threat. The use of poll-worker controlled polling places in this system addresses voting coercion far more effectively than the paper-based absentee ballot system it was intended to replace.

4.2.2.1 Finding: If poll workers are corrupt, then the software does not, on its own, prevent them from casting votes on behalf of voters

The Pnyx.core ODBP 1.0 voting system is advertised to provide: "Authenticity of ballots...[a] reliable means to verify the...identity of the voter that casts it" [26, §1].

It was not clear to us whether the Pnyx.core ODBP 1.0 system achieves this goal. Depending upon the procedures in use, it might be possible for polling place workers to cast votes that appear instead to have been cast by registered voters—the software does not appear, on its own, to prevent this kind of attack. Polling place workers have the ability to retrieve any voter's username, password, and assigned ballot style from the authentication laptop and transfer it to the smartcard. A smartcard loaded with such information is sufficient to allow anyone who possesses it to vote from the Voting Client. Thus, a malicious pollworker could conceivably vote in place of a registered voter. We did not find any detailed analysis of these kinds of attacks in the voting system documentation.

We emphasize that the possibility for polling place workers to vote on behalf of voters is a threat in any voting system in which pollworkers are trusted to authenticate voters, not just Scytl's system. Nonetheless, the attack identified above demonstrates that Scytl's claim regarding this threat appears to be too strong.

The defenses employed by Scytl against this threat include:

- Only one vote is accepted per voter: if a voter were to try to vote after a polling place worker voted in his place, then the voter would receive an error that she had already voted.
- Dual control: there will be two workers administering each polling place.

The first defense is reasonable only until the end of the voting period, as polling place workers could wait until shortly before the end of the election and then cast votes on behalf of all voters who never appeared during the voting period. Additionally, corrupt poll workers could use their own software to carry out this attack, downloading all user names and passwords in advance and casting ballots shortly before the close of polls.

We do not know whether such an attack would be feasible in practice. We were not able to perform a comprehensive assessment of whether other procedural defenses would be likely to detect this kind of attack. For example, it is possible that the voter's (hand-written) signature on the Voter Certificate might provide an effective defense against ballot box stuffing by corrupt poll workers, but we did not evaluate this possibility in depth. This would depend upon procedures surrounding how these signatures are checked after the election, which was beyond the scope of this review. In addition, it would depend upon details about the voter registration system, such as whether the voter registration system allows poll workers to view electronic scanned images of voter signatures, and the voter registration system was beyond the scope of this review.

The second defense is unfortunately weakened in the ODBP project by another requirement, which we understand was made by the state of Florida: The voter's smartcard must be inserted into the voting laptop by an official before the voter votes; then, the card must be removed from the voting laptop by the official after the voter votes. Thus, twice for each voter, the authentication laptop is not under dual control.

Action: Users of this system should consider whether it would be beneficial and cost-effective to add a third polling place worker to each site. This would enable continuous dual control of the authentication laptop.

4.2.2.2 Finding: Voter credentials are not adequately protected against malicious individuals with access to the authentication laptop or malicious software running on this laptop

Each voter's *credentials* consist of a login (user id) and password. These are created during the initialization phase of an election, encrypted, and stored on the credential provisioning service. A polling place worker downloads a voter's encrypted credentials to the authentication laptop, which then decrypts them and writes them to a smartcard.

When the authentication laptop decrypts the credentials, they are briefly available in plaintext on the authentication laptop. With these plaintext credentials, anyone or any machine with access to the voting service could cast a vote on behalf of that voter.

Thus, if malicious software were introduced to the authentication laptop, then the software could obtain all voters' plaintext credentials. If this software was then able to access the secret key material necessary to connect to the voting service VPN (which is different from the secret key material on the authentication laptop which only enables communication with the Credential Provision Service), or transfer the credentials to another machine that already had access to the voting server, then it could submit votes on behalf of all voters. The voting service has been written to only accept one vote per voter; nonetheless, malicious software could prepare votes for all voters, and then submit them at the very last minute of the election. The system would then record votes for those individuals who had not voted.

Again, we do not know whether such an attack would be feasible in practice. It has been claimed that system audit logs record all transactions and their times. Thus, if system logs are reviewed, it is possible that unusual voting patterns such as a large number of ballots cast at the close of the period could be detected and trigger an investigation.

Action: Scytl should document this issue. Scytl should reconsider performing decryption of credentials on the authentication laptop and examine other alternatives, such as decrypting on a smartcard or on an air-gapped machine.

4.2.2.3 Finding: Voting Client laptop software can be modified

The Voting Client laptop does not contain a hard disk drive, but is instead booted from a "Live CD", which has been prepared by Scytl for the ODBP. While the contents of this Live CD were provided to the team during this review, it

was not possible to boot a machine using the Live CD because of hardware incompatibilities. In particular, the Live CD had been modified to only include drivers necessary to boot the particular laptop/printer/touchscreen combination to be used in the ODBP, and we did not have access to this hardware.

Action: In future reviews, any specialized hardware required to execute any component of the system should be made available to the review team.

The Live CD is produced by Scytl, and distributed with an SHA-1 hash which can be used to verify that the CD used is the one distributed for the ODBP. However, some problems exist with using the hash in this way:

1. The Live CD production point (Scytl) is a single point of failure. Malicious code injected at that point, prior to distribution, would not be detectable using the SHA-1 hash (as the hash would be calculated on the modified CD).

2. Someone with access to the voting kiosk laptop (such as a poll worker) could replace the real Live CD with a modified version which contained malicious logic. In all respects this modified Live CD could perform all of the election functionality without detection (as all of the required "secrets" are either stored on the Live CD itself, or provided to the Voting Client laptop during the election process).

   Such a modification could be detected by manually calculating a hash of the Live CD contents on a "trusted" laptop. However, such a switch might not be detected if the Live CD was replaced, then reinserted prior to the check. It is also plausible that the CD could be replaced at run-time without requiring a reboot, although it was not possible to confirm or refute this during the test.

   Calculating the hash on the voting laptop itself, using the Live CD software to perform the calculation, provides a weaker defense than calculating the hash on a distinct trusted laptop. If calculated on the voting laptop, then the hash could detect an erroneous software version introduced inadvertently, an attack that does not attempt to defeat the hash, or an attack that makes a mistake in defeating the hash. However, the hash mechanism would not successfully defend against attacks by malicious insiders who can subvert the hash. (c.f. footnote 7 on page 45.)

   It should be noted that this method of modifying the voting machine software is likely to be substantially easier than modifications to common DREs or even manual voting machines.

   Action: Ensure that very clear procedures are in place to allow poll workers to verify the hash value of the Live CD using a trusted laptop prior to any significant operations, such as the start or end of the election, or the reboot of the Voting Client for any reason.

   Action: Seal the Live CD in the CD drive during the election using tamper-evident seals.

3. Third, the Live CD is used to boot the Voting kiosk laptop, at which point it retrieves the Voting Client Software (a Java application) from the Voting Server. It is this Java application which actually communicates with the Voting Server: the Java application records the votes independently from any software on the Live CD. The Voting kiosk accepts the Java application provided it has been signed by an appropriate cryptographic key. As the Voting kiosk laptop has no persistent storage, this Java application resides solely in memory. Assuming that someone with knowledge of the cryptographic key signed a malicious version of the Java application and supplied it to the Voting kiosk (either through a man-in-the-middle attack against the VPN connection, or by placing it on the Voting Server itself), the Voting kiosk would execute the malicious Java application. While carrying out this attack might not be trivial, the most problematic issue is that it does not seem to be possible to verify (e.g., at the end of the election, or even during the election) what software was actually executed on the Voting Client—any suitably signed Java application will be executed, and no record of that software is kept on the diskless Voting Client for later verification.

   Action: Include the Java Application on the Live CD and use that static version to run the election, rather than relying on code downloaded over the network.

4.2.2.4 Finding: We do not know whether the voting system adequately protects against denial-of-service attacks

We were not able to determine whether the voting system adequately defends against denial-of-service attacks, though we were able to determine some relevant information. We suspect that it would be possible, given sufficient resources, to mount a denial-of-service attack against the Scytl servers. However, we are not certain about what impact this would or would not have. In particular, it is not clear to us whether it is possible to vote on the voting kiosks if the voting proxy server is unreachable or not responding. It is also not clear to us whether there will be a fallback procedure that will allow voters to continue voting (e.g., on paper ballots) in case the kiosks are not available, or whether voters would be turned away in that case. Consequently, it is difficult to ascertain how well the system protects against denial-of-service attacks.

One important mitigation against denial-of-service attacks is that the overseas polling locations are expected to be open and available to voters over a period of ten days. This means that an attacker would likely need to sustain the denial-of-service attack over close to that entire time period to have a significant impact. The impact of short-lived denial-of-service attacks will be proportionately reduced, and in some cases might be mitigated even further if voters can return another day. A second mitigation against denial-of-service attacks, at least for the November 2008 election, is that election officials only expect hundreds or thousands of overseas voters to vote using the Pnyx.core ODBP 1.0 voting system. This reduces the motivation for would-be attackers to bother attacking the system.

One possible threat is a distributed denial-of-service (DDoS) attack against the voting proxy servers. For instance, an adversary might use a botnet (a collection of compromised hosts) to send a large quantity of traffic towards the voting proxy in hopes of overloading the data center's network link and rendering the voting proxy unreachable. Scytl shared with us information about their data center and the capacity (bandwidth) of the bottleneck link from the data center to the Internet. We decline to share the specific number here, to avoid providing information that might aid an attacker.

However, we used this information to compute a rough estimate of how much it might cost an attacker to mount a DDoS attack. Researchers who have studied the cybercrime underground report that attackers can purchase bots (compromised hosts) at an estimated cost of approximately $5–25 per host [10]. We estimated that each compromised host has a 100 Kbps upload link (which would be representative of a typical consumer-grade DSL or broadband link). We estimated that each compromised host could be used for several days; this amounts to an assumption that, if the hosts send traffic with a spoofed IP source address, it would take this long for defenders to trace down and block that host. We also assumed that only a fraction of compromised hosts would be capable of sending traffic with a spoofed source IP address. Based on these assumptions, we estimated that a budget of $5,000 might be sufficient to buy enough compromised hosts to mount a DDoS attack that would last for a week and overwhelm the access link provisioned for the Pnyx.core ODBP 1.0 voting system. We warn that this is a rough estimate, and this figure could easily be off by a factor of 2–4$\times$ or more. We have not attempted to actually perform such an attack, so these estimates must be treated as unconfirmed, back-of-the-envelope guesstimates.

We note that there are other challenges an attacker might face. Most importantly, the attacker would need to identify the IP address of the voting proxy, or at least identify the range of addresses used by the data center. While this would be easy to do given insider access—for instance, a corrupt poll worker with access to the Live CD could easily obtain this information, and a corrupt voter might be able to obtain this information—this information might be hard to obtain for an attacker on the other side of the world, with no access to the voting system. In addition, Scytl has developed a procedure for changing this IP address in case of attack, and this procedure might force a would-be attacker to learn this information several times. We are generally reluctant to rely heavily upon the secrecy of the IP address of critical server, because that information may be difficult to keep secret. Nonetheless, in practice it would appear that this is a significant hurdle that an attacker would face, and this hurdle might greatly reduce the population of attackers with the capability to mount such an attack. This could significantly reduce the likelihood of a denial-of-service attack.

Our analysis is predicated upon the assumption that the election will involve only a few polling places, as we expect to be the case in November 2008. If the number of polling places were significantly increased in future deployments, the effectiveness of this countermeasure could degrade significantly. In addition, it is not clear whether Scytl's procedure for changing the voting proxy IP address will realistically scale to a large number of polling places.

Another risk is the possibility of denial-of-service attacks against the clients (i.e., the polling places) rather than against the servers. The polling place are likely to have Internet connections with only limited bandwidth, providing little resilience against DDoS attacks. Because the locations of the polling locations are public, it may be easier for an

attacker to determine the IP address ranges used by polling sites than to determine the IP address of the data center. It may also be possible to attack polling places in certain locations in an attempt to mount a denial-of-service attack, where certain demographic populations of overseas voters are specially targeted. For instance, if military voters deployed to combat zones tend to vote differently from civilians living in a major first-world city, then an attacker might be able to selectively target one of these polling sites to swing the vote in a particular direction. Unfortunately, we were not able to evaluate the risk of denial-of-service attacks against clients in any depth, so the magnitude of this risk remains unknown.

We did not study in any detail whether it may be possible to mount application-level denial-of-service attacks. In these attacks, instead of flooding the victim with a huge amount of traffic and overwhelming the victim by simple brute force, the attacker sends carefully crafted packets intended to cause the victim's CPU to become overloaded with useless computation (even if the network has spare capacity). This kind of attack requires more technical sophistication, and it is correspondingly more difficult to determine whether software systems contain vulnerabilities that could enable such an attack. However, see section 4.3.9.2 for further discussion of this issue.

Ultimately, denial-of-service attacks are very difficult to address completely. It is partially an issue of economics: given a sufficient budget, an attacker can create an enormous amount of traffic and overwhelm even the largest site. One of the challenges is that the relative prevalence of compromised hosts on the Internet makes a DDoS attack of major magnitude relatively affordable for cybercriminals, while buying an Internet connection of sufficient capacity to withstand such an attack is expensive.

For these reasons, we feel that the most effective countermeasure against denial-of-service attacks may be to ensure that there is a fallback procedure so that voters can vote even if the voting servers are unreachable, the network is down, or the e-voting system does not appear to be working. For instance, officials might consider providing a supply of paper ballots that overseas voters could use as an emergency fallback if the Pnyx.core ODBP 1.0 voting system is not available. Our impression is that, considering the difficulty of this problem, Scytl is taking reasonable technical steps that partially address the risk of denial-of-service attacks against the servers, but technical countermeasures have limitations. The most effective defense may be one based upon non-technical measures, such as fallback procedures.

Action: Election officials should give serious consideration to developing procedures that enable voters to vote in the face of extended network or application failure.

4.2.2.5 Finding: In some places, the system documentation fails to clearly and accurately represent the properties provided by voter receipts and other auditing mechanisms

The Pnyx.core ODBP 1.0 system provides voters with receipts that they can use after the election to check up on their votes. The documentation provided to us discusses the purpose and nature of these receipts in multiple places. In several of these places, we noted that the documentation describes these receipts in ways that we feel fails to accurately disclose the properties actually provided by these receipts.

- The "Auditing Pnyx.Core" document states, on p.4 Section 2.4, that "the list of ballot identifiers can be published to allow voters to check the accuracy of an election."

  However, receipts do not provide the advertised assurances. For instance, there is no way for voters to check that votes were recorded in a way that accurately reflects the voter's intent, no way for voters to check that votes were not modified while in transit, and no way for voters to check that votes were counted accurately. At most, the receipts provide a way for voters to check that their ballot was not omitted entirely from the final tally. However, even this check is contingent upon the proper and correct operation of the Mixing Service and of the election officials, and receipts provide no way to check this assumption. See also finding 4.2.3.1 for further discussion.

  Consequently, we consider that this description may tend to provide an inaccurate impression of the nature of the receipts in the Pnyx.core ODBP 1.0 voting system.

- The "Auditing Pnyx.Core" document states, on p.3 Section 2.2, that the Scytl system allows "verification of the accuracy of the election process by means of a Voter Receipt [...] issued at the end of the voting process."

  As described above, the receipts do not provide the advertised assurances. Consequently, we consider that this description provides a misleading description of the level of verification afforded by voter receipts.

- The "Auditing Pnyx.Core" document also states, on p.4 Section 2.4, that "The file with the clear text votes [that is output by the Mixing Service] can be used for parallel recounts." This statement refers to a 'results' report

that can be requested by election officials and that contains a dump of the electronic cast vote records (see also Finding 4.2.1.4).

Unfortunately, this does not provide a useful way to perform a meaningful recount. While it is possible to re-tabulate the election using the 'results' file, this is not a meaningful or independent recount; it is simply a re-tabulation of the election results from the same internal electronic records that were used for the initial tabulation. Of course, if there are any errors or omissions in the 'results' file, then re-tabulations will produce consistently incorrect vote tallies. The only way to perform a meaningful recount is to perform a manual recount of the Voter Choice Records.

Consequently, we consider that this description could regrettably provide readers with a mistaken impression of the auditing properties afforded by the 'results' file.

In each case, the lack of clarity in the documentation could potentially cause election officials and other readers to misunderstand how receipts work and wrongly conclude that they provide greater protection than they actually do.

Action: Scytl should re-write the system documentation to avoid misleading or unclear claims. Scytl should add a clearer explanation of the auditability and verifiability properties that are and are not provided by the system, by voter receipts, and by the 'results' file.

4.2.2.6 Finding: There have been two previous independent security reviews of related Scytl software

The Pnyx.core ODBP 1.0 voting system incorporates and builds upon the Pnyx.core libraries. We understand that there have been two prior independent security reviews of the Scytl Pnyx.core libraries. These two reviews were performed by:

1. The Canton of Neuchatel, a jurisdiction in Switzerland, performed an internal security audit of the Scytl Pnyx.core software. Apparently, that audit report was never made available to Scytl.

2. The Finnish Ministry of Justice commissioned a group at the University of Turku to perform a security review of the software. They analyzed Pnyx.core version 1.7.1b, which we are informed is almost identical to the version used in the ODBP project, version 1.7.1c. A summary report on their audit is available on the Internet [14].

The Switzerland report was not available to us, and we have no knowledge of its findings. The summary report from the University of Turko was released only near the end of the review period, so as a result it was not helpful during the critical early part of our review period as we oriented ourselves and familiarized ourselves with the Scytl voting system. We downloaded and read that University of Turku report as soon as it became available, but we did not have access to any other technical details from that group.

As future reviews of the Scytl software are performed, we recommend that future reviewers should be provided with complete and unfettered access to all reports and technical data associated thereupon. We believe that in the future subsequent reviews could benefit from our experience. In particular, in accordance with the Statement of Work, we delivered confidential appendices and other materials to the FLDoE separately from the public report, to protect proprietary information or information that could endanger an election. Future reviewers should be provided with full access to all of these materials, to enable them to replicate our analysis and build upon our work. We anticipate that this could save future reviewers a significant amount of time.

4.2.3 The scientific literature identifies several types of verifiability and auditability properties that can be used to characterize the properties of a voting system (e.g., universal verifiability, voter verifiability, software independence, end-to-end auditability, independent auditability). Which of these properties, if any, does the target software provide?

Universal verifiability. Originally introduced by Sako and Kilian [23], this property requires that anyone be able to verify that the election results are correct using only the public outputs of the election system. The Pnyx.core ODBP 1.0 voting system does not provide universal verifiability. In particular, the only public output of the Pnyx.core ODBP 1.0 system (other than the election results) is the list of ballot identifiers. Since this list does not include any information about the choices made by voters, it cannot be used to verify the election results.

Voter verifiability. This property requires that individual voters be able to verify that the vote recorded for them by the voting system accurately reflects the choice the voter intended to make. In ODBP, the voter's choice is recorded in two ways:

1. The choice is recorded in digital form and stored in a digital ballot box (called the "voting service" in Section 3.2). The voter has no means of verifying that this digital copy is accurate. (Indeed, a malicious voting client could change the choice arbitrarily.) Thus, the digital record does not provide voter verifiability.

2. The choice is recorded on paper and stored in a receptacle similar to a traditional ballot box (called the "voter choice record" in Section 3.2). The voter can inspect this paper and verify that the choice it records is accurate. Thus, the paper record does provide voter verifiability.

Note that no mechanism in the voting system enables the voter to verify that the digital record contains the same choice as the paper record. Therefore, voter verification of the paper record does not by itself imply that a tally of the digital records accurately reflects voters' choices.

Counted as cast. This property, not specifically mentioned in the above question, is a component of universal verifiability and voter verifiability. It is a property that Scytl claims that the Pnyx.core ODBP 1.0 system provides. However, this claim is overstated: The system does not provide the counted-as-cast property (see §4.2.3.1).

Software independence. This property, defined by Rivest and Wack [21], requires that an undetected change or error in election system software cannot cause an undetectable change or error in the election results. The Pnyx.core ODBP 1.0 remote voting system satisfies this property because of the presence of the voter choice records. These records (after being printed and verified by the voter) need never encounter software again. Thus, the records can be used to detect any errors in the election results, including errors originating from software.

However, if the voter choice records were to be removed from the Pnyx.core ODBP 1.0 voting system, then the system would no longer be software independent. Changes or errors in the voting client or the mixing service could both lead to undetectable changes or errors in the election results.

Note that, roughly speaking, software independence implies that it is *possible* to audit the election results to check for errors, i.e., that there exists a set of procedures that would be capable of detecting an error in the election outcome. It says nothing about whether those procedures are actually used. Consequently, software independent equipment must be supplemented by appropriate procedures, to achieve the benefits of software independence.

Auditing properties. End-to-end and independent auditing of an election run by the Pnyx.core ODBP 1.0 voting system could be performed by manually counting the voter choice records. We are unaware of any other means of performing such audits.

The system documentation provided to us does not describe procedures for performing a manual count or audit of the voter choice records. The documentation does describe various kinds of electronic audits that can be performed. These electronic audits do not provide the same properties as a manual count and do not provide an end-to-end, independent method for verifying the accuracy of election results. The system documentation does not describe these limitations of the electronic audit methods.

4.2.3.1 Finding: Voter receipts fail to ensure votes are counted as cast

In the voting literature, *counted as cast* is a verifiability property which requires that votes cannot be changed after they are cast by voters. For example, if Alice casts a vote (in an election for favorite ice cream flavor) for Vanilla, then "counted as cast" would mean that the voting system (including the software, hardware, and humans involved) could not undetectably change Alice's vote to another flavor (say, Chocolate).

In the Pnyx.core ODBP 1.0 voting system, a voter receives a *receipt* at the end of the process of casting his vote. Scytl claims the following property regarding voter receipts:

> "Ballot verifiability. Voters [can] independently verify that their ballots have been correctly accounted for" [26, §1]. Note: The mechanism Scytl uses for this verification is the voter's receipt.

The ODBP project description similarly claims:

> "[C]ounted as cast receipt allows [voters] to individually verify that their vote was included in the final election tabulation" [9, §2].

Note that the proposed format for the physical voter receipt (i.e., the piece of paper the voter receives) also contains the phrase "Counted as cast" as a large heading.

The Scytl voter receipt does provide a way for each voter to verify that the mixing service—even in the worst case when the service is faulty or corrupted—tallied a vote for that voter, i.e., to check that their vote was not lost or deleted. Hence, the receipts provide a property that could be named "vote counted."

However, the Scytl voter receipts do not ensure that a vote cast by a voter cannot be changed, for example by a malicious insider that was able to corrupt either the voting kiosk or the mixing service, because the receipt contains no information about the voter's vote. Nor does the receipt protect against faults in the vote capture functionality of the voting kiosk or the tabulation functionality of the mixing service, because the receipt does not provide a way for a voter to verify that these parts of the two components have operated correctly. Thus, the receipts do not provide the counted-as-cast property.

Action: References to "counted as cast" should be removed from descriptions of ODBP. Receipts could perhaps be described instead as "vote counted" receipts. (Note that this is what they are named in [9, §11.1.1].)

Technical details: When a voter casts a vote, a *ballot identifier* is created for that vote. This is supposed to be a random (unique and unpredictable) number associated with the vote. This number is printed on the voter's receipt, and it is also contained in the encrypted vote that the voter casts. After all the votes have been tallied by the mixing service, the mixing service is supposed to output a list of all the ballot identifiers contained in the votes it processed. This list is then posted publicly, and a voter may check whether the ballot identifier printed on his receipt is present in the list. However, since the ballot identifier is independent of the choice the voter made (i.e., the candidate for which he voted), the ballot identifier cannot guarantee that the voter's choice was not changed. Indeed, a compromised mixing service could output the correct ballot identifiers, yet change every choice in every vote. The voters' receipts would in no way enable these these changes to be detected. See also finding 4.2.2.5 for further discussion.

4.2.4 Does the target software provide mechanisms for election auditing? If so, are those audit mechanisms reliable and tamper-resistant? What kinds of security failures are or are not likely to be detected using these audit mechanisms?

The system provides a method for election auditing by means of a 100% manual count of the voter choice records with subsequent reconciliation with the electronic tallies. The process for such a manual count, however, is not documented in the system, nor is any special provision made in the software for such an audit.

Note that the *voter receipt* generated by the Pnyx.core ODBP 1.0 system is not a sufficient audit mechanism. The receipt in this system allows a voter to verify that their vote was included in the published tally and thus provides a valuable capability that is typically not present in current postal voting systems. However the receipt does not help to verify that the mixing service correctly counted the vote, or that the voting client correctly recorded the voter's intention.

Finally, although process audits after an election can be completed to test the mixing service, by say, re-running the entire mixing service on the recorded inputs, such an audit is not sufficient since it does not check whether the electronic records correctly reflect voter intentions.

4.2.4.1 Finding: The integrity protection offered by the Pnyx.core ODBP 1.0 remote voting system exceeds that of existing vote-by-mail systems

In the Pnyx.core ODBP 1.0 voting system, two copies of each vote are stored:

1. An electronic record that is individually, digitally signed and end-to-end encrypted, and
2. A paper record of the voter's ballot (the VCR) that is verified by the voter and delivered back to the supervisor of elections by the poll workers that operate the remote kiosk locations.

These two copies are stored and delivered independently. The redundancy thus afforded by the copies provides significantly better reliability than vote-by-mail systems. In addition, the presence of redundant copies with different failure

modes means that, if appropriate audits and other procedures are performed, this system provides a clear improvement upon the reliability and integrity of vote-by-mail systems that transmit and store a single ballot copy, with few integrity checks possible.

Finally, both forms of storage independently provide more information for auditing certain aspects of election integrity than is available in vote-by-mail systems. For example, in the Pnyx.core ODBP 1.0 system it is possible to record the number of voters who signed in and attempted to vote, the number of ballots cast, and the number of ballots received. Much of this information is simply unavailable in vote-by-mail elections. Consequently, the Pnyx.core ODBP 1.0 system provides more information for auditors than is available in vote-by-mail elections.

4.2.4.2 Finding: Information needed to verify signatures on voter receipts is not specified for publication

The paper receipt printed for a voter's vote contains a digital signature by the voting service on the vote's ballot identifier. This signature provides evidence that the voting service accepted a vote with that identifier.

The voting client has access to the voting service's public key, thus can verify the signature. However, if the voter wanted independently to verify this signature, then the voter would also need access to this public key, as well as a certificate for the key. To our knowledge, publication of this key and certificate is not specified for the ODBP project.

Action: Okaloosa County should consider publishing the voting service public key and certificate to enable voters to verify the signature on their receipts.

4.2.5 What are the trusted components of the target software, and for what purpose are they trusted? Are trusted components and their security implications properly documented?

In high assurance systems, "trusted components" are system components whose contamination or malicious takeover would place the system in an unrecoverable or untrustworthy state. While "trust" might sound like a desirable feature, "trusted components" are undesirable elements of any system architecture, because they are components whose failure or compromise could cause the system to behave in unacceptable ways. Ideally, high assurance systems would incorporate sufficient redundancy and checks and balances to eliminate the need for trusted processes and components. In practice, the ideal of zero trusted components is rarely accomplished. Thus, a common practice in creating high assurance systems is to choose an architecture that minimizes and isolates trusted components within the system architecture.

We identified three trusted system components in the reviewed software.

- The Mixing Service is trusted to preserve the integrity of the election and the secrecy of voter's ballots. It can violate all election requirements.
- The Voting Laptop is trusted to accurately record the voters' vote and to preserve the anonymity of the voter. It can violate both integrity and secrecy requirements.
- The Key Generation Service is trusted to generate strong cryptographic keys and keep them secret.

We do not claim that these are the only trusted components, only that we identified these three as being trusted during our review. The important takeaway here is that if any of the components that we identify as trusted are compromised during election operations, the system could become vulnerable to dangerous exploits. In order to make the system safe in the face of these risks, extraordinary measures must be taken to ensure that trusted components are not compromised either before or during the election process.

The trusted nature of these components was not documented in the documentation provided to the review team. We see this omission as a significant documentation flaw that should be corrected.

4.2.5.1 Finding: The mixing service is trusted

Scytl makes the following claims regarding the security properties offered by the Pnyx.core ODBP 1.0 voting system [26, §1]:

- "Accuracy of results. [Impossible] for anyone to remove or alter the ballots."

- "Privacy of voters... impossible to correlate the votes to the identities of their respective voters."

Similar claims are made in the ODBP project description [9, §11.3]. However, the mixing service can easily violate the claims above:

- The mixing service can alter ballots, violating accuracy.
- The mixing service can correlate votes with voter identities, violating privacy.

Alteration of ballots would be detectable with a manual count of voter choice records (assuming those records have not also been altered). Correlation of votes and voters, however, would not necessarily be detectable.

Therefore, the mixing service is a trusted component. Since it generates and certifies keys used by other components, the mixing service is also the root of trust for much of the system. The mixing service therefore constitutes a critical vulnerability in the security of the Pnyx.core ODBP 1.0 voting system.

Faulty software. The mixing service could violate accuracy or privacy if there is an error in its implementation.

Action: The correctness of the mixing service software should be validated and certified.

Ideally, during the tallying process, the canvassing board would verify that the mixing service is running the certified software. Unfortunately, we are unaware of any reliable way to perform this verification. Nor did we validate the correctness of the mixing service software submitted for certification; that was beyond our resources.

Malicious software. Some possible threats to the mixing service include:

- insiders with authorized access to the mixing service (e.g., system administrators or operators),
- individuals (e.g., election workers) with authorized access to the physical room where the mixing service is stored or used, and
- Scytl employees, who might also belong to the other two categories.

These threats might be able to introduce malicious software on the mixing service by:

- embedding it in the mixing service source code,
- planting it during software installation, or
- inserting media into the mixing service machine (e.g., a USB dongle into an open USB slot, a CD-R into a CD drive, or a device into an open Firewire slot). The threat might have to deliberately load the malware from the media, or it might be possible to find and exploit an operating system vulnerability (e.g., in the filesystem) to automatically load the malware.

Unfortunately, we know of no practical way to reliably detect the presence of malicious software[7]. Consequently, the security of the mixing service depends upon procedural controls.

Action: The procedure that election officials will follow to secure the mixing service computer should be fully documented; partial documentation already exists in [9, §11.3]. We suggest that this procedure include controlling physical access to the mixing service computer. The procedure might also include physically blocking all USB, Firewire, and other ports.

Scytl informed us that they plan to configure the mixing service computer to disable autorun, which is a good step to include in this procedure.

Scytl initially informed us that they planned to install the mixing service software onto a computer, configure it as needed, and send the computer to Okaloosa County. This would introduce an additional risk that the Scytl employees

[7] Scytl initially suggested to us that malicious software on the computer could be detected by running a hash or checksum program on that computer. However, relying on a hash program for this purpose amounts to asking the computer to validate itself; if the computer is not trusted, then no answer from that computer can be trusted. By way of analogy, airport security could ask every traveler: "Are you a terrorist?", but because a true terrorist would presumably answer "No," such a question would be of limited value. In the same way, asking software "Are you malicious?" is of limited value, because malicious software could arrange to answer "No."

who perform the task could introduce malicious software. However, the Florida Division of Elections subsequently informed us that FLDoE performed the installation and configuration themselves during their functional testing of the voting system, and they plan to deliver these pre-installed and configured systems to Okaloosa County before the election (pending certification). In addition, the FLDoE informed us that they used hashing to validate this installation step. We consider this good practice, as it eliminates one potential source of risk.

Action: For any future delivers of equipment, election officials should consider following the model established by FLDoE and install and configure the mixing service computer on their own.

Finally, we note that there exist cryptographic techniques that provide accuracy and privacy of a mix, without relying on fully trusted software. The mixing service could be redesigned to use mixes based on zero-knowledge proofs [18] or randomized partial checking [12]. Even with just a single mix server, these could help to prevent the mixing service from undetectably altering votes. Guaranteeing privacy is likely to require multiple mix servers, each using independent secrets (i.e., canvassing board private key shares) and perhaps independent software implementations, which might be expensive.

4.2.5.2 Finding: The voting terminal is trusted

The Voting Client software must be trusted, for several purposes. First, we must rely upon it to accurately record the voter's votes, in the form that the voter intended. We must rely upon it to present all contests that the voter is eligible to vote on, and to present the options accurately. Second, we must rely upon it to protect ballot secrecy. We must rely upon it to avoid leaking how the voter voted to unauthorized parties, whether by subliminal channels or direct communication or by inadvertent leakage.

It may seem obvious that the software on the voting terminal must be correct in order to ensure proper system operation, but there are high assurance techniques that can prevent and/or detect malicious manipulation in sensitive devices such as the voting terminal. In this case, we find that if malware were introduced onto the voting kiosk, it could violate the requirements for a secure election. Thus, the voting kiosk and voting client software is a trusted component and must be protected with physical security measures.

Additionally, because of the way that the voting client software is downloaded on election morning from the voting proxy, it will be difficult or impossible to confirm that the voting terminal software was, or was not, executing the proper software during the operational period. This aggravates the challenges of attaining high assurance while using trusted components.

4.2.5.3 Finding: The key generation service is trusted

As mentioned earlier in §3.3.1, the credential generation service generates the signature keys for each voter and stores them in an encrypted format. More specifically, it stores the signature key and certificate on the voting server in an encrypted format using a randomly chosen password and it stores the password in an encrypted format on the Credential Provisioning Service using the key of the polling worker.

This key generation software must be trusted. First, the security of the voting protocol relies on the generation process to use a secure random number generation process. Each voter must have a different key; if the generation process were to use a predictable random number generator to generate signature keys, then a malicious participant might be able to forge signatures on ballots and therefore modify or cast ballots at will. There is another attack which is more difficult to detect. The key generation process could use a small but random seed to generate all of the signature keys. It could do this using a secure random number generation process, thereby generating keys which appear to be correctly generated. However, an adversary who knows the small seed could also forge signatures on ballots for any of the voters. A similar attack could be carried out on the passwords used to encrypt the signature keys.

4.3 Software Security Analysis

4.3.1 Does the target software contain effective controls or other safeguards to prevent or detect unauthorized tampering with election results, election data, or audit logs?

The software contains multiple controls and safeguards, including cryptographic techniques, that attempt to detect tampering. Many of these controls and safeguards are effective, though others have some shortcomings. The controls and safeguards do not comprehensively prevent all possible forms of tampering.

As a remote voting application, many of the cryptographic controls focus on detecting tampering with data while it is in transit over a network. Other controls are designed to protect against tampering with encrypted ballots while stored at the voting server. These controls can detect simple modification of an encrypted ballot. But these controls cannot detect deletion of a ballot while stored at the voting server. There are other mechanisms that can mitigate this risk, e.g., voter receipts and the official canvass. However, it is not clear how effective voter receipts will be because this depends upon human factors (e.g., how many voters will actually check their receipts?) and procedures (e.g., what will election officials do when they receive a complaint?) that were outside the scope of this review.

4.3.1.1 Finding: Software components can fabricate logs upon the first audit request

Various software components in the reviewed system keep electronic audit logs. These logs make use of a hash-chaining and digital signature mechanism to ensure log integrity.

However, software components do not commit externally to their logs. That is, the hashes and signatures are known to no one other than the software component until the time that the first audit request is made to the component, which might be long after the election is over. This means that, at that point, a corrupted component can generate an arbitrary log. After that point, the component will be caught in any attempt to change the log of events that occurred before that point, but there is no guarantee that the log it initially reported is actually correct.

The documentation available to us did not clearly identify the threat model which these cryptographic mechanisms are designed to address.

Action: Components should be queried for their logs periodically throughout an election. This will reduce the window of vulnerability during which a corrupted component can lie about its log. Scytl should clearly document the threat model that its cryptographic log mechanisms are intended to counter.

4.3.2 Are dangerous coding commands (e.g. strcpy) or structures employed?

Coding commands and constructs that are widely regarded as dangerous are present in the Scytl source code. Their presence in the source code does not necessarily imply the existence of resulting vulnerabilities and we did not find any actual vulnerability resulting from these constructs. Nonetheless, such constructs are dangerous because of the ease in which they can be used incorrectly, which in turn can lead to bugs or security vulnerabilities. For example, Microsoft is sufficiently concerned with the use of such dangerous functions that it has banned the use of certain APIs in its code in an effort to reduce the potential for vulnerabilities [11, p. 241–249], and high assurance development demands similar standards in election system source code.

It should be noted that we identified issues both within code written by Scytl and in external library components they access, including xerces, boost, and ace.

A search of the first batch of source code provided to the review team found substantial use of three well known C/C++ functions widely considered to be dangerous:

- strcpy(): we identified 395 uses, including 17 in pynx-core.
- strcat(): we identified 118 uses, including 20 in pynx-core.
- sprintf(): we identified 296 uses, including 29 in pynx-core.

The use of clearly defined coding standards designed to avoid dangerous constructs, combined with enforcement methods, such as static analysis or even simple searches, would reduce the likelihood of such issues occurring in this code. We found no documentation of secure coding standards in the documents provided to us.

We also found operations that are not routinely identified as dangerous that were used in a manner that made them dangerous. For example, in one instance identified in the Fortify scan results, a function in the basis library directs input from an input stream to a statically sized character array. While an attempt is made to ensure that the buffer is big enough, a more appropriate approach would have been to read the input into a C++ string, as opposed to a C style character array.

Our two scans of the Pnyx.core ODBP 1.0 source code also identified potential memory leaks, failure to check for error codes returned by functions, potential dereference of null pointers, and buffer overflows. The Fortify and Klockwork static analysis tools reported hundreds of warnings. It was not possible for the review team to manually review all of these warnings reported by the static analysis tools, and static analysis results typically include many false positives (warnings that are erroneous and do not represent an actual vulnerability).

In the best case, the constructs that we detected exist only in non-production code such as testing stubs where the resulting direct vulnerability is minimal.

In the worst case, if the dangerous coding constructs that we detected exist in production code, they could lead to faults difficult to detect.

Removing all occurrence of these constructs in the code base could eliminate all doubt.

Action: Voting system vendors should employ strong secure programming standards and processes to ensure that these standards are followed. The standards should be documented and provided to future reviewers. Voting system vendors should adopt static analysis tools designed to improve the security and quality of their code.

4.3.3 Does the target software contain malicious code that could compromise the election results?

It is generally infeasible to detect malicious software inserted into a large software code base, or to verify the absence of such malicious software. Because of time limitations and because of the general ineffectiveness of manual code review at finding malicious code, we did not devote resources or time to specifically look for malicious code. Moreover, we examined only a small fraction of the code base. Thus, our only conclusion in this area is that we did not see any sign of malicious code in the code that we carefully reviewed.

4.3.4 Is the cryptography designed and implemented appropriately for the purpose to which it is intended? Is the cryptography used correctly?

The cryptography employed by the Pnyx.core ODBP 1.0 system is used almost entirely to protect information while it is being transmitted over networks and stored on servers. There do not seem to be any fundamental flaws in the way cryptography has been used for this purpose. However, our review did identify a few small problems including: (a) layering violations in protocol design, (b) use of the same secret key for multiple purposes, and (c) incomplete verification of data during a protocol. Our review of the cryptography made no attempt to verify whether the implementation matches the stated design: instead, we focused on analyzing the cryptographic protocols as they were explained to us by Scytl.

Finally, we note that while there are cryptographic techniques to allow voters to verify that their votes are counted correctly as they have been cast, this system does not use such methods.

4.3.4.1 Finding: The randomness used to shuffle votes by the mixing service is cryptographically weak

The mixing service *shuffles* votes, randomizing the order in which they are output. This is to prevent correlation of voters and votes based on the order that votes are input to the service.

However, the random order that is used is not cryptographically strong. In particular, it is not unpredictable. This vulnerability could enable adversaries to learn what shuffle was used, thus violating the privacy of all voters.

Action: Cryptographically-strong randomness should be used for the shuffle.

Scytl agreed that this defect was not intended and promptly informed us that they have prepared a patch to the software to fix this problem. We have not reviewed their patch, but we expect that this defect in the software will be straightforward to fix. They explained that they intend to fix the problem by using the OpenSSL `RAND_bytes()` function, which should indeed yield an appropriate and satisfactory solution to this shortcoming.

Technical details: Scytl's code uses the C standard library random number generator `rand()` to generate randomness for the shuffle. This is a linear congruential generator (LCG), which is not cryptographically strong. In fact, learning just three sequential values from a LCG suffices to reveal the sequence of all future values from it. On some platforms, it contains other serious flaws: e.g., the least significant bit of successive outputs alternate 0,1,0,1,0,1,.... In addition, there are only 2^{32} possible seeds for this PRNG. As a result, one can narrow down the set of possible shuffles that could have been used. In some cases this may make it possible, given the shuffled votes, to determine how specific voters have voted. For instance, if we have side information (such as knowledge of our own vote and its location in the input and output sequence, as well as similar knowledge for a few friends), exhaustive search over all 2^{32} possible seeds might reveal the shuffle and enable reconstruction of how other voters voted.

In addition, the pseudorandom number generator is improperly seeded, due to an obscure bug in the seeding routine. The seeding code generates 4 pseudorandom bytes, stores them in a string, and dereferences a `char *` pointer to the start of the string, and then casts the result to a 32-bit integer type. The code is slightly convoluted; it appears that the programmer intended to generate a random 4-byte integer but got the logic wrong. As a result of this bug, the dereference returns a 8-bit pseudorandom value which is eventually used as the seed. Consequently, there are only 256 possible seeds and only 256 possible shuffles.

The rest of the Pnyx.core ODBP 1.0 code (e.g., in the voting client) correctly uses the (cryptographically strong) OpenSSL random number generator, and so should the mixing service.

4.3.4.2 Finding: The voting client will accept bogus public keys

During the voting protocol, the voting client receives a set of public keys and their certificates from the voting service [25, p. 75][27]. These are supposed to be the public keys of the canvassing board (P_{env}), the election administrator (P_{adm}), and the voting service (P_{VS})[8]. To check that these are indeed the keys of those agents, the client needs to verify the certificates that came with the keys.

However, Scytl disabled the functionality of the voting client that performs this certificate verification. If a voting service (either erroneously or maliciously) provides incorrect keys to the client, then the client will not detect this. The client might therefore encrypt votes and accept signatures under incorrect keys. This vulnerability could enable these attacks:

- A malicious voting service sends an incorrect public key for the canvassing board's encryption key (S_{env}). Then the voter's vote will be encrypted incorrectly, causing it to be discarded during tallying or possibly allowing an unauthorized individual to learn how the voter voted.
- A malicious voting service sends an incorrect public key for the voting service's signing key (S_{VS}). Then the voter's receipt will appear valid to the voting client, but in fact, will not be valid. Therefore, either the voting service or the mixing service would be free to discard the voter's vote, and the voter would be unable to protest this due to the lack of a valid receipt.

Scytl's stated reason [27] for disabling this functionality relates to the root signature on the certificates. This root signature is under a key that will belong to the Okaloosa Board of Elections and will be created a few days before the election. The Pnyx.core ODBP 1.0 voting client implementation requires this key to be present during compilation. However, certification of the system is occurring long before the key will be created, and recompiling to add the key would invalidate the certification. So to avoid this invalidation, Scytl chose to disable the code that relies on the key.

Whereas we understand how this choice solves the problem of invalidation, we do not understand why the Pnyx.core ODBP 1.0 voting client implementation could not receive this key as input during run-time configuration (instead of during compilation). For example, a kiosk worker could provide the key when the machine is booted each day. This would enable the voting client to check the certificates, thus defending against the attacks identified above.

Action: Future reviews should consider requiring enabling verification of the certificates, which would further seem to require Scytl to redesign this part of the voting client.

[8] All three public keys are sent to the client, along with the corresponding X509 digital certificate. See step 5 on p.75 of [25].

4.3.4.3 Finding: The voting client will accept bogus election identifiers

During the voting protocol, the voting client must include an *election identifier* (a unique number) as part of the construction of a voter's vote. In the current Pnyx.core ODBP 1.0 protocol, the client receives this identifier from the voting service as part of a message signed by the election administrator.

However, the client never checks this identifier against any other source. Thus, an erroneous or malicious voting service could supply an incorrect election identifier (e.g., one created during a test election with the same administrator key) to the client. This would cause the voter to cast what seems like a legitimate vote.[9] But the vote would not count in the current election: The mixing service would discard the vote because it did not contain the correct election identifier.

Action: The voting client functionality should be extended to input, as part of its configuration, the correct election identifier for the current election. The polling place worker, for example, could enter this value at the beginning of the day. The client functionality should further be extended to check this value against the value supplied by the voting service.

4.3.4.4 Finding: A cryptographic key pair is used for both encryption and signing

It is considered bad cryptographic practice to use a key both for encryption and digital signatures [2, p. 44]. This is for two reasons: it can create vulnerabilities (e.g., acquiring a decryption amounts to acquiring a signature under that key), and the *cryptoperiod*, or lifetime of the key, for each purpose is usually different (e.g., a decryption key may need to be destroyed long before a signing key, or vice-versa).

In the Pnyx.core ODBP 1.0 voting system, the canvassing board's "envelope" key pair $\langle P_{env}, S_{env} \rangle$ is used both to encrypt votes by the voting client, and to sign election results by the mixing service.

While we have not discovered any attacks that exploit this vulnerability, it is likely the case that the signing key should be destroyed immediately after the election (to prevent any future statements from being signed by the key), whereas the decryption key should be preserved long after the election (to enable any future audits). Consequently the reuse of this key for both purposes appears to be bad practice.

Action: A second key pair should be introduced. One key pair should be used exclusively for signing, and the other key pair exclusively for encryption.

4.3.5 To what extent does the system use cryptographic algorithms that comply with recognized standards or that are demonstrably secure under appropriate assumptions?

We discovered the following cryptographic *primitives* (basic building blocks) in the Pnyx.core ODBP 1.0 voting system: RSA encryption, RSA digital signatures, 3DES encryption in CBC mode, and SHA1 hashing. All of these are well-known algorithms that comply with recognized standards. However, the particular bit strengths chosen by Scytl for these primitives may be insufficient for use in ODBP.

The Pnyx.core ODBP 1.0 voting system also uses cryptographic *protocols* (higher-level constructs that employ the primitives). The primary examples of these are the voting protocol (see Section 3.2) and the mixing protocol (see Section 3.3.5). These are not well-known algorithms; rather, they are of Scytl's own design. Scytl has not produced a security proof for these protocols.

4.3.5.1 Finding: The bit strength of the cryptographic schemes was not chosen conservatively

The *bit strength* of a cryptographic scheme describes how much work is required to break the scheme by brute force. NIST [2] has estimated the bit strength of various schemes and the time period for which they may be used to secure data. Depending on how long the state of Florida requires election data to be retained, the bit strengths used in the Pnyx.core ODBP 1.0 voting system may be insufficient by NIST's standards. If, as discussed in §3.3.6, all election data and keys will be destroyed within 22 months, then Scytl's key lengths (with one exception, SHA-1 hashing) are sufficient for an election that takes place in the year 2008, as does ODBP.

[9] The voter actually could detect the incorrect election identifier by examining his receipt, which contains the identifier in plaintext. However, voters would need to be trained to do this, and the correct election identifier would need to be made well-known by publishing it somewhere at the beginning of the election.

Action: The bit strength of these cryptographic schemes should be increased so as to secure the data the schemes protect for as long as the state of Florida will retain election data.

Technical details: The Pnyx.core ODBP 1.0 voting system uses the cryptographic schemes described in Table 4.1. The bit strength and security period in Table 4.1 are as estimated by NIST [2, Tables 2–4]. Note in particular that 1024-bit RSA is estimated to be secure only until 2011 (and that SHA-1 is already deemed insecure). Thus, if election data will be retained by the state of Florida until 2011, then 1024 bit RSA may not be sufficiently strong for use in securing this data, according to NIST's criteria.

Table 4.1: Bit strength of cryptographic schemes

Scheme	Key length	Bit strength	Secure until
SHA-1	N/A	69	insecure as of 2007
RSA	2048	112	2011–2030
RSA	1024	80	2008–2010
3DES (3TDEA)	168	112	2011–2030

Note that the Pnyx.core ODBP 1.0 system uses RSA in two different ways. First, it uses RSA with 2048 bit keys in Open VPN. Second, it uses RSA with 1024 bit keys in its application-level protocols.

4.3.5.2 Finding: No security proof is given for any protocols

In the cryptographic literature, *security proofs* are used to increase confidence in the security of a protocol. A typical security proof starts with some well-known computational hardness assumption (informal examples include "factoring is hard" and "taking discrete logarithms is hard"), then shows that any successful attack on the protocol in question would violate the hardness assumption. Such proofs have been employed to establish the security of voting schemes, from some of the earliest [3] to the state-of-the-art [13].

However, Scytl has not constructed security proofs for any of its protocols. In general, the absence of such proofs reduces our confidence in the security of the protocols. This is particularly true for the voting protocol, which was not clearly and completely specified in documentation available to us (see Finding 4.1.1.1). For the mixing protocol, though, such a proof appears trivial. This is because the entire protocol executes on the mixing service, which is completely trusted.

Action: A security proof for at least the voting protocol should be required from Scytl as part of future reviews.

4.3.6 Do the key management processes generate and protect strong keys?

Due to time constraints, we did not study the implementation of the key generation process. Thus, we do not know whether the cryptographic keys are generated securely. The system as designed does not protect against some attacks involving stolen smartcards or compromise of the Mixing Service computer.

4.3.6.1 Finding: The canvassing board and election administrator key shares are not, by default, adequately protected against loss or theft of smartcards

The canvassing board's private key S_{env} and the election administrator's private key S_{adm} are both split into *secret shares*. These shares are stored on smartcards under password-based encryption.

However, if an adversary were to gain access to one of these smartcards, no cryptographic or tamper-resistant mechanism prevents the adversary from copying the encrypted share off of the card. The adversary would need very little time to perform this copy. The adversary could then, based on the minimum password strength required, quickly crack the encryption. This would enable the adversary to learn the private key share. If the adversary were able to repeat this attack and learn a sufficient threshold of the key shares, then the adversary could reconstruct the private key. This would enable a variety of attacks on election privacy and integrity.

Action: To make it more difficult for adversaries to copy values off of the smartcards, physical security measures should be required—e.g., each card might be kept in a separate safe, and the safes might all be dual control. To make it infeasible for adversaries to crack the encryption, the minimum password strength should be increased. While this does increase the burden on humans for remembering passwords, it is a worthwhile and not unreasonable burden.

Technical details: By default, the minimum password strength for the encryption is 8 characters, at least one of which must be non-alphanumeric. Such a password, according to NIST's estimates [4, Table A.1], has only about 18 bits of entropy. This is easily crackable.

4.3.6.2 Finding: The architecture does not protect private keys against compromise of the Mixing Service computer

Before the start of the election, the canvassing board's private key S_{env} and the election administrator's private key S_{adm} are both split into *secret shares* using *Shamir secret sharing*. These shares are stored in smartcards and the smartcards are stored until it is time to count the votes. At that time, canvassing board members authorize tabulation of the election by inserting the smartcards into the Mixing Service computer and providing their passwords. The Mixing Service computer recovers their shares and uses the Shamir scheme to reconstruct the canvassing board's private key S_{env}. This private key is temporarily stored in RAM on the Mixing Service computer and used as needed to decrypt the encrypted votes, then is deleted after it is no longer needed.

Note that this private key provides the ability to identify how individual voters have voted and trace votes back to their source. It also provides all the information that is needed to count the votes. Consequently, it is very sensitive and must be protected. Yet it exists in cleartext on the Mixing Service computer for a brief period at certain points during the election cycle. If this computer is compromised or infected with malware while or at any time before the private key is stored on the Mixing Service, then unauthorized individuals may obtain access to this key. Likewise, an insider with access to the Mixing Service computer during the time when the private key is being used to decrypt votes could make a copy of the private key.

In short, the Shamir secret sharing is of partially limited utility. It appears to limit the time window during which the private key is available on the Mixing Service, but it does not prevent malware introduced onto the Mixing Service before that time from obtaining the private key. As a result, the secret sharing mechanism is possibly less effective than it might appear.

Action: To improve the effectiveness of distributing the key among several members of the canvassing board, it might be worthwhile for Scytl to consider using what is known in the literature as a *threshold cryptosystem*. However, this would require redesigning the system architecture to include multiple mix servers, one for each board member.

4.3.7 Is data integrity protected and checked at each use?

There was insufficient time during the review period to examine this issue in depth. The protocol provided by Scytl does include data integrity checks for data sent across the network. It was not determined during this review whether the implementation correctly performed such checks.

4.3.8 Is device integrity protected for removable media?

Data entry and exit points are common computer application attack avenues and the technology to allow computing devices to strongly attest to the authenticity and integrity of removable media is immature. The challenge this causes to application security is that it is very difficult to prevent malicious actors from inserting non-authentic devices into application nodes. Since the state of practice requires unique solutions for removable media handling, we attempted to consider how the reviewed system handles removable media attestation.

One important piece of removable media involved is the Live CD. Apparently there is a process for checking that this CD has the expected contents, by distributing the expected hash, though this protection mechanism has certain limitations. (See finding 4.2.2.3 for further discussion.) Thus, the Live CD must be rigorously protected by physical security measures.

Due to time constraints, we were unable to investigate issues associated with other removable media that may be used in the system, such as the removable media used to transfer data to and from the mixing service computer.

4.3.9 Are communications channels properly privacy and integrity protected?

The cryptographic mechanisms provide privacy and integrity protection for all communications channels sent over the Internet. We expect these protections will be sufficient to prevent outsiders from modifying or intercepting those communications. These protections are not perfect, and we should be prepared for the possibility that they could be compromised by unexpected vulnerabilities, but they appear to be adequate for a small-scale election with only a limited number of remote polling places.

This level of protection is important, because it is the primary line of defense against hacking by outsiders with no special access. The Pnyx.core ODBP 1.0 voting system relies upon the Internet for transmitting software to the voting clients, for transmitting encrypted votes, and for other purposes. These communications are critical, so we must ensure that unauthorized individuals cannot interfere with them. There are many individuals who have a motive to attack a US election and who have no special access to the voting system but who do have access to the Internet. For instance, those individuals might be thousands of miles away, residing in a foreign country, not on US soil and beyond the reach of US law. Therefore, we must ensure that it is not possible to attack the Pnyx.core ODBP 1.0 voting system over the Internet. Fortunately, the system uses several cryptographic mechanisms to encrypt all communications that transit the Internet, and based on the information currently available to us, we expect this will be sufficient to prevent such attacks.

We did identify poor design practices and several areas for concern in some of the cryptographic mechanisms used for securing communication channels. (See, e.g., Findings 4.3.4.2, 4.3.4.3, 4.3.4.4, 4.3.5.1, 4.3.5.2, 4.3.9.1.) These shortcomings are indeed troubling and represent a failure to follow accepted design and implementation principles and practices. However, our ultimate analysis is that—while these weaknesses should be fixed—other mechanisms in the system appear to compensate for these weaknesses.

In addition, we did not evaluate whether these mechanisms provide privacy and integrity protection against insiders. It is conceivable that insiders with special access might be able to circumvent the cryptographic protections, but we simply did not examine this kind of threat.

We did not evaluate the possibility of remote attacks on the Scytl voting servers (e.g., the proxy service and the voting service). For instance, can an outsider with no special access break into the machine that hosts the proxy service, the voting service, or the associated OpenVPN servers and databases? Are there any vulnerabilities in the operating systems or network services running on those machines? What network services and ports are exposed on those machines? Unfortunately, while these issues are critical to the security of the Pnyx.core ODBP 1.0 voting system, we were not able to evaluate them. We did not have access to the configuration and other information that would have been needed to evaluate the risk of this kind of attack. Scytl assured us that these machines are "hardened" against attack, but we were not able to verify these assurances for our self in detail. We were provided with the source code and binary executables for some of these services, but we were not provided with sample server machines configured as they would be in an election, and the voting system documentation does not clearly describe how these machines will be configured during the election. Consequently, it was simply not possible to check for vulnerabilities of this nature, so we leave this as an open question for future security reviews of the Pnyx.core ODBP voting system. We recognize that this is a substantial gap in our analysis.

4.3.9.1 Finding: Mutual authentication is provided by lower-level protocols, not by the application-layer voting protocol

Scytl's implementation of security on the channel between the voting client and voting proxy relies on three layers of cryptographic protocols:

1. VPN: Both the client and voting proxy will only accept connections on a VPN. Accessing the VPN requires a certificate that should only be available to insiders (election officials, polling place workers, etc.). The VPN provides encryption and authentication for all traffic transmitted through it.

2. SSL: The client uses SSL to connect to the voting proxy. The SSL-encrypted traffic is transmitted through the VPN.

3. Application-level: The voting protocol uses a challenge-response protocol, namely ISO/IEC 9798-3. The application-layer protocol messages are transmitted via the SSL-encrypted channel.

Scytl claims that this architecture provides "in-depth security" [25, p. 73].

At Layer 1, the VPN provides mutual authentication. Scytl informed us that each laptop receives its own public key pair, and this key is used to authenticate the laptop to the voting proxy [28]. Similarly, each laptop uses the voting proxy's public key to authenticate the voting proxy. Consequently, the VPN provides mutual authentication of both endpoints and then establishes a secure channel that is then used for all further communications.

Layer 2 does not provide mutual authentication. The use of SSL at Layer 2 authenticates the proxy to the client, but not the client to the proxy, because Scytl omitted the use of client certificates in the SSL protocol.

Layer 3 does not provide mutual authentication. The application-layer voting protocol authenticates the client to the proxy, but not the proxy to the client, because Scytl omitted this portion of the ISO/IEC 9798-3 protocol.

The presence of mutual authentication in Layer 1 means that their combination does provide mutual authentication. This provides a defense against man-in-the-middle attacks and other attempts to tamper with messages while in transit. The absence of mutual authentication in the application-layer protocol arguably is in tension with layering principles, because the security of the application-layer voting protocol relies upon the existence and proper configuration of the VPN. In addition, splitting the implementation of mutual authentication across multiple layers in this way weakens the claim of in-depth security.

Action: Scytl should consider completing the use of ISO/IEC 9798-3 to provide mutual authentication.[10] Optionally, the use of SSL could be extended to include client certificates. Both actions would strengthen the claim of in-depth security with respect to mutual authentication.

Action: Assuming the application-layer voting protocol is extended to provide mutual authentication in Layer 3, Scytl should consider extending the protocol to exchange and authenticate a session key and use it to encrypt all further messages. Further, the integrity of these messages should be protected via digital signatures or message integrity codes, which may require authenticating an additional key. All this might best be accomplished with a different protocol than ISO/IEC 9798-3, since that protocol uses only signatures, not encryption.

4.3.9.2 Finding: OpenVPN is not high-assurance software, but it appears to provide a reasonable level of communications security

The Pnyx.core ODBP 1.0 voting system uses the open-source OpenVPN software to establish a virtual private network (VPN). This is a key defense against attacks by outsiders with no special access to the system—e.g., attacks by hackers with no access other than an Internet connection. Because of the critical nature of the OpenVPN software, one of us (Wagner) examined the OpenVPN software. The lesson from that examination seems to be that while OpenVPN is not perfect, it appears to provide a reasonable level of communications security. We provide technical details next.

Scytl informed us that they use the most recent version of the OpenVPN software, and that they intend to upgrade the OpenVPN software as necessary to use the most up-to-date version in any particular election. At the time of this review, the most recent version of OpenVPN was version 2.1_rc7, so we downloaded the source code for this version from the official repository and analyzed its security properties. Consequently, in some places our analysis may be specific to version 2.1_rc7. Our review process included manual code review, documentation review, and application of the Fortify SCA automated source code analysis tool. We shared our findings confidentially with the OpenVPN developers, and the OpenVPN developers graciously responded with further information and improvements to the software.

OpenVPN appears to have several points in its favor:

- OpenVPN appears to use cryptography properly. It uses standard algorithms (Blowfish and SHA1, by default), modes of operation (Blowfish-CBC and SHA1-HMAC, by default), standard key-exchange protocols (TLS, typically with a Diffie-Hellman-based ciphersuite). Its design avoids many subtle and common errors: for instance, it uses independent keys in each direction; it uses cryptographically pseudorandom IVs in CBC mode; it provides robust replay detection; and it uses full-strength cryptographic keys, not export-weakened algorithms.

  OpenVPN's packet encryption format uses the Encrypt-then-Authenticate generic composition, a choice that has been proven in the cryptographic literature to have excellent theoretical properties. In particular this mode

[10] As part of this, the second message in the protocol should be extended to include the identity of the voting proxy. This is an optional field under ISO/IEC 9798-3 that Scytl chose to omit. Its omission introduces a vulnerability to the attack that Gavin Lowe found on the Needham-Schroeder protocol [15]. Including the field would eliminate the vulnerability.

of operation provides IND-CCA2 and INT-CTXT security[11] under reasonable assumptions about the security of the individual primitives. Consequently, when OpenVPN is used appropriately and assuming there are no implementation errors in OpenVPN, OpenVPN's cryptographic algorithms should authenticate both endpoints and should establish a secure channel that offers both confidentiality and integrity protection.

This is indeed an important advantage, and one that should not be underestimated. There is a history of VPN software that uses non-standard algorithms, modes of operation, and protocols that subsequently turned out to be weak, insecure, or sub-standard. It was refreshing to see that, in contrast to past competitors, OpenVPN avoids these pitfalls and follows best practices in cryptography. Assuming that OpenVPN is free of implementation errors or other bugs, the cryptographic algorithms that OpenVPN uses appear likely to provide a high level of security: we would be quite surprised if the easiest way to attack OpenVPN is to attack the cryptographic algorithms directly.

- OpenVPN uses the OpenSSL library for the implementation of all cryptographic algorithms, which is a good choice that should help to avoid many kinds of implementation errors in the cryptography. OpenVPN uses OpenSSL's cryptographic PRNG for random number generation and key generation, which is a good choice that should help avoid problems associated with poor randomness.

- OpenVPN is open-source software, and the source code is freely available for public download on the Internet. This is potentially positive for security and trust, because it provides interested parties with an opportunity to analyze the software. In particular, this means that other researchers and readers of this report can examine the security of OpenVPN for themselves and confirm or refute our findings about OpenVPN.

- OpenVPN appears to have an active user community. Some users have written tutorials and documentation describing how to use OpenVPN in several scenarios. There is an active mailing list where users can share problems they encounter and exchange solutions and work-arounds that have worked for them. This is positive, because it makes it more likely that OpenVPN users can find technical support.

- OpenVPN appears to be actively developed. At least one OpenVPN developer continues to fix bugs and improve the software. In our experience interacting with this developer, he responded rapidly to our reports of potential security weaknesses in the software. He also developed and released software updates promptly, in cases where revisions to the software were appropriate. This is positive, because it makes it more likely that OpenVPN will continue to be maintained in the future. It also makes it more likely that fixes will be developed rapidly for any security vulnerabilities that might be discovered in the future.

However, our examination also revealed some areas of potential concern:

- OpenVPN is written in an implementation language that increases the risk of security vulnerabilities. OpenVPN is written in C, a language that is not memory-safe and that is known for buffer overruns and other classes of vulnerabilities. Unfortunately, when programming in C it is easy to make an inadvertent mistake with serious security consequences, and the C programming language and standard libraries do not help programmers avoid these mistakes. Consequently, programmers must be especially careful to avoid mistakes; and since programmers are human, mistakes sometimes happen even when everyone is careful. The risk is that a single mistake when handling untrusted data can compromise the security of the entire program. For this reason, OpenVPN's use of the C programming language and standard libraries represents a potential security risk.

- OpenVPN uses a monolithic architecture that increases the risk of serious security vulnerabilities. OpenVPN consists of approximately 45K source lines of C code (excluding blank lines and comments, as counted using David Wheeler's sloccount tool). Typically, all of this code runs in the same process, with root privilege. As a result, a single security bug anywhere in the code could cause sweeping harm.

  For high-security applications, it is generally considered good practice to architect the system to reduce the size of the Trusted Computing Base (TCB)[12]. For instance, kernelized architectures and "privilege separation"

[11] IND-CCA2 and INT-CTXT refer to two technical notions of security for communication security. Roughly speaking, IND-CCA2 refers to confidentiality, even against adaptive chosen-plaintext/ciphertext attacks, and INT-CTXT refers to integrity against adaptive chosen-plaintext/ciphertext attacks. These notions have been made precise and analyzed in depth in the scientific literature on cryptography.

[12] The TCB is a technical term that refers to the portion of the code that must be trusted, i.e., the portion where a bug in that part of the code could defeat system security.

techniques are examples of approaches that attempt to follow this principle. In these architectures, the software is structured so that most of the software is untrusted, and bugs in the untrusted portion of the software cannot defeat system security (or at least their impact will be limited). The justification for this principle is that long experience with software engineering has shown that bugs are inevitable—any sufficiently complex piece of software is likely to have bugs and other defects, even if great care is taken to avoid or minimize them, and the more code one writes, the more bugs one should expect—so it makes sense to choose architectures that reduce the impact of likely bugs, or that reduce the expected number of bugs in the most critical portions of the software by reducing the complexity of that portion of the software.

OpenVPN does not follow this generally accepted design principle. As a result, our experience with engineering systems suggests that we should expect that OpenVPN contains bugs, and some of these bugs could be harmful to system security. These bugs may be very difficult to find, and they may at present be unknown to adversaries, but it is very unlikely that OpenVPN is completely free of bugs.

For instance, standard estimates are that one should expect at least 1–2 bugs per thousand lines of code, even with excellent software development practices (higher bug rates should be expected for middle-of-the-road or subpar development practices). With 45K lines of code, this suggests that we should expect OpenVPN to contain at least 45–90 bugs—most of them presumably undiscovered to date. Because of the monolithic architecture, any one of those bugs could (at least potentially) have serious consequences: there is no architectural feature that would be guaranteed to limit the impact of those bugs.

The latest releases of OpenVPN contain new code to support a form of privilege management. However, this feature must be enabled with a special configuration, and it is available only on certain operating systems.

- OpenVPN does not use high-assurance software development processes. There is no clearly documented threat model, only partial documentation of the design of the application, and no assurance case setting out the evidence that the system meets its security goals. We did not see any evidence of formal code reviews, design reviews, architectural risk analysis, independent security reviews, or other processes that would normally be a part of a high-assurance software development project. The software engineering practices we saw appeared to be consistent with practices that are typical for most commercial or open-source software, rather than for the tiny fraction of software projects that require high assurance. The trade-off is that ordinary software development processes are significantly more flexible and affordable than high-assurance development, but come with a higher risk of bugs, defects, and vulnerabilities.

- Scytl is using a beta version of the OpenVPN software, rather than a stable version. In particular, the system documentation specifies use of 2.1_rc7 or later [29]. Version 2.1_rc7 is a beta release of OpenVPN: as the OpenVPN web site states, "The OpenVPN 2.1 beta series is ready for testing and limited production usage" [20].

  By way of background, the term "beta version" is sometimes used to refer to recent or less mature versions of the software that may not have been fully tested. It refers to a model of software testing based on practice among some commercial firms. In this model, after developers writing new code, it is subject to internal testing. Then, the software is subjected to so-called "alpha testing" and "beta testing". Alpha testing involves use of preview versions of the software by a very limited pool of users, to check for problems. Then, after a period of alpha testing, any problems found are fixed and a beta version is shipped to beta testers. Beta testing involves testing by a broader pool of users. At some point, bugs found by beta testers are fixed and a stable version is made available for general use. It is understood that alpha testers are guinea pigs and alpha versions of the software may have severe bugs that were not detected in the developers' testing; beta versions are less likely to have severe bugs, if they have already been alpha tested, but are less solid than an official stable release of the software.

  Even when formal alpha testing and beta testing is not used, many software developers label their software releases as an "alpha version", "beta version", or "stable release" to convey their expectations about the level of quality and likelihood of bugs of the software. When developers tag software as an "alpha version", this often means: "user beware, this software may contain severe bugs." Software tagged as a "beta version" indicates that the software may not have been tested as thoroughly as a stable release. Finally, stable releases are those that have already been used successfully by many users, so there is an corresponding expectation of higher quality and fewer bugs. The tradeoff between a beta version and a stable release is usually that the beta version is more recent and contains more features, but also has a higher risk of bugs, while the stable release is older, more mature, and less likely to contain bugs, but also may be missing some of the newest features.

The OpenVPN developers make available both a beta version of their software and a stable release. Scytl has chosen to use the latest beta version. This choice means that the Pnyx.core ODBP 1.0 voting system gets the benefit of the latest features and additions to the software, but also increases the risk of bugs in OpenVPN that could affect an election.

- In most configurations, OpenVPN appears to be highly susceptible to denial-of-service attacks. In the worst case, such attacks could potentially render voting servers unreachable and disrupt ability to vote electronically.

  The technical explanation is as follows. When a new peer attempts to contact OpenVPN, the first thing that OpenVPN does is attempt to authenticate the peer using a public-key cryptographic protocol. This requires the OpenVPN instance to perform a non-trivial amount of computation, at least several milliseconds of CPU time. This means that simply by opening a connection to an OpenVPN instance, an attacker can cause that OpenVPN instance to perform milliseconds of computation. An attacker who initiates new connections rapidly enough (e.g., opening thousands of new connections per second) will likely overwhelm the capacity of the OpenVPN machine with useless work. This could cause the OpenVPN machine to become so overloaded as to be unusable or unreachable for legitimate clients. One way that an attacker could mount such an attack is by renting a number of nodes on a botnet and using it to mount a distributed denial-of-service attack. Such attacks can be very difficult to fight.

  We did not investigate this attack in enough detail to determine how much it might cost an attacker to disrupt an election in this way. The cost will depend partially upon how much control the attacker has over the amount of computation that OpenVPN is forced to perform for each new connection, and upon the bandwidth required to initiate a new connection; these costs are not immediately clear and would require careful examination. As a speculative, back-of-the-envelope estimate, suppose that it were possible to cause an OpenVPN instance to perform 2 ms of computation per connection attempt, and each connection attempt required sending approximately 500 bytes. Under this set of assumptions, a steady stream of 500 new connection attempts per second would suffice to overload the OpenVPN instance, and this would require 2 Mbps of bandwidth. Researchers who have studied the cybercrime underground report that attackers can purchase bots (compromised hosts) at an estimated cost of approximately $5–25 per host [10]. If we conservatively estimate that each compromised host has a 100 Kbps upload link (which would be representative of a typical consumer-grade DSL or broadband link), then about 20 compromised hosts would be needed to mount such an attack, and these 20 bots could be purchased for approximately $100–500. We did not test this kind of denial-of-service attack, so this analysis is inherently highly speculative and should be treated as unconfirmed.

  This rough estimate assumes that the defenders do nothing about the attack, but in practice defenders would certainly notice such an attack and might block the IP address of each bot that is discovered to be participating in such an attack or take other countermeasures. We expect that it would be possible for the defenders to defeat the simplistic attack outlined above.

  Nonetheless, it exposes a category of risk: OpenVPN is not designed or intended to protect against denial-of-service attacks. As far as we can tell, defeating such attacks was simply not one of the design goals of OpenVPN. Consequently, there may well be other, more sophisticated denial-of-service attacks that could be used to overload an OpenVPN instance and the machine it is running on and that would be harder to defeat. We are concerned that this could potentially cause the voting proxy and voting server to be unusable or unreachable and could prevent voters from casting their votes electronically, which could in turn disrupt an election. The effect this would have upon voters would likely depend upon the backup procedures in use by election officials.

  In short, we suggest that election officials should be prepared for the possibility that OpenVPN could possibly be susceptible to difficult-to-stop denial-of-service attacks. We do not know of a specific vulnerability of this sort, but given the design and stated goals of OpenVPN, it also would not be surprising if such an attack were possible. We expect there are individuals who would welcome the chance to disrupt a US election or disrupt the ability of our uniformed military voters from casting voters. We expect that if OpenVPN is susceptible to denial-of-service attacks, these (hypothesized) attacks could plausibly be mounted by an attacker anywhere in the world, possibly beyond the reach of US law. We also know that some cybercriminals are highly technically skilled and would have the capability to analyze OpenVPN to determine whether it could be attacked in this way. If a difficult-to-stop denial-of-service attack against OpenVPN were discovered, there might be a temptation to use it to attack US elections.

We suggest that election officials examine what sort of backup procedures may be appropriate to enable overseas voters to vote, in case the voting servers are unreachable or unavailable. The availability of a way to vote, even if the voting machines are unusable, would provide an effective defense against the risk of denial-of-service attacks and would mitigate any concerns about OpenVPN's possible susceptibility to denial-of-service attacks.

As a technical mitigation, OpenVPN provides a configuration option that is intended to deter denial-of-service attacks. Scytl could consider enabling this configuration option. We describe the details later.

- We found that the version of OpenVPN that we analyzed (2.1_rc7) contained a remotely exploitable security vulnerability that could, under certain circumstances, enable a malicious OpenVPN server to take control of the OpenVPN client. This vulnerability was present in the beta release of OpenVPN that we analyzed, but it was not present in the stable release of OpenVPN. We reported this vulnerability to the developers of OpenVPN [30], and they confirmed our analysis and promptly issued a new beta version (2.1_rc9) that eliminates this vulnerability. This vulnerability is only exploitable under certain conditions. We quote from the OpenVPN change log:

  > Security Fix – affects non-Windows OpenVPN clients running OpenVPN 2.1-beta14 through 2.1-rc8 (OpenVPN 2.0.x clients are NOT vulnerable nor are any versions of the OpenVPN server vulnerable). An OpenVPN client connecting to a malicious or compromised server could potentially receive an "lladdr" or "iproute" configuration directive from the server which could cause arbitrary code execution on the client. A successful attack requires that (a) the client has agreed to allow the server to push configuration directives to it by including "pull" or the macro "client" in its configuration file, (b) the client successfully authenticates the server, (c) the server is malicious or has been compromised and is under the control of the attacker, and (d) the client is running a non-Windows OS.

  This security vulnerability could potentially affect the Pnyx.core ODBP 1.0 voting system. If an attacker gained access to the machine running the OpenVPN server, then the attacker could potentially leverage this access to take control of all voting clients and introduce malicious software onto the voting kiosks. This could allow malicious software to spread from the Voting Proxy machine to the Voting Client machine, which means that the Voting Proxy becomes a trusted component whose security is critical. Because malicious software on a Voting Client could have severe consequences, this issue heightens the importance of protecting the Voting Proxy and associated servers against compromise.

- The version of OpenVPN that we analyzed (2.1_rc7) did not consistently follow secure programming practices, used risky code constructs, and was not programmed defensively. For instance:
  - The code used risky APIs, such as `system()` (instead of `execve()`).
  - Many of the buffer-handling functions had no local protection against integer overflow. While we are not aware of any specific vulnerability, this coding practice makes it difficult to be sure that the code is free of buffer overflow vulnerabilities.
  - Some length fields were used for multiple purposes, and an invalid length was assigned in some places to represent an error condition. This introduces a risk of buffer overflow vulnerabilities, if the length fields are not consistently checked for invalid values, and makes it more challenging to review the code to verify the absence of buffer overrun bugs.
  - Return values from system calls were not consistently checked for error conditions.
  - In some cases, file paths, X509 subject names, and other strings were silently truncated if they were too long. This violates good security programming practices and could in obscure situations potentially introduce security risks [6, STR03-C].
  - Temporary files were created using an insecure scheme, which generated predictable filenames and opened them using APIs that would follow symbolic links. Because temporary files are ordinarily opened in the current directory that OpenVPN is started from, the security consequences may be minor, but we considered this poor programming practice.
  - Under some conditions passwords are exposed via the environment and thus visible (on Unix systems) to other users who can run `ps`. This behavior was enabled only if a specific configuration directive is enabled.
  - Under some conditions passwords could be exposed in cleartext in the OpenVPN logs. It appears this only occurs if the debugging level is specifically set to a sufficiently high value and only if plugins are in use.

- We found format string bugs. These were likely harmless, as it appears they could only be exploited given control over portions of the OpenVPN configuration file or command-line arguments. Nonetheless we considered this poor programming practice.
- We found path traversal issues where the code built file paths using string concatenation. The code correctly handled many common path travel risks, such as the presence of `..` and slashes in the paths, but on Windows platforms, it did not handle special Windows filenames (e.g., `CON`, `LPT1`, `PRN`, etc.).
- We found a remote information leak that could potentially leak cryptographic keys, depending upon how the compiler laid out information in the OpenVPN address space. This occurred due to a combination of two issues in the code. First, the code does not consistently overwrite cryptographic keys and other secrets after they are no longer needed, so they may be left in memory after they are used. Second, one place in the code read uninitialized data from the stack and included it among information that is returned over the network to the remote peer. If the stack contained confidential information left over at the same location on the stack where it was read without initialization, the confidential information could be revealed to a remote adversary.
- The client did not perform proper input validation on configuration directives received from an OpenVPN server. In particular, OpenVPN supports a mode of operation where the server can push certain configuration directives to the client; this mode is very useful. However the client is too trusting of the server and does not fully validate the options. This systematic deficiency is associated with the security vulnerability disclosed, and could have other consequences as well.

We also find some code quality issues that appear to have no direct relevance to security, but that would support the expectation that OpenVPN is not completely free of bugs:

- We found a bug relating to wrap-around of sequence numbers after 2^{31} packets are sent. In practice, it is unlikely that this would affect most users, but it could in principle cause very rare and hard-to-debug reliability issues.
- We found dead code, i.e., code that was never used or called.

We reported all of these issues to the OpenVPN developers and asked them to check our analysis. An OpenVPN developer responded immediately and responded with his analysis of our reports. In almost every case where we had identified a legitimate criticism, the OpenVPN developer reported that he had revised the code to address these criticisms and would be releasing another version of the software that incorporates these fixes. We did not examine these fixes.

In several cases, the OpenVPN developer reported that our analysis was in error. We accept the OpenVPN developer's response and retract those claims. We thank the OpenVPN developer for these corrections.

We are grateful to the OpenVPN developer for his prompt response. We consider it a positive sign that the OpenVPN team responds to criticism in such a constructive, professional, and prompt manner and works to improve its software.

All of our reports to the OpenVPN developers may be found in the OpenVPN bug tracker on SourceForge [19]. Some of these reports were temporarily marked confidential, to provide time for OpenVPN users to upgrade to new versions of OpenVPN and to avoid aiding attackers who might wish to attack OpenVPN.

- We found several places where the documentation did not fully explain the security hazards associated with certain configurations.

  For instance, OpenVPN supports a `chroot` configuration option to enable jailing OpenVPN within a subdirectory of the directory hierarchy. However, this configuration option is not secure unless one also specifies in the configuration file that OpenVPN should drop privileges by changing its uid and gid, using the `user` and `group` configuration directives. Consequently, using the `chroot` configuration directive without also setting the `user` and `group` directives appropriately would likely lead to a false sense of security. The documentation did not explain this hazard, and the code did not check for this pitfall.

  We also identified an undocumented security hazard associated with the parameters passed to scripts. The user can define various scripts that OpenVPN will execute when various events occur. OpenVPN passes various arguments to these scripts describing the nature of the event that occurred. Under some conditions, an adversary

could potentially cause these arguments to start with a dash ('–'); if this were misinterpreted by the script as a command-line argument, it could have security consequences. This hazard is obscure and arguably unimportant, but it probably should have been documented.

We also identified an undocumented potential hazard associated with dynamic loading of plugins.

We reported these to the OpenVPN developers and they reported that they have revised the code to address these hazards.

All in all, OpenVPN appears to be a reasonable choice to defend against outsider attacks, given the likely scale of the November 2008 election. OpenVPN is not perfect: we should be prepared for the possibility that vulnerabilities could be discovered in OpenVPN at some point in the future.

We can suggest several steps that may help mitigate some of the weaknesses we found. We stop short of strongly recommending that these steps be taken, because we have not analyzed whether these steps would have other side effects, and because enabling these mitigations might require re-testing the voting system to ensure that these changes do not negatively impact the reliability of the voting system. These mitigations would need to be evaluated carefully before adoption. Nonetheless, we describe these possible mitigations, in case they are helpful:

- Scytl should consider enabling the `tls-auth` configuration option. This option directs OpenVPN to apply a symmetric-key message authentication code (MAC) to all packets, and most notably, to the control packets that are used to initiate a new connection. Any packet not containing a valid MAC will be dropped. This MAC would not be sufficient on its own, but it provides an additional layer of security against denial-of-service attacks and a certain degree of hardening against some kinds of remote exploits.

  The disadvantage of `tls-auth` is that it requires one to configure a symmetric key at every client and every server, which introduces a key management burden. Thus, the easiest way to deploy `tls-auth` may be to generate a single symmetric key and use it for all clients and servers in a particular election, changing it from election to election. Normally, reuse of a cryptographic key in this way would represent poor key management practice, but for the specific purpose of the ODBP we believe a single shared key would represent a reasonable tradeoff between usability and protection against denial-of-service attacks.

  The OpenVPN manual describes the purpose of the `tls-auth` configuration option well:

  > In a nutshell, –tls-auth enables a kind of "HMAC firewall" on OpenVPN's TCP/UDP port, where TLS control channel packets bearing an incorrect HMAC signature can be dropped immediately without response. [...]
  >
  > –tls-auth is recommended when you are running OpenVPN in a mode where it is listening for packets from any IP address, such as when –remote is not specified, or –remote is specified with –float.
  >
  > The rationale for this feature is as follows. TLS requires a multi-packet exchange before it is able to authenticate a peer. During this time before authentication, OpenVPN is allocating resources (memory and CPU) to this potential peer. The potential peer is also exposing many parts of OpenVPN and the OpenSSL library to the packets it is sending. Most successful network attacks today seek to either exploit bugs in programs (such as buffer overflow attacks) or force a program to consume so many resources that it becomes unusable. Of course the first line of defense is always to produce clean, well-audited code. OpenVPN has been written with buffer overflow attack prevention as a top priority. But as history has shown, many of the most widely used network applications have, from time to time, fallen to buffer overflow attacks.
  >
  > So as a second line of defense, OpenVPN offers this special layer of authentication on top of the TLS control channel so that every packet on the control channel is authenticated by an HMAC signature and a unique ID for replay protection. This signature will also help protect against DoS (Denial of Service) attacks. An important rule of thumb in reducing vulnerability to DoS attacks is to minimize the amount of resources a potential, but as yet unauthenticated, client is able to consume.
  >
  > –tls-auth does this by signing every TLS control channel packet with an HMAC signature, including packets which are sent before the TLS level has had a chance to authenticate the peer. The result is that packets without the correct signature can be dropped immediately upon reception, before they have a chance to consume additional system resources such as by initiating a TLS handshake. –tls-auth can be strengthened by adding the –replay-persist option which will keep OpenVPNs replay protection state in a file so that it is not lost across restarts.

It should be emphasized that this feature is optional and that the passphrase/key file used with –tls-auth gives a peer nothing more than the power to initiate a TLS handshake. It is not used to encrypt or authenticate any tunnel data.

Note that `tls-auth` does little or nothing to defend against insiders. In general, the more parties with access to the `tls-auth` key, the less useful it becomes. Consequently, the utility of `tls-auth` may be degraded somewhat if the number of overseas polling places increases. At the same time, the key management burdens increase as the number of overseas polling places increase. Nonetheless, for the number of polling places envisioned in November 2008, we consider that this tradeoff might be worthwhile.

- Scytl could consider using a stable release of OpenVPN, rather than a beta release. This may not contain some of the most recent features, but the code may be better tested.

- Scytl could consider using a more recent version of OpenVPN that contains fixes to the code to address the poor coding practices we found. However, at present this requires using a beta version of OpenVPN rather than a stable release.

  This conundrum will be easier to manage once the current beta version becomes stable enough to be released as a stable release. At that point, Scytl can use the stable release (instead of future beta versions) and gain the advantage of code quality improvements.

  In any case, Scytl should probably avoid using 2.1_rc7 or other versions that contain a known vulnerability (the vulnerability listed above).

- On Linux platforms Scytl might consider compiling and running OpenVPN in a way that takes advantage of gcc and glibc's defenses against double-free and buffer overrun vulnerabilities. In particular, Scytl might consider compiling OpenVPN using gcc's `-fstack-protector-all` compilation option, and might consider setting the environment variable `MALLOC_CHECK_=2` to enable glibc's stack and heap memory integrity checks. However, if these steps are taken, OpenVPN should be tested to ensure that these mitigations do not impair OpenVPN's reliability: these mitigations rely upon terminating the OpenVPN process if evidence of an attack or vulnerability is detected at run time, so these mitigations could conceivably have a negative impact on reliability if OpenVPN contains bugs or other poor coding practices.

- On Unix platforms Scytl might consider using the `chroot`, `user`, and `group` options to run OpenVPN in privilege separation mode by running OpenVPN in a chroot jail. It is important to use the `user`, and `group` options to set the uid and gid to an unprivileged user that cannot escape the jail, and to configure the chroot filesystem correctly. If it is not configured properly, this will lead to a false sense of security. However, if configured appropriately, this might help reduce the damage that could be caused by a vulnerability in OpenVPN.

  The latest versions of OpenVPN allow running OpenVPN in an unprivileged setting, on Linux platforms only. This may add further security.

- Scytl should consider keeping the root CA key on a separate, air-gapped machine that is not connected to any network, if this is not already the existing practice.

Because we did not have access to the OpenVPN configuration files, we did not check whether Scytl is already applying any of these mitigations.

Action: Scytl should evaluate the mitigations listed here to determine whether any of them are beneficial.

4.3.9.3 Finding: The way that the OpenVPN software is used represents good practice and appears likely to prevent outsider attacks on privacy or integrity

The Pnyx.core ODBP 1.0 voting system uses OpenVPN to establish a secure channel between voting clients and the proxy service. Based on our understanding of how OpenVPN is used, it appears that OpenVPN is used properly. A separate public-key certificate is used for each voting client, so each client-proxy connection is separately keyed. Assuming OpenVPN is properly configured, this architecture seems likely to be effective at preventing outsiders from eavesdropping upon or tampering with electronic communications between the voting client and proxy service, while those communications are in transit.

One caveat: we are relying upon Scytl's description of how they intend to configure OpenVPN in the election. We have not examined the OpenVPN configuration files ourself. There are two relevant configuration files: the configuration file on the proxy service machine, and the configuration files on the voting client. We were not provided with any configuration files for the proxy service or voting service machines, so it was not possible for us to examine them. We were provided with a LiveCD for the voting client, and we presume that this LiveCD contains the OpenVPN configuration files on it, but because we have not been able to boot the LiveCD on any of our computers, we have not been able to check this or to examine the contents of that configuration file. Similarly, we have not examined the processes used to generate keys for OpenVPN, and we did not find documentation that describes in detail how OpenVPN should be configured. Of course, errors in these configuration files can introduce unexpected security vulnerabilities; since we did not examine those configuration files for ourself, our analysis assumes that OpenVPN is configured properly.

Action: Scytl should ensure that the necessary configuration steps are fully documented, so that election officials can carry them out on their own without Scytl's help. Future security reviews of the Pnyx.core ODBP voting system may wish to examine the OpenVPN configuration. It would be helpful if future submissions of this voting system for certification included all relevant configuration files.

A second caveat: Our analysis is predicated upon the kind of threats that we envision might be plausible for an election like the one that may occur this November 2008. It is expected that the Pnyx.core ODBP 1.0 voting system will be used with only a few overseas polling locations and may service only hundreds or thousands of voters, and this reduces the reliance upon OpenVPN. We would suggest that our analysis of OpenVPN and communications security be revisited if there is a dramatic increase in the number of polling places and voters. The security threats that must be addressed may change at scale. For instance, more analysis regarding the possibility of attacks by poll workers (who may have access to OpenVPN keying material) might be warranted if the number of polling sites is significantly increased.

Action: If in the future the Pnyx.core ODBP voting system is used on a significantly wider scale than what is expected in November 2008, election officials might want to consider re-examining OpenVPN.

Chapter 5

Conclusion

This report describes the results of our analysis of the Pnyx.core ODBP 1.0 remote voting software. Our findings are mixed: the report contains both good news and bad news for election officials considering using this system.

A major finding of our analysis involves the role of the Voter Choice Records (VCR) in the election system. While not downplaying the need for carefully designed and implemented procedures throughout the election system, the opinion of the team was that the procedures for handling, auditing, and recounting the VCRs will be pivotal in ensuring the integrity of the election, even if all other procedures are properly designed and followed. The detailed VCR procedures were not provided to us at the time of our review, but we can envision several possible scenarios:

- *Best case:* If voters check the accuracy of Voter Choice Records carefully, and if chain of custody for Voter Choice Records is protected sufficiently, and if Voter Choice Records are subject to a routine 100% manual count after every election, and if any discrepancies discovered between the Voter Choice Records and electronic records are handled appropriately, then as best we can tell, the Pnyx.core ODBP 1.0 system appears to address or partially address most, if not all, of the threats it may face.
- *Worst case:* If few voters check the accuracy of Voter Choice Records, or if the Voter Choice Records are never recounted and never audited, or if the chain of custody is not protected, or if discrepancies between the paper and electronic records are not handled appropriately, then the Pnyx.core ODBP 1.0 system may be subject to attacks that could potentially compromise the integrity of votes cast on the Pnyx.core ODBP 1.0 system or could potentially enable manipulation of the final tally of votes cast on the Pnyx.core ODBP 1.0 system. In this case, undetected software failures or undetected attacks could potentially cause significant, undetected shifts to the final vote tallies.
- *Otherwise:* There are many intermediate scenarios in between these two extremes. Analysis of such scenarios would depend heavily upon their details.

Another major finding of our analysis is that even though the Pnyx.core ODBP 1.0 system advertises cryptographic voter receipts that voters can use to verify their votes, the value of these receipts is limited. If the mixing service and voting laptop are not infected with malicious software and are implemented correctly, then these receipts do provide a mechanism that enables voters to check that their ballot was received and accurately counted by the election system—an improvement upon typical postal voting systems. But the receipts' contents are completely unrelated to the choices made by voters on their ballots. Thus, if either the mixing service or voting laptop contains malicious software or (relevant) software faults, then the receipts do not guarantee that votes were recorded or counted accurately. Manual recounts or audits of the Voter Choice Records are the only reliable way we know of to detect such errors in this system.

We are pleased to report that the Pnyx.core ODBP 1.0 system offers some improvements over previously proposed or fielded e-voting systems for overseas voters. For example, the system provides ballot storage redundancy (through electronic and paper copies) that is not present in current vote-by-mail systems. Additionally, the use of supervised polling locations and voting kiosks whose hardware and software is under the control of election officials improves on previous systems by mitigating outsider threats. Likewise, the ability to print Voter Choice Records provides election officials with the capability of performing manual recounts that are independent of software, improving upon the integrity and transparency of previous systems. Furthermore, the use of cryptography to protect all communication

channels improves on voting by cleartext email or fax. Finally, the system provides strong protection against coercion and improper influence, due to its use of polling places supervised by election workers and private booths for voters. This improves upon the confidentiality of previous systems.

One strength of the Pnyx.core ODBP 1.0 software is that it appears to provide a reasonable degree of protection against many kinds of outsider attacks, including Internet-based attacks, which could be mounted by adversaries located far away from the voting system. One weakness of the Pnyx.core ODBP 1.0 software is that the software mechanisms appear to provide little protection against insider malfeasance or actions by malicious insiders who exceed their authority; instead, these threats must be addressed in this system by other, non-technical mechanisms, such as audits or manual counts of the Voter Choice Records.

We were unable to provide definitive, final answers to many of the important and central questions posed to us by the State of Florida. This was not due to a lack of effort; rather, it reflects limits on current human ability to engineer computing systems so that they will behave in predictable ways. It is currently very difficult and costly to achieve a reasonable level of confidence that complex software is free of bugs, defects, flaws, or vulnerabilities that could affect its ability to perform as expected. The Pnyx.core ODBP 1.0 voting system contains hundreds of thousands of lines of source code, making it complex and beyond our ability to scrutinize every line of code carefully. Yet a single flaw in any one line of code could potentially have severe consequences. It is vanishingly unlikely that any system of this size is completely free from bugs, implementation defects, and design flaws—indeed, we found a number of such shortcomings. As per the scope of work, we made no attempt to assess or interpret the severity of these vulnerabilities and we defer that judgement to the Florida Division of Elections based upon the technical data we have provided. In addition, because we conducted our work primarily in our spare time, over weekends and evenings, on an unpaid basis, our ability to analyze detailed technical issues was limited. In the end, resource constraints forced us to focus primarily on the high-level design of the system and the implications of the architectural choices made by Scytl. We performed only spotchecks on a tiny and carefully chosen subset of the source code of the system, leaving implementation quality issues largely unexamined.

This has unfortunate implications for our ability to make definitive statements about the security of the Pnyx.core ODBP 1.0 voting system. We have attempted to apply our best professional judgement, but in many cases the conclusions we have drawn are limited or clouded by caveats and uncertainty. Moreover, the security of a voting system depends not just upon technology, but also upon procedures, people, and other non-technical elements; in this review, we were asked to focus primarily on the software. We consider it likely that a more thorough review of the system would identify other issues—perhaps minor issues, perhaps major problems or vulnerabilities—that we overlooked.

For these reasons, we recommend that our conclusions should be taken as provisional and subject to revision as more information becomes available. If the Pnyx.core ODBP 1.0 voting system is adopted and if its use is expanded beyond the limited extent envisioned for November 2008, we recommend that a follow-on study be commissioned to examine the system in further detail.

Despite the shortcomings of this report, we hope that it will be useful to election officials and others as they evaluate this voting system.

Acknowledgments

As part of our work we used an automated source code analysis tool, Fortify Source Code Analysis (SCA), made by Fortify Software. Fortify Software donated the tool to us free of charge for use on this project and we thank them for their contribution. We note that one member of the team (Wagner) is on Fortify Software's Technical Advisory Board.

We also acknowledge the contributions of the ACCURATE Center in this review. We note that the lead investigator (Wagner) is an ACCURATE member and that the ACCURATE Center provided hardware resources that significantly increased the team's capabilities.

The ASSERT Center at the University of Alaska Fairbanks provided support for this review, primarily in the form of hardware and software for investigator Hay.

Finally, the University of South Alabama School of Computer and Information Sciences provided substantial hardware and administrative resources resources for the team, including setting up and funding team conference calls.

We thank Vern Paxson for helpful advice on how to measure the cost of denial-of-service attacks and Nick Weaver for helpful advice on laptop security.

We are grateful to FLDoS, Scytl, and Carol Paquette for providing comments on a draft version of this report. We believe their comments greatly improved the final product. Of course, any errors or inaccuracies in this report are the responsibility of the authors alone.

Bibliography

[1] Ben Adida and Douglas Wikstrm. How to shuffle in public. In *TCC 2007*, pages 555–574, 2007.

[2] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management—part 1: General (revised), March 2007. SP-800-57.

[3] Josh Daniel Cohen Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, September 1987.

[4] William E. Burr, Donna F. Dodson, and W. Timothy Polk. Electronic authentication guideline, April 2006. SP-800-63.

[5] Tom Cargill. Exception handling: A false sense of security, November 1994. `http://www.informit.com/content/images/020163371x/supplements/Exception_Handling_Article.html`.

[6] CERT C Programming Language Secure Coding Standard, 2008. `https://www.securecoding.cert.org/confluence/display/seccode/CERT+Secure+Coding+Standards`.

[7] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, Jan-Feb 2004.

[8] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *Electronic Voting Technology Workshop*, 2008.

[9] Operation Bravo Foundation. Okaloosa distance balloting pilot: Procedures and system description for secure remote electronic transmission of ballots for overseas civilian and military voters, December 2007.

[10] J. Franklin, V. Paxson, A. Perrig, and S. Savage. An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants. In *Proceedings of ACM CCS 2007*, October 2007.

[11] Michael Howard and Steve Lipner. *The Secure Development Lifecycle*. Microsoft Press, 2006.

[12] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proc. of USENIX Security Symposium*, pages 339–353, August 2002.

[13] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proc. ACM Workshop on Privacy in the Electronic Society*, pages 61–70, 2005.

[14] Juhani Karhumäki and Tommi Meskanen. Audit report on pilot electronic voting in municipal elections, June 2008. `http://www.vaalit.fi/uploads/5bq7gb9t01z.pdf`, linked from `http://www.vaalit.fi/14173.ht`.

[15] Gavin Lowe. An attack on the Needham-Schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–136, November 1995.

[16] Steve McConnell. *Code Complete*. Microsoft, second edition, 2004.

[17] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[18] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proc. ACM Conference on Computer and Communications Security*, pages 116–125, 2001.

[19] OpenVPN Bug Tracker. `http://sourceforge.net/tracker/?group_id=48978&atid=454719`.

[20] OpenVPN Downloads. `http://openvpn.net/index.php/downloads.html` (downloaded August 28, 2008).

[21] Ronald L. Rivest and John P. Wack. On the notion of "software independence" in voting systems, July 2006. `http://vote.nist.gov/SI-in-voting.pdf`.

[22] Peter Y. A. Ryan and Steve A. Schneider. Prêt à voter with re-encryption mixes. In *ESORICS*, pages 313–326, 2006.

[23] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme—A practical solution to the implementation of a voting booth. In *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 393–403, May 1995.

[24] Bruce Schneier. *Applied Cryptography*. Wiley, 1996.

[25] Scytl. Pnyx.core: Functional description, August 2004. Version 1.7.1c.

[26] Scytl. Pnyx.core: The key to enabling reliable electronic elections, December 2005.

[27] Scytl. Response to Bugzilla 109: Voting protocol, third round, July 2008.

[28] Scytl. Response to Bugzilla 121: Network connectivity of authentication laptop, July 2008. Attachment 100.

[29] Scytl. Technical Data Package: Pnyx.core Release ODBP 1.0, 2008.

[30] David Wagner. [2015709] configure –enable-iproute2 + –pull is a security hole?, July 2008. Report #2015709 on the OpenVPN bug tracker, `http://sourceforge.net/tracker/index.php?func=detail&aid=2015709&group_id=48978&atid=454719`.

CHAPTER 6: Scytl's comments on the review report of Pnyx.core ODBP 1.0

Table of Contents

1. Introduction

Scytl would like to thank the review team led by the SAIT Laboratory for all the time and effort that they have put into this independent expert review of Pnyx.core ODBP 1.0. As explained by the review team in the report, the review process included significant interaction between the reviewers and Scytl, and the reviewers "felt that this was positive and could serve as a helpful model for future reviews"[1]. We fully agree with this statement since we strongly believe that both transparency and vendor collaboration are critical in the review of any voting system.

Pnyx.core ODBP 1.0 is a secure remote electronic voting platform custom-tailored to meet the specific requirements for the Okaloosa Distance Balloting Project (ODBP) and the election law in the State of Florida. The development of Pnyx.core ODBP 1.0 began in 2007 based on Pnyx.core 1.7.1, a security platform for remote electronic voting that allows building upon a common security framework multiple voting systems in order to meet the specific legal and electoral requirements of each country. Based on this Pnyx.core platform, multiple governments worldwide have successfully implemented Internet voting projects in countries such as the United Kingdom, Switzerland, Finland, Argentina, Spain and the Philippines.

Pnyx.core is a product in continuous evolution. Pnyx.core 1.7.1 was the version commercially-available at the time that Scytl was selected to provide the Internet voting technology for the ODBP project in 2007 and, therefore, was the version used to develop Pnyx.core ODBP 1.0. During the first quarter of 2008, Scytl released Pnyx.core 2.0 which represented a significant improvement over the 1.7.1 version with additional security and functional features. A 3.0 version is currently under development and is expected to be released during 2009.

This Chapter 6 contains Scytl's comments on the review team's findings. In section 2, we explain the critical aspects that in our opinion must be considered when evaluating the security of any voting system and we also provide comments that can be generally applied to all the findings in the report. In section 3 we address each of the specific findings in the report, providing an explanation of Scytl's position for each of those findings. Finally, section 4 summarizes our overall conclusions of the review report.

[1] See Chapter 1 of the review report.

2. General comments

In section 2.1 we provide an overview of the aspects that must be considered when evaluating the security of any voting system and how they apply to the Pnyx.core ODBP 1.0 review. This introduction provides the context for section 2.2, where we provide comments that can be generally applied to the findings and statements in the review report.

2.1. General considerations when evaluating electoral security

Pnyx.core ODBP 1.0 is a software voting solution based on Pnyx.core 1.7.1 that implements cryptography at the application layer (i.e., implements specialized cryptographic techniques to provide security guarantees to the voting process). These specialized security measures go beyond generic security measures such as firewalls, Virtual Private Networks and the like. Pnyx.core ODBP 1.0 also uses these generic measures, but the underlying basis of the system's security is our new and specialized security layer.

For any voting (or other mission-critical) system, the technology employed is key but is not sufficient in itself to guarantee the security of an election. Technology must be complemented with other components:

- Procedures detailing who can do what on which component and under which circumstances and conditions.
- Procedures regulating the interaction of individuals with the voting system at different moments and defining the individuals who can perform different tasks.
- A pre-election auditing process to ensure that the voting system (the technology) behaves correctly and provides the required security levels for the election before it is implemented. A successful audit process usually leads to a certification of the system being audited.
- An on-going and post-election auditing process to verify that the voting system is behaving correctly during an election and after it has ended.

All voting systems (electronic and non-electronic) have some components that must be trusted to ensure a fair election. These trusted components must be protected through logical, physical and procedural security measures in order to have a fair and secure election. In an ideal world no trust would be placed on any component, but, as noted in the review report 4.2.5, this ideal is seldom achievable in practice. For example, in paper-based elections, there are components such as the ballot boxes that must be kept physically secured in order to guarantee the integrity of the election. Furthermore, all voting systems rely on trusted personnel. In paper-based elections, voters must trust that election officials will correctly tally

the ballots cast or that they will not impersonate voters by placing ballots on their behalf in the ballot box.

Pnyx.core ODBP 1.0 implements a cryptographic protocol that reduces the number of trusted components to three:

- The Voting Laptop, used by the voters to cast their ballots.
- The Mixing Service, used for decrypting the ballots.
- The Key Generation service, which generates all the cryptographic keys and digital certificates (a local Certification Authority).

The software that runs on the Voting Laptop to display the different voting options and encrypt the ballots is digitally signed by the Okaloosa Election Office. Any modification to this software will be detected by the Voting Laptop.

Both the Mixing Service and the Key Generation Service operate on the Mixing Server, a standalone, physically-protected computer controlled by Okaloosa Election Officials. These services, and the rest of the Pnyx.core ODBP 1.0 software components, have been compiled, installed and tested by the Florida Division of Elections (FLDoE) on the Mixing Server, and will be validated against the certified version of the voting system before being used in the election, to ensure that they have not been changed in any way.

These trusted components can be given strong protection by the implementation of appropriate security procedures, similar to those used for trusted components of other types of voting systems.

Pnyx.core ODBP 1.0 requires placing trust in the same individuals who must be trusted for the proper and secure operation of any type of voting system, whether it is paper-based or electronic:

- Poll workers (Kiosk Officials), who must behave properly when authenticating voters and employing the Voting Laptops at each Kiosk site.
- Okaloosa Elections Officials, who configure the voting system and digitally sign the Voting Laptop software.
- Okaloosa Canvassing Board, who are responsible for decrypting the ballots and for generating the results, always in a collaborative manner. A single member of the Canvassing Board cannot act alone.
- Florida Division of Elections staff, who tested the system and are responsible for deciding whether to certify it
- Election auditors, who will verify the election tabulation results.

The above-mentioned individuals are the same individuals who must be trusted in any kind of election, may it use paper ballots or electronic voting systems. Because the system is configured and administered entirely by Okaloosa elections staff, it is not necessary to place any trust in the technicians who may assist with the installation of the system.

2.2. Overview of Scytl's comments on the report

After a careful review of the report, we have found several statements and findings that we are in agreement with and several others where we disagree.

In the first group, we would include several conclusions that can be inferred from the report:

- No significant flaws, bugs or malware were found in the reviewed software components and security architecture of the system
- No significant weaknesses were discovered in the cryptographic algorithms
- The reviewers considered the audited system to be superior to the alternative remote voting systems that it is intended to replace (i.e., postal voting, e-mail voting and fax voting) in terms of criteria such as integrity of the votes, voter privacy and election results auditability.

In the second group, we have the findings and statements where we are in disagreement with the reviewers. In general, these findings and statements can be placed in three categories:

- Findings or statements related to missing documentation (including software related documentation or documents related to procedures).
- Findings or statements related to the risk of malware operating/running on trusted components of the voting system.
- Findings or statements related to the risk of trusted individuals behaving maliciously.

Each of these categories is discussed below.

2.2.1. Missing documentation

We have to say that we regret any miscommunication that might have occurred between the reviewers and Scytl. Scytl was completely transparent and provided the reviewers with ALL the information and documentation that the reviewers and/or FLDoE requested. Scytl also answered all the questions from the review team in a prompt and accurate manner, even during weekends and holidays, as we knew the reviewers were working on their free time and required our responses as soon as possible. We must note that we were never requested to

provide the documentation mentioned in sections 4.1.1, 4.1.1.1, 4.1.1.3, 4.1.1.4 and 4.2.1.6 of the review report. If we had known this information was needed, we would have supplied it immediately to the review team.

Similarly, section 4.1.3.1 of the review report states that the review team was able to build the application but not to deploy and run it. Since the FLDoE had stated that the review team was not going to run any functional testing, they were not provided with the documentation explaining how to deploy and operate Pnyx.core ODBP 1.0. The FLDoE executed the system functional testing and had all the required documentation needed to successfully perform this testing. The review team could have requested this information from FLDoE or from Scytl.

2.2.2. Risks related to malware running on trusted components

There is a series of statements and findings (sections 4.1.2.1, 4.2.1.1, 4.2.1.2, 4.2.1.3, 4.2.2, 4.2.2.2, 4.2.2.3, 4.2.3.1, 4.2.5.1, 4.2.5.2, 4.2.5.3, 4.3.1.1, 4.3.4.3 and 4.3.6.2 in the review report) related to vulnerabilities of Pnyx.core ODBP 1.0 when malware is injected into some component of the voting system, or the certified software[2] is replaced with malware, and similar situations.

Of course, if the correct software is replaced by bogus software, then the system will behave incorrectly and will not be able to provide the security measures required for the election. This happens with any software application, may it be used in an election or for controlling the launching of nuclear missiles. The only practical countermeasure to prevent this risk from happening is to employ measures that protect the certified software from being modified/replaced, and the system from being injected with malware that can affect the election. These measures are the procedures and external tools employed by the certifiers of the system (in this case FLDoE) and the electoral authorities deploying the system (in this case Okaloosa Elections Office). I*n* the ODBP project the following security measures are being implemented to mitigate the previously mentioned risks:

- All the software components of Pnyx.core ODBP 1.0 have been reviewed, audited and tested by FLDoE.
- The certified system will be hashed by FLDoE and directly supplied to Okaloosa Elections Office. This includes all the binaries of the voting system and the Live CD used in the Voting Laptops.

2

- Okaloosa Elections Office will also digitally sign the software used on the Voting Laptops.
- All the hardware elements used in the project will be hardened and tested under the supervision of Okaloosa elections officials prior to their delivery to the kiosk sites. They will be sealed with tamper-evident seals. All unused ports (e.g. USB ports not employed by the system) will similarly be sealed.
- The Mixing Server will be stored in a secure vault under the surveillance and control of Okaloosa Elections Office.

With these security measures in place, the likelihood of exploitation of any of the vulnerabilities listed above is extremely remote in the ODBP project.

2.2.3. Risks related to trusted individuals behaving maliciously

There is another series of statements and findings (sections 4.1.2.1, 4.2.1, 4.2.1.4, 4.2.1.5, 4.2.2, 4.2.2.1, 4.2.2.2, 4.2.2.3, 4.2.3.1 and 4.3.6.1 in the review report) that are related to trusted individuals who operate or interact with the voting system behaving maliciously, including poll workers (Kiosk Officials), Canvassing Board members, Florida Division of Elections staff and Okaloosa Election officials. This series of comments overlaps somewhat with the statements and findings related to malware running on trusted components discussed in section 2.2.2.

The Pnyx.core ODBP 1.0 includes several measures to ensure that no single individual can affect the integrity of the election and the privacy of the voters (for example, by requiring the collaboration of multiple members of the Canvassing Board to carry out certain critical tasks). However, if the entire Canvassing Board acts dishonestly or poll workers collude to behave maliciously, there are very few things any voting system, be it electronic or not, can do to counter this. The operation of Pnyx.core ODBP 1.0 splits responsibilities and requires multiple parties to perform critical tasks to prevent any one individual from tampering with the system. In addition, system logs can be used to identify who performed critical transactions on the system.

Pnyx.core ODBP 1.0 also includes effective measures to prevent technicians from accessing the voting system and affecting the outcome of the election, and puts the control of the election entirely in the hands of the Okaloosa elections officials and the Canvassing Board (as it happens with paper-based elections).

2.2.4. General Observation

As a general observation, we believe the review report lacks an important element: the assessment of the likelihood of the exploitation of potential vulnerabilities. While the report usually assesses the impact of a successful attack on a potential vulnerability, very rarely does it assess the feasibility or degree of difficulty involved in carrying out a successful attack. Stating potential vulnerabilities without quantifying the probability of their occurrence provides an incomplete picture of the security and integrity of the system and can be misleading for readers of the report. There is no information to support a determination of which possible vulnerabilities are the most critical or to what extent mitigation measures are necessary to ensure that an election can be carried out securely. We will be carefully evaluating these findings and recommendations as we complete the detailed operating procedures for the ODBP project to ensure that all appropriate protections are in place.

3. Specific comments on the findings of the review report

This section contains Scytl's comments on specific statements and findings included in the review report. In order to facilitate the reading of our comments, we have kept the same section numbering as the review report. The sections/findings of the review report we do not have comments for are not included in this section.

3 System Overview

3.1 Architecture

Section 3.1 in the review report contains some inaccuracies regarding Scytl's role during the execution of the election we would like to point out. The reviewers' report states that:

County election officials. These include individuals who are responsible for administering and operating the Mixing Service (see below).

Scytl employees. It is envisioned that in the November 2008 election, some Scytl employees will be responsible for administering the Voting Proxy and Voting Service software (see below)."

In fact, county election officials will be trained before the election so they are capable of operating all the voting system's components, including the ones hosted at Scytl's data center, without the intervention of Scytl. Only FLDoE certified Okaloosa elections officials will be allowed to operate Pnyx.core ODBP 1.0 components. Scytl's only role during the election will be that of providing immediate support to staff in Okaloosa and ensure that the Okaloosa election officials carry out the operations correctly. In addition to providing remote support to all Absentee Voting Kiosk Sites during the voting period, and monitoring the performance of the data center.

3.2 Protocol Overview

3.2.1 High-level view

Section 3.2.1 in the review report contains another inaccuracy. The first bullet point states "…using the public key of the mixing service…" when it should state "…using the public key of the Canvassing Board…".

3.2.2 Key Management

Section 3.2. in the review report contains another small inaccuracy. The third bullet point states "…the mixing service's decryption key…" when it should state "…the Canvassing Board's decryption key…".

3.3 Protocol Details

3.3.7 Addendum

Regarding this addendum, we must say that the session id from the printed 'Voter Choice Record' is not part of the Pnyx.core cryptographic protocol, and therefore it is not included in the protocol specification. This is a specific requirement for Pnyx.core ODBP 1.0, and the session id from the VCR is stored inside the encrypted ballot. Every customer has different requirements, and the Pnyx.core cryptographic protocol is very flexible as more information can be included inside the ballot in addition to the voting options. The component of Pnyx.core ODBP that is responsible for inserting this information inside the ballot before encryption is the ODBP Voting Kiosk.

4 Findings

4.1 Software Quality Analysis

4.1.1 Is accurate complete life-cycle documentation available?

As stated in section 2, Scytl provided all the information and documentation that was requested by the reviewers. We regret that due to miscommunication the review team could not evaluate everything they wanted.

4.1.1.1 Finding: The cryptographic protocol specification needs improvement

As stated in this finding, the protocol is described in [24, x5], although maybe does not fulfill the detailed formatting of an academic document. Our format uses [24] as a reference, while the proposed format is closer to [17]. Nevertheless, we are grateful to the review team for detecting some formatting issues and we will review the protocol specification with the notation and recommendations you suggest.

4.1.1.2 Finding: Test plan and results documents were not included in the material submitted for evaluation

As previously stated, ALL these documents were available but never requested from Scytl. The system has been thoroughly tested following a detailed test plan by both Scytl and the FLDoE

during the functional testing. This included a nightly build with an automated functional, unit and integration test suite and corresponding plans. Test results have all been documented accordingly.

4.1.1.3 Finding: Documentation of the threat model and security architecture was not included in the material submitted for evaluation

Once again these documents were never requested from Scytl. Documentation of the threat model, system architecture and design rationale, does exist and addresses, among others, the issues discussed in this finding.

4.1.1.4 Finding: No assurance document providing convincing evidence that the voting system meets the requirements, under specified conditions, was included in the material submitted for evaluation

As commented before these documents were never requested from Scytl. Pnyx.core ODBP 1.0 is based on Pnyx.core 1.7.1, which can be used to run elections worldwide. Therefore, assurance documentation is adapted for each project depending on its specific requirements. In the ODBP project, Scytl worked closely with the FLDoE to define the security requirements of the voting process, and assess the security features of Pnyx.core ODBP 1.0 in regards to the certification process. The role of the FLDoE has been to evaluate that Pnyx.core ODBP 1.0 fulfills such requirements.

4.1.1.5 Finding: System documentation contained occasional inaccuracies

As stated by the review team, Pnyx.core ODBP 1.0 is based on Pnyx.core 1.7.1, which is the 'core' of all Scytl's voting systems employed in multiple countries worldwide. Therefore, the system includes several options some of which are not used in the ODBP project, but that are included in the generic documentation regarding Pnyx.core 1.7.1c that was submitted to FLDoE. We acknowledge that these discrepancies might have caused some confusions and misunderstandings. However, the rest of the documentation related to Pnyx.core ODBP 1.0, as well as the Q&A sessions and conference calls that were held with the review team, tried to clearly state such differences in the documentation in order to avoid any possible confusion.

4.1.2 Is the architecture of the software well-suited to the requirements of the voting system?

We understand that a networked environment is prone to more attacks that a non-networked environment, but Pnyx.core ODBP 1.0 implements several measures that mitigate these risks, as the review report points out when stating that the system resists fairly well against external attacks.

4.1.2.1 Finding: The Voting Client relies on software that is downloaded across the network at run-time, and only stored in volatile-memory

These concerns are mitigated because the Voting Client application is digitally signed by FLDoE and/or Okaloosa Elections Office after certification, and the voting kiosk will only accept this signed application. Therefore, if Scytl modified the Voting Client after the certification, the voting kiosk will fail to validate the application's digital signature and will not run it. Therefore, the desirable audit commented above is feasible even when the Voting Client is downloaded over a network. Of course, as stated in section 2, if the Voting Kiosk or the signing key has been compromised, then it does not matter whether the Voting Client is downloaded or not, as it can be modified. For this situation to occur trusted individuals, such as FLDoE staff and/or poll workers, would have to behave dishonestly.

4.1.3 Is the target software build environment complete and properly documented?

4.1.3.1 Finding: The resources necessary to deploy a functional voting system were not available to the review team during the audit period

In the first place, the source code of the whole application, including the cardlet source code, was provided to FLDoE. More specifically, this source code and associated documentation were delivered on July 17th, 2008, after we detected it was missing from the previously submitted batch of source code and documents due to questions posted by the review team. We do not understand why this information supposedly never reached the review team.

In the second place, Scytl was notified by FLDoE that the review team was not responsible for running any functional testing. Therefore, we prepared the material in order for the review team to be able to compile and review the source code, not to deploy the compiled system and run test elections. In fact this task was performed by FLDoE staff, which successfully complied, built and deployed Pnyx.core ODBP 1.0 for several elections during their functional testing. We understand that the results of this functional testing will be publicly available when this report is published.

4.1.6 Are the programming language(s) that are employed well-suited to their use?

As response to the statement or question regarding why Pnyx.core ODBP 1.0 uses C++, we have to answer that this is due to performance reasons. While it is true that for the ODBP project a limited amount votes will be cast, and performance is not an issue, the C++ code is inside Pnyx.core 1.7.1, which has been designed to handle very large amounts of votes. We must remind that Pnyx.core 1.7.1 main purpose is to allow Internet voting at large scale.

4.1.7 Does target software follow software engineering practices that are well-suited to avoid or minimize the risk of security vulnerability?

We have only one comment here. The report states "…the lack of detailed cryptographic specifications…", but as mentioned and stated in Finding 4.1.1.1, this statement is misleading, as there is a detailed cryptographic specification.

4.1.7.1 Finding: Insufficient and incomplete use of static analysis tools

As part of an organization wide quality program, Scytl continuously improves its software development methodologies, processes and tools. We are currently assessing such tools as the ones mentioned in this finding, in addition to implementing others in our nightly build process.

4.1.7.2 Finding: Changes to the source code do not appear to result in changes to the external software version number

Scytl understands the importance of software version numbering, and fully endorses its use. In regards to the described finding, we must say the following:

- We have discussed this several times with the review team, explaining the rationale behind changing software versions from an internal point of view and from a 'commercial' point of view, but it seems each party has its own vision on the subject, which is of course legitimate.
- Scytl follows strict version control processes and uses industry standard tools for ensuring it, such as subversion and BugZilla (connected using SCMbug), and previously CVS.
- As many software companies, Scytl uses different 'commercial' version numbers (known by the customers) and 'development' version numbers, which are internal. The voting solution used in the ODBP project is Pnyx.core ODBP 1.0, which includes several components, including Pnyx.core 1.7.1c (commercial version numbers). After the minor changes carried out on the solution (two files), we have kept the name for certification as Pnyx.core ODBP 1.0 (which includes Pnyx.core 1.7.1c), as these changes were done before the affected components were even compiled and tested by FLDoE. Internally (for Scytl's development purposes) the updated version is named Pnyx.core 1.7.1c1. Of course, the FLDoE was promptly informed of all this when it happened.

4.1.9 Is the structure and implementation of the software well-suited to the security requirements?

4.1.9.1 Finding: We identified two bugs in certificate validation code

We acknowledge the existence of these two bugs, but as stated in our answers to the questions posted by the review team, these issues had already been detected and assigned a low priority in our bug-fixing tasks. These bugs had no effect on the ODBP project, as stated in the findings. It is common practice in the software industry to prioritize detected bugs and issues on software based on their criticity. Before the software was submitted to the FLDoE, these reported bugs had been detected by our internal quality assurance process and classified as non critical, thus allowing other higher priority tasks to be performed before them.

4.1.9.2 Finding: We identified several logic errors in the retry logic in the voting client

As with the previous finding, this is a known low priority issue for the ODBP project, which was detected by our internal quality assurance process before submitting the voting client to FLDoE. Scytl follows industry bug fixing standards where any detected issues are documented, analyzed, prioritized, managed & tracked, scheduled for resolution, fixed, QA'ed, etc. Lower priority issues with no or minimum impact are not always scheduled for immediate resolution, as in this specific case.

4.2 Security Architecture Analysis

4.2.1 Does the target software contain effective controls or safeguards to protect ballot secrecy?

Section 4.2.1 in the review report contains a very useful comparison from a secrecy point of view between Pnyx.core ODBP 1.0 and the other remote electronic voting channels currently accepted by the U.S. for UOCAVA voters. We congratulate the inclusion of this comparison in the report, since it give a reference for a better understanding of the security improvements offered by Pnyx.core ODBP 1.0. However, we disagree with some of the statements in this section which are also further discussed in other sections of the report:

- Pnyx.core ODBP 1.0 ballot secrecy against outsiders: The relevant potential flaw related to the random generator used on the Mixing server was solved during the review and submitted to the FLDoE (and to the review team) before the functional testing started (as discussed in our comments on finding 4.3.4.1). Therefore, the potential vulnerability is no longer possible. In any case, this flaw could not be potentially exploited by outsiders, since it required access to information obtained from the Mixing Server and therefore, only available to trusted personnel.

- Pnyx.core ODBP 1.0 ballot secrecy against insiders: the report states that “there are several ways that insiders might compromise ballot secrecy” but only mentions one: when an insider with access to the Mixing Server has access to the election private key and the encrypted votes. Therefore, there is only one way (not several) and it requires privileged access to the Mixing Server. In the ODBP project, the Mixing Server is in an isolated computer controlled and supervised by Okaloosa Elections Office.

4.2.1.1 Finding: If the software used during the election contained malicious logic, ballot secrecy could be compromised

As stated in section 2, any malicious code can make the voting system behave incorrectly and affect the election security. There are other measures, such as the system audit and certification, which helps to ensure (or in the worst case to reduce to acceptable levels the possibility) that the voting software does not contain malicious code.

Pnyx.core ODBP 1.0 only relies on three trusted components that can be secured against malicious code using physical procedures, which usually helps to simplify the auditing process.

4.2.1.2 Finding: If the voting client software contained malicious code, it could use voter receipts as a subliminal channel to make the receipts salable

This is a special case of the previous finding 4.2.1.1, and we have already stated that if the code is modified, then the system will behave incorrectly, as with any other software system. In our honest opinion, maybe the report could have included finding 4.2.1.1 as the actual finding, with descriptions of potential attacks, such as this one, as examples of exploiting the potential vulnerability inside the same finding 4.2.1.1. Otherwise, the number of reported vulnerabilities will depend on the different examples of exploiting malicious code.

4.2.1.3 Finding: Voter choice records and encrypted ballots contain a component whose implications for ballot secrecy are unclear

As explained in our comments on section 3.3.7 Addendum, the session id of the VCR is not a component of the cryptographic protocol, but a special random number assigned to the VCR in the ODBP project. Therefore, it does not have any cryptographic purpose. This is equivalent to the unique ballot identifier found in paper ballots in some countries, where they are used to fight against ballot-spoofing. Since the contents of the VCR and electronic ballot must be the same, the session identifier is also included in the electronic vote as a field of the ballot.

The risks related to using a session identifier on the VCR are the same as using unique ballot identifiers in elections where paper-ballots are used.

4.2.1.4 Finding: The 'results' report from the Mixing Service could potentially enable coercion, vote-buying, and improper influence, if released to untrusted individuals

First of all, there is a slight misconception regarding the results report from the Mixing Service. The output of the Mixing Service is a digitally signed file with each decrypted individual ballot. This file is then used by a component of Pnyx.core ODBP 1.0 named "ODBP Report Generator" to generate a tabulated results report, which contains the total votes received by each candidate organized by Ballot Style, as required by Okaloosa Elections Office. The report seen by everybody with access to the Mixing Server is the one with the tabulated results. Of course, the Canvassing Board will also be able to review the file with each decrypted individual ballot.

Secondly, this finding is common to any voting system (including paper-based) that supports write-in candidates or multiple races, because they allow pattern voting. For instance, in a paper-based election, this vulnerability can be exploited by attackers if members of the public are allowed near the paper ballots during manual recount. Therefore, the recommendation of restricting access to the clear text votes is valid for any type of voting system. As stated before, Pnyx.core ODBP 1.0 will decrypt the ballots and generate the results in the Mixing Server, an isolated computer supervised and controlled by Okaloosa Election Officials.

4.2.1.5 Finding: Under some conditions, the 'results' report from the Mixing Service could potentially violate ballot secrecy and enable reconstruction of how individual voters have voted, against their knowledge

This finding was clarified during the review but not considered in the report: the batch size used by the Mixing service is not prefixed and can be configured according to the election size. In the case of Okaloosa the batch size was configured to process up to 1999 votes in a single batch. Considering that the maximum number of votes will be 1000, the requirement highlighted by the review team is fulfilled in excess and no secrecy risks are present.

4.2.1.6 Finding: Data retention issues are in tension with ballot secrecy

It is our understanding that the reported issues related to data retention are based on keeping the private key required for decrypting the votes. It is important to notice that this key only exist split into different shares (parts) under the possession of Canvassing Board members.

Based on previous experience, retention policy usually only requires keeping the decrypted votes in order to enable future recounts. This can be fulfilled by means of storing the file obtained from the Mixing Service (see our comments on finding 4.2.1.4), since it contains the list of clear text votes digitally signed by the Canvassing Board. Therefore, any recounts required after the election can be done in the same way as traditional elections with paper ballots, but with the advantage that any attempt of tampering the ballots will be detected (i.e., the digital signature of the file with the decrypted votes preserves their integrity).

However, it is not yet clear (this issue is under discussion at the moment of writing this report) if more information will require storage during the 22 months retention period. If the encrypted votes and the Canvassing Board shares must be retained in order to execute new mixing processes, then the shares and related passwords must be kept in a safe place by FLDoE and/or Okaloosa Elections Office. It is important to note that, even in the case that somebody gains access to this information during the retention period, the digitally signed votes do not contain any personal information (only an anonymous login identifier). Therefore it is impossible to correlate the digital certificates with the physical voters unless the attacker also has access to the ODBP Issuing Point component.

4.2.1.7 Finding: The ballot secrecy property (from the scientific literature) provided by the Pnyx.core ODBP 1.0 voting system is receipt freeness

Scytl agrees that constructing a mathematical proof would increase confidence and we will seriously evaluate this for future audits.

4.2.2 To what extent does the software architecture, implementation, and documentation address the threats which the target software might face?

This section contains some ambiguities as some statements cannot be latter verified according to the described findings. We understand these ambiguities must have been influenced by the time constraints the review team was subjected to. These ambiguous statements are:

- "Insiders: The software architecture alone does not address many insider threats": it is obvious that any software architecture also requires an adequate implementation, as software by itself does not guarantee security of, in the same way a ballot box by itself does not prevent the introduction of bogus ballots by insiders. Additionally, the implementation and processes surrounding any systems management are also of

paramount importance (i.e., physical and procedural security controls). In this manner, Pnyx.core ODBP 1.0 architecture has provisions that facilitate the implementation of procedural and physical security measures. For example:

- The secret sharing scheme reduces the risk the election integrity being compromised by insiders since individuals cannot act alone without the approval of the Canvassing Board.
- Trust is only placed on three components out of nine that compose Pnyx.core ODBP 1.0, thus reducing the chances of insiders manipulating the election. It also facilitates the audit of the overall platform (i.e., audit processes can be more intensive in these components to prevent the inclusion of malware).
- Since the Mixing service is executed in an isolated environment, strong physical and procedural access measures can be implemented to prevent any malicious practice.

- "There are various ways in which these insiders could manipulate the results of the election": Mentioning 'various ways' without providing any example or reference is ambiguous and allows speculation. We have only been able to detect two findings related to this matter (we provide our comments on them below) in the review report.
- The risk of introducing malicious code in the client software relies on at least one malicious insider: Although footnote 6 in this section of the review report provides some sort of clarification on this comment, it is important to note that the Java client is digitally signed using the Okaloosa Elections Office private key. Therefore some sort of collusion is required between the insider that manipulates the code, the custodians of the private key used to sign the manipulated code and the administrators of the voting server platform, which drastically reduces the feasibility of such potential attack.

Finally, we must disagree on the role given by the review team to the Voter Choice Records (VCR) as the "main defense against many or all the attacks". The VCR is an additional component that works alongside other security measures implemented by the system. Therefore, the overall security of the system does not only depend on the VCR and the security controls implemented to protect them. Voting receipts also have an important role, allowing voters to verify that their electronic votes were processed by the Mixing process (this property cannot be fulfilled by VCR) and hence received by the Canvassing Board. Furthermore, the implemented cryptographic protocols ensure the privacy and integrity of the electronic votes (VCR integrity, as any paper vote, depends on physical and procedural measures). VCR is another component of the Pnyx.core ODBP 1.0 system, but not the pivotal one. Furthermore, in case VCRs are lost or compromised, Pnyx.core ODBP provides other means for facilitating the independent audit of electronic votes.

4.2.2.1 Finding: If poll workers are corrupt, then the software does not, on its own, prevent them from casting votes on behalf of voters

As previously stated, if poll workers behave dishonestly, any voting system that relies on them for authenticating voters are subject to this vulnerability. However, Pnyx.core ODBP 1.0 and the ODBP project implement several measures to mitigate this risk:

- The system only allows one vote per voter. An impersonated voter will obtain an error message if he tries to vote, thus detecting the fraud.
- There will be two poll workers per Kiosk site. Both should act dishonestly in order to carry out an attack.
- Votes can only be cast from Voting Kiosks (Voting Laptops).
- After the polls close (the election is closed), Pnyx.core ODBP 1.0 does not accept any more votes.
- Voters must manually sign a Voter Certificate, i.e. a legal document which includes the state oath. These Voter Certificates will be sent to Okaloosa Elections Office from each Kiosk Site. After the polls close, the Canvassing Board will review them to ensure that the signature on each Voter Certificate matches the one on the Voter Registration system.

4.2.2.2 Finding: Voter credentials are not adequately protected against malicious individuals with access to the authentication laptop or malicious software running on this laptop

Voter credentials are kept unencrypted (in plaintext) in the RAM memory (volatile) of the authentication laptop during a short period of time (i.e., the time required to store them on the voter smartcard). The memory is overwritten once the voter credential has been stored. All the credentials are encrypted on the servers hosted at the data center, and only one credential at a time can be stored on the tamper-proof smartcard.

Given the difficulty of successfully setting up an attack on this potential vulnerability (malicious code on the authentication laptop, access to only one credential at a time, which in fact is being requested for a voter physically present in the kiosk site that is going to cast a vote, exporting the stolen credential outside the authentication laptop…), we understand the position of the reviewers stating that they cannot ensure its feasibility.

4.2.2.3 Finding: Voting Client laptop software can be modified

As previously mentioned, Voting Client laptop software is a trusted component whose integrity is protected once certified by FLDoE. This protection is provided by the digital signature of the code and hash on the Live CD. We must also note that there are also some inaccurate statements in this finding:

- Procedures to prevent the LiveCD from the voting laptop being switched are considered. Tamper proof seals will be used on the CD deck after the Live CD is inserted.
- The integrity of the LiveCD is checked by poll workers before being inserted into the voting kiosk (voting laptop). The process for verifying each Live CD follows the standard process specified by FLDoE.
- The integrity check of the LiveCD is done on the authentication laptop. Therefore, this check is not done by the LiveCD itself or in the voting kiosk.
- The LiveCD is based on a standard Linux distribution that can be downloaded from the Internet. The changes made during the hardening are documented by Scytl and were supplied to FLDoE. Therefore, Scytl is not the only entity capable of creating this LiveCD.

Additionally, it is important to mention that even though part of the voting client application is downloaded, the LiveCD only accepts Java applications that are correctly signed by Okaloosa Elections Office.

4.2.2.4 Finding: We do not know whether the voting system adequately protects against denial-of-service attacks

Scytl dimensioned the DDoS counter-measures according to the size of this specific project. For a higher number of polling places additional measures can be taken based on a similar approach.

4.2.2.5 Finding: In some places, the system documentation fails to clearly and accurately represent the properties provided by voter receipts and other auditing mechanisms

We believe the comments regarding this finding can be misleading given that statements taken out of the documentation provided by Scytl appear in the report out of context.

We have never claimed that voting receipts are the only component that allows checking the election accuracy in Pnyx.core ODBP 1.0. In fact, the same “Auditing Pnyx.Core” document used as reference by the review team explains in the Introduction (p. 2), that the election

accuracy is protected by means of auditing the platform components and auditing the data components. In no case is it stated that the accuracy is only preserved by the voting receipt. Furthermore, in page 18 of the same document we state that "The Voting Receipt proves that a vote with the given Ballot Identifier has been cast and accepted by a Voting Service". Therefore, voting receipts alone cannot provide "counted-as-cast" properties. However, if we can trust the Canvassing Board, the Mixing Server and the Voting kiosk, Pnyx.core ODBP 1.0 voting receipts do indeed act as 'counted-as-cast' receipts. Given the procedural and physical measures protecting the Mixing Server, and the Canvassing Board's nature, we can assume that in the ODBP project, Pnyx.core ODBP 1.0 voting receipts will allow voters to verify that their ballots were counted as cast, and therefore, ensure the accuracy of the election.

Finally, regarding the statement about parallel recounts using the file with clear text votes produced by the mixing process, we use the term 'parallel' to indicate that, using the same data, a third party could perform a new recount and validate the election's results (as done in paper-based elections when required). We assume that the review team understands 'parallel recounts' as a way of making an independent parallel recount based on a second independent register of votes, such as the VCRs. If our interpretation is correct, then we understand their comments regarding parallel recounts.

> *4.2.3 The scientific literature identifies several types of verifiability and auditability properties that can be used to characterize the properties of a voting system (e.g., universal verifiability, voter verifiability, software independence, end-to-end auditability, independent auditability). Which of these properties, if any, does the target software provide?*

Although we agree on the majority of the definitions within this section, we found two discrepancies that are not completely aligned with available literature on the subject[3] [4] [5]:

---

[3] Karlof, C., Sastry, N. and Wagner, D. (2005) "Cryptographic Voting Protocols: A Systems Perspective", In Proceedings of the 14th Conference on USENIX Security Symposium, Baltimore, MD.

[4] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In Electronic Voting Technology Workshop, 2008.

- The "cast-as-intended" definition is missing. "Cast as intended" means that voters can verify that the cast ballot recorded by the system contains the selected options, (i.e., that they have not been altered in any way before being cast). In its place, review team is using the term "voter verifiability". However, the "voter verifiability" term is commonly used to specify whether a voting system allows voters to participate on the verification process (verifying that the election is accurate).
- The "voter verifiability" definition is inaccurate. "Voter verifiability" allows voters to verify the accuracy of the election, and supported verification mechanisms include "cast-as-intended" and/or "counted-as-cast". In the report, the review team considers that voter verifiability is equivalent to "cast-as-intended", and, from our point of view, this is inaccurate.

Finally, we disagree with the statement that voting receipts in Pnyx.core ODBP 1.0 do not fulfill the counted as cast statement (later discussed in our comments to finding 4.2.3.1).

4.2.3.1 Finding: Voter receipts fail to ensure votes are counted as cast

In our opinion, the definition of "counted as cast" given by the review team is inaccurate. "Counted as cast", as stated in literature[6], allows voters to verify that their cast votes are counted; but this does not imply that voters can verify that their votes were properly recorded when cast (i.e. "cast as intended"). Therefore, we do not agree with the definition of "counted as cast" provided by the review team;, it also seems to assume the properties of "cast as intended".

Based on the definition of "counted as cast" described above, Pnyx.core ODBP 1.0 achieves the "counted as cast" property by means of using a voting receipt, which contains a random number (receipt id) which is only known to the voter, and can only be retrieved by the Canvassing Board when decrypting the vote. This can be achieved if the Canvassing Board acts honestly and the Mixing Server and Voting Kiosk are trusted components. As mentioned

---

[5] D. R. Sandler, K. Derr, and D. S. Wallach. VoteBox: a tamper-evident, verifiable electronic voting system. Proceedings of the 17th USENIX Security Symposium (USENIX Security '08), 2008

[6] Karlof, C., Sastry, N. and Wagner, D. (2005) "Cryptographic Voting Protocols: A Systems Perspective", In Proceedings of the 14th Conference on USENIX Security Symposium, Baltimore, MD.

before, both components are strongly protected by different physical and procedural measures.

4.2.4 Does the target software provide mechanisms for election auditing? If so, are those audit mechanisms reliable and tamper-resistant? What kinds of security failures are or are not likely to be detected using these audit mechanisms?

Scytl agrees with the fact that the voting receipt does not prove that the Mixing server is behaving correctly. However, this component of the voting system is protected with strong physical and procedural measures, and therefore it is easier to secure than other online systems.

4.2.4.2 Finding: Information needed to verify signatures on voter receipts is not specified for publication

This finding is correct and we will study the possibility of publishing the public key/digital certificate in a user-friendly manner for future implementations.

4.2.5 Trusted components of the target software

4.2.5.1 Finding: The mixing service is trusted

We agree with this finding, since this statement is always true on any mix-net scheme. In these schemes, at least one node must be considered trusted to ensure voter privacy. In our case we only implemented a mix-net of one node and therefore, it must be trusted.

As part of our R&D effort, we are studying ways on improving the auditability of the correct behavior of the Mixing process, such as the ones proposed in [12] and [18]. Regarding privacy, as the report mentions, this property can only be achieved if there is no collusion of the mix-nets. Implementing this in a real environment is difficult, since it requires that each mix node is provided (including the software) by different parties. This makes the counting process more complex and introduces substantial delays. If the software is provided by the same vendor, it must be trusted to some degree, and therefore the situation is similar to having a single mixing node.

As stated in section 2, the Mixing Server, as a trusted component, is strongly protected and supervised by Okaloosa Election officials to ensure that any of the potential vulnerabilities described by the review team are not feasible.

4.2.5.2 Finding: The voting terminal is trusted

We also agree with this statement. For mitigating the described risks, Pnyx.core ODBP 1.0 implements mechanisms that prevent the execution of uncertified software. As previously mentioned, the LiveCD only executes voting applications that are digitally signed by Okaloosa Elections Office digital certificate. Therefore, any other application downloaded from the voting proxy will not be executed. Any modification on the valid voting client software is not allowed passed this checkpoint, due to the integrity features of the digital signature.

4.2.5.3 Finding: The key generation service is trusted

This component has the same role as a Certification Authority of a Public Key Infrastructure; therefore it also needs to be trusted and for that reason is installed in the protected Mixing Server.

4.3 Software Security Analysis

4.3.1 Does the target software contain effective controls or other safeguards to prevent or detect unauthorized tampering with election results, election data, or audit logs?

In addition to the security measures used to detect the deletion of ballots, there is also another mechanism that is not described in this section: the system logs. The system maintains tamper-proof audit logs, which are digitally chained and signed. These audit logs provide significant security enhancements compared with traditional logs, since the integrity of the logging order is preserved by means of hash chains that are digitally signed at configured intervals of time (e.g., once every minute). The description of audit features and processes are available in the “Auditing Pnyx.core” document, page 10.

Furthermore, individual verifiability can also identify (even small) manipulations of the tallying process, including the case where only a small percentage of voters verify their own voting receipts. For instance, in an election with 2,000 cast votes, only 30 voters would be required to verify the presence of their votes in the tabulated results in order to achieve a probability of more than 90% of detecting a manipulation of just 150 of the ballots. If the number of voters that verify their respective votes in this election increase to just 60 voters, the probability of detecting the manipulation of 150 ballots would rise to more than 99%. More details can be found from this reference[7].

[7] NEFF, C. A. Election confidence—a comparison of methodologies and their relative effectiveness at achieving it (revision 6), December 17 2003.

4.3.1.1 Finding: Software components can fabricate logs upon the first audit request

Logs are not stored on local files but on a database. The contents of this log can be accessed during the election through a logviewer (a component of Pnyx.core). In addition, as a common security policy in all the elections that Scytl manages, contents of the database are backed up every predefined quantity of time (usually every 5 minutes). Therefore, log history is kept on backup media for future audits. In case logs are fabricated upon request, they will be no coherent with the contents of the backups.

4.3.2 Are dangerous coding commands (e.g. strcpy) or structures employed?

Scytl has a perfect understanding of these commands, and understands that, if not used correctly, can lead to bugs or security vulnerabilities. In fact Scytl follows rigorous programming standards, and the use of strcpy, strcat and sprintf are strictly forbidden in production code.

While these functions indeed appear in the source code of Pnyx.core ODBP 1.0, they do not exist in any of the production code that will be used during the election; they appear in places such as:

- Source files that are solely used for testing.
- Code used for unit testing.
- Source files that are not compiled/used in this project.
- Source files where the code has been commented out.
- Code used to compile a Voting Service to be run as a WINDOWS service. This is not production code and furthermore it would be out of scope for this project since the Voting Service is run from a Linux machine.

We would have hoped that the review team would have had more time to further investigate each of these strcpy, strcat and sprintf instances to detect where they were located.

4.3.4 Is the cryptography designed and implemented appropriately for the purpose to which it is intended? Is the cryptography used correctly?

Available at: http://www.votehere.net/papers/ElectionConfidence.pdf.

4.3.4.1 Finding: The randomness used to shuffle votes by the mixing service is cryptographically weak

During the mixing server code audit process, reviewers found an issue on the random number generation that could be potentially exploited, under certain circumstances, by the collusion of malicious insiders and malicious voters. A solution was provided for this finding in two days, updating the FLDoE and the review team with the corresponding new source code (changes in 10 lines of code in two files). The fixed Pnyx.core ODBP 1.0 was indeed the one compiled, installed and tested by FLDoE.

Therefore, we strongly disagree with the misleading title of this finding which states that the randomness used to shuffle votes by the mixing service is cryptographically weak. To be factual, it should say ‘was’ instead of ‘is’. This has been previously commented to the review team, but the verbal tense in the finding title has not been changed.

4.3.4.2 Finding: The voting client will accept bogus public keys

The verification process recommended by the reviewers is part of Pnyx.core, and it is usually used in the elections managed by Scytl. However in this case, it was disabled on Pnyx.core ODPB 1.0 due to constraints related to the certification process. This decision was made after assessing the voting system from a holistic perspective. Although the system does not use this verification, any external threat that may try to exploit it is mitigated through the use of other security controls, such as VPNs based on strong authentication mechanisms.

4.3.4.3 Finding: The voting client will accept bogus election identifiers

We acknowledge this finding, but the potential vulnerabilities associated to it are prevented by other security measures, such as using VPNs that prevent Voting Laptops to connect to any fake Voting Server.

4.3.4.4 Finding: A cryptographic key pair is used for both encryption and signing

Scytl understands the concerns expressed by the reviewers, especially in the context of a generic use of a Public Key Infrastructure. However, in the context of Pnyx.core ODBP 1.0, the private key is only used during a limited period of time (i.e., the electoral process). This private key is only used twice during the election for digitally signing information. Furthermore, both digital signatures are carried out at the end of the election. Therefore, risks typically present on general PKI environments, in which keys are valid for years and are used frequently for digitally signing information, do not apply in this case. Nevertheless, we will consider supporting two different keys in future whenever it could be beneficial to the project in question.

4.3.5 To what extent does the system use cryptographic algorithms that comply with recognized standards or that are demonstrably secure under appropriate assumptions?

4.3.5.2 Finding: No security proof is given for any protocols

See our comments on finding 4.1.1.1.

4.3.6 Do the key management processes generate and protect strong keys?

4.3.6.1 Finding: The canvassing board and election administrator key are not, by default, adequately protected against loss or theft of smartcards

Although we agree on the finding, which was already known by Scytl, the likelihood of the attack in question is extremely low: it would require stealing at least two smartcards from Canvassing Board members (without such members noticing), and gaining access to the Mixing Server, which is secured and under continuous supervision of Okaloosa election officials. In any case, the new version of Pnyx.core uses a different approach for storing the shares in smartcards: they are now protected by an 8-digit PIN code that blocks the card after three invalid access attempts. This way we are able to increase the security of the system while continuing to be user-friendly at the same time.

4.3.6.2 Finding: The architecture does not protect private keys against compromise of the Mixing Service computer

As we have stated in various occasions, the Mixing Server is a trusted component with strong security measures, and therefore, the feasibility of the described attack is very low.

Pnyx ODBP v 1.0 does not implement threshold cryptography from the point of view of "threshold decryption" or "threshold signatures". However, it is using the concept of "threshold scheme" (Shamir) for protecting the private key that decrypts the votes.

4.3.9 Are communications channels properly privacy and integrity protected?

4.3.9.1 Finding: Mutual authentication is provided by lower-level protocols, not by the application-layer voting protocol

We understand the review team concerns on this subject. However, we believe that their recommendations are done from a general perspective instead of considering the specific implementation in Pnyx.core ODBP 1.0.

Level 1 provides mutual authentication: both ends use a digital certificate to authenticate each other before establishing the tunnel. Level 1 provides an encrypted channel to the other layers. On the other hand, combining server authentication at Level 2 and client authentication at Level 3 also provides a second layer of cross-authentication.

4.3.9.2 Finding: OpenVPN and communications security

We congratulate the review team for this analysis on OpenVPN and the findings they made which allowed the OpenVPN developers to improve the software and fix some serious bugs. We also accept all the suggestions provided to improve the security configuration of OpenVPN. If fact, several of them have been already implemented (others cannot be implemented due to functional testing constrains).

Regarding the main reason of using a “release candidate” version instead of the stable one, is that the new version supports hardware cryptographic tokens. Therefore, private keys can be stored on smartcards instead of being stored on disk, which is evidently less secure. Notice that the Certification authority used to issue the digital certificates of the OpenVPN nodes is also stored on a smartcard that is air-gapped from any system.

4. Conclusion

As stated in the review report, the use of cryptographic mechanisms in voting systems represents a significant "paradigm shift from election systems currently used in elections" (Chapter 3). For this reason, this report represents a pioneering experience of collaboration between academics and a vendor in the review of a voting system whose security relies primarily on cryptographic mechanisms complemented by physical and procedural security measures.

As explained above, we do not fully agree with some of the findings in the review report. In general, the source of our disagreement is more in the manner in which the findings are presented than in the findings themselves. For example, we consider that some of these findings have not been adequately portrayed since the likelihood of their occurrence and their potential impact have not been considered nor evaluated by the review team. This is probably due to time constraints which have not allowed the review team to have access to some of the procedural and physical security measures that will be implemented in the ODBP project. A thorough analysis of these security measures would have allowed the review team to adequately assess the potential likelihood and impact of the identified risks.

We are also in disagreement with the preponderant weight given to the 'Voter Choice Records' over any other security element in the solution. This is consistent with the views of certain members of the review team regarding the impossibility of having secure electronic voting without software independence (i.e., without a paper audit trail). These views in favor of a paper audit trail are clearly reflected in this review report, which is legitimate but should be mentioned so the readers can understand the rationale for the reiteration of the comments regarding the importance of the 'Voter Choice Records' in the report. We believe that electronic voting with adequate logical, physical and procedural security measures can be as secure, if not more secure, than paper-based voting systems.

Despite our disagreement with some of the findings, we are generally pleased with the overall conclusions that can be derived from this review report. In particular, we are pleased with the following facts:

- The reviewers did not find any significant bugs or malware in the Pnyx.core ODBP 1.0 software that could put the integrity of the election at risk. Although the reviewers stated that they did not do a "line-by-line" analysis of the source code due to time constraints, the fact remains that a group of experts of this caliber,

with significant experience in code reviews, did not find any significant bugs or malware.

- The reviewers did not find any significant flaws in the security architecture designed for Pnyx.core ODBP 1.0.
- The reviewers did not find any significant weaknesses with the cryptographic algorithms implemented by Pnyx.core ODBP 1.0. Although in some cases, alternative cryptographic algorithms were recommended by the reviewers, in none of those cases were the algorithms implemented by Pnyx.core ODBP 1.0 considered unacceptable.

Furthermore, we are especially pleased with the fact that the reviewers considered that the Pnyx.core ODBP 1.0 solution is superior to the current remote voting systems that it is intended to replace (i.e., postal voting, e-mail voting and fax voting) in terms of criteria such as integrity of votes, voter privacy and election results auditability (See section 4.2.1 of the review report). This conclusion is especially important because, in our opinion, any new voting system should always be evaluated based on how it compares against current voting systems.

Finally, we consider this review as a very challenging but useful exercise that can help close the gap between the vendor and academic communities. This review proves that the transparent collaboration of vendors and academics can significantly help states evaluate new voting systems.