KNOWiNK Poll Pad 3 v2.4.8.01 Electronic Poll Book System Source Code Review Test Report for California

KNI-19001-SCRTR-01

Vendor Name	*KNOWiNK*
Vendor System	*Poll Pad 3 v2.4.8.01*

Prepared by:

4720 Independence St.
Wheat Ridge, CO 80033
303-422-1566
www.SLICompliance.com

Accredited by the Election Assistance Commission (EAC) for Selected Voting System Test Methods or Services

Revision History

Date	Release	Author	Revision Summary
October 12th, 2019	1.0	*J. Panek, M. Santos*	Initial Release
November 19th, 2019	1.1	*J. Panek*	Updates made per request from CA SoS

Disclaimer

The information reported herein must not be used by the client to claim product certification, approval, or endorsement by NVLAP, NIST, or any agency of the Federal Government.

Trademarks

- SLI is a registered trademark of SLI Compliance.
- All products and company names are used for identification purposes only and may be trademarks of their respective owners.

TABLE OF CONTENTS

Test Report Identification

Portion of Test Plan Addressed

This Test Report details the security and integrity review of the iOS/Swift and Ruby on Rails source code of the **KNOWiNK Poll Pad 3 v2.4.8.01** electronic poll book system. An assessment of potential vulnerabilities is provided where applicable.

References

The following key documents were used in preparing this Test Report.

1. California Electronic Poll Book Regulations
2. Swift-Style-Guide
   https://github.com/raywenderlich/swift-style-guide
3. iOS / Swift best practices for 2018
   https://theswiftdev.com/2018/02/09/ios-swift-best-practices-for-2018/
4. Guides.rubyonrails.org
5. Ruby-doc.org
   https://www.sitepoint.com/10-ruby-on-rails-best-practices

Source Code Review Process

A manual security and integrity review of the iOS/Swift and the Ruby on Rails code for the **KNOWiNK Poll Pad 3 v2.4.8.01** electronic poll book system was performed to analyze for findings against the following requirements:

- Evaluation of potential vulnerabilities and related issues (code quality and standards compliance), considering that an exploitable issue in a component that is not in itself security relevant could be used to subvert more critical data. This is an issue whenever the architecture of the system does not provide strong separation of the components.
- Adherence to other applicable coding format conventions and standards including best practices for the coding language used, and any IEEE, NIST, ISO or NSA standards or guidelines which the contractor finds reasonably applicable.
- Analysis of the program logic and branching structure.
- Search for exposures to commonly exploited vulnerabilities, such as buffer overflows, integer overflow, inappropriate casting or arithmetic.
- Evaluation of the use and correct implementation of cryptography and key management.
- Analysis of error and exception handling.

- Evaluation of the likelihood of security failures being detected.
  - Are audit mechanisms reliable and tamper resistant?
  - Is data that might be subject to tampering properly validated and authenticated?
- Evaluation of the risk that a user can escalate his or her capabilities beyond those authorized.
- Evaluation of whether the design and implementation follow sound, generally accepted engineering practices. Is code defensively written against:
  - Bad data;
  - Errors in other modules;
  - Changes in environment;
  - User errors; and
  - Other adverse conditions?
- Evaluation of whether the system is designed in a way that allows meaningful analysis, including:
  - Is the architecture and code amenable to an external review (such as this one)?
  - Could code analysis tools be usefully applied?
  - Is the code complexity at a level that it obfuscates its logic?
- Search for embedded, exploitable code (such as “Easter eggs”) that can be triggered to affect the system.
- Search for dynamic memory access features which would permit the replacement of certificated executable code or control data or insertion of exploitable code or data.
- Search for use of runtime scripts, instructions, or other control data that can affect the operation of security relevant functions or the integrity of the data.

Review Results and Analysis

SLI conducted a source code review of the **KNOWiNK Poll Pad 3 v2.4.8.01** electronic poll book system for compliance against the California Electronic Poll Book Regulations.

The source code was evaluated for potential vulnerabilities and related issues (code quality and standards compliance), considering that an exploitable issue in a component that is not in itself security relevant could be used to subvert more critical data.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was the following determinations:

- Improper use of Controller-View-Model programming structure was observed, which has the potential for inadvertent data loss.
- Indications of code being implemented that allows non-current files to be used.
- Indications of back door code that would allow unauthorized access.
- Indications of superfluous code comments being present.
- Indications of a superfluous code comment regarding database check-ins and check-outs.
- Indications of code breaking and specific defined items not persisting in the database.
- Indications of servers being run in development mode.
- Indications of application instability.
- Indeterminate implementation of encryption within the system.
- Missing data checks between runs.
- Delete function not appropriately logged.
- Severe errors not being logged.
- Unique ID generation not being verified.

The source code was reviewed for adherence to applicable coding format conventions and standards, including best practices for iOS/Swift and Ruby on Rails.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was the following determinations:

- All error messages should be unique in the system so that errors reported can be traced to a single point in the code base.
- Commenting within source code should be meaningful.
- Functions should be called in a non-conflicting manner.
- Incorporated values should be implemented correctly.
- Unused/deprecated code should be removed.

The source code was reviewed to analyze program logic and branching structure.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was the following determination:

- Improper use of Controller-View-Model programming structure was observed, which has the potential for inadvertent data loss.

The source code was reviewed for exposures to commonly exploited vulnerabilities, such as buffer overflows, integer overflow, inappropriate casting or arithmetic.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was a determination that no issues were found.

The source code was reviewed for use and correct implementation of cryptography and key management.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was the following determination:

- Indeterminate implementation of encryption within the system.

The source code was reviewed for error and exception handling.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was a determination that no issues were found.

The source code was reviewed for likelihood of security failures being detected.

- Are audit mechanisms reliable and tamper resistant?
- Is data that might be subject to tampering properly validated and authenticated?

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was the following determinations:

- Delete function not appropriately logged.
- Severe errors not being logged.

The source code was reviewed for risk that a user can escalate his or her capabilities beyond those authorized.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was a determination that no issues were found.

The source code was evaluated for whether the design and implementation follow sound, generally accepted engineering practices. Is code defensively written against:

- Bad data;
- Errors in other modules;
- Changes in environment;

- User errors; and
- Other adverse conditions.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was the following determinations:

- Indications of code breaking and specific defined items not persisting in the database.
- Indications of servers being run in development mode.
- Indications of application instability.
- Indeterminate implementation of encryption within the system.
- Missing data checks between runs.
- Delete function not appropriately logged.
- Severe errors not being logged.

The source code was reviewed for whether the system is designed in a way that allows meaningful analysis, including:

- Is the architecture and code amenable to an external review (such as this one)?
- Could code analysis tools be usefully applied?
- Is the code complexity at a level that it obfuscates its logic?

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was a determination that no issues were found.

The source code was reviewed to search for embedded, exploitable code (such as "Easter eggs") that can be triggered to affect the system.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was the following determinations:

- Indications of code implemented that allows non-current files to be used.
- Indications of back door code that would allow unauthorized access.
- Indications of superfluous code comments being present.
- Indications of a superfluous code comment regarding database check-ins and check-outs.

The source code was reviewed to search for dynamic memory access features which would permit the replacement of certificated executable code or control data or insertion of exploitable code or data.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was a determination that no issues were found.

The source code was reviewed to search for use of runtime scripts, instructions, or other control data that can affect the operation of security relevant functions or the integrity of the data.

The expected outcome for this review was that no issue would be found.

The actual outcome for this review was the following determinations:

- Severe errors not being logged.
- Functions should be called in non-conflicting manner.
- Incorporated values should be implemented correctly.

Findings

This section summarizes findings from the **KNOWiNK Poll Pad 3 v2.4.8.01** electronic poll book system source code review.

Source Code Review Discrepancies

The following discrepancies were found during the source code review of the **KNOWiNK Poll Pad 3 v2.4.8.01** electronic poll book system.

- Code was found indicating a specific function is called twice, once conditionally and once unconditionally.
- All error messages should be unique in the system so that errors reported can be traced to a single point in the code base.
- The system logs “WHAT” as a string; this appears to be a meaningless statement.
- A call to generate a unique ID should be checked against current database before acceptance; otherwise, a duplicate may occur.
- Controllers should not throw model windows. That should be done by the view to maintain proper architecture.
- Two files were found containing comments stating there is a possible race condition.
- t.datetime should be set up to auto-populate.
- Indications of superfluous code comments being present.
- Indications of a superfluous code comment regarding database check-ins and check-outs.
- One file was found containing a to-do comment stating there is a possible probe.

- Numerous files were found containing commented out sections of source code.
- Numerous files were found containing general to-do comments.
- Numerous files were found containing superfluous comments.
- A comment was found stating a coding inconsistency is causing issues.
- A to-do comment was found indicating broken functionality in code.
- An audit log file was found where almost all of the code is commented out due to "100 failures on master," as stated in the comment.
- One file had to-do comments stating the code is breaking and specific defined items aren't persisting in the database.
- As a general note, this source code contains numerous instances of numbers other than 0, 1, −1 being used without comment, or being set to a constant.
- As a general note, this source code contains numerous instances of to-do comments that suggest code blocks are not optimal, there are possible issues or code is broken.
- A general to-do comment was observed stating servers are being run in development mode. This is a potential vulnerability if not changed before the software is officially in use.
- As a general comment, while it looks like encryption is part of the iOS/Swift code package, for P2P it must be used to be secure. No evidence was found of use of encryption.
- It appears that the syncable exporter can be restarted, but there is no checking to see if the input may be modified between runs.
- The software has the ability to delete log files and that activity is not itself logged, violating standard security practice.
- As a general comment, severe errors should log (not display) traceback for forensic purposes for all fatal conditions.
- Comments were observed about application instability after running an import function.

Potential Vulnerabilities

To the extent possible, reported vulnerabilities include identification of the applicable requirement as well as an indication of whether the exploitation of the vulnerability would require access by:

- Voter: Usually has low knowledge of the Electronic Poll Book System’s software and/or hardware design and configuration. Some may have more advanced knowledge. May carry out attacks designed by others.

- Poll worker: Usually has low knowledge of the Electronic Poll Book System's software and/or hardware design and configuration. Some may have more advanced knowledge. May carry out attacks designed by others. They have access to the software and/or hardware for up to ten days, but all physical security has been put into place before the machines are received.
- Elections official insider: Wide range of knowledge of the Electronic Poll Book System's software and/or hardware design and configuration. May have unrestricted access for long periods of time. Their designated activities include:
  - Set up and pre-election procedures;
  - Election operation;
  - Post-election procedures; and
  - Archiving and storage operations.
- Vendor insider: With great knowledge of the Electronic Poll Book System's software and/or hardware design and configuration. They have unlimited access to the Electronic Poll Book System's software and/or hardware before it is delivered to the purchaser and, thereafter, may have unrestricted access when performing warranty and maintenance service and when providing election administration services.

SLI will not verify or demonstrate exploitability of the vulnerability, but the report of the vulnerability will identify factors involved in the exploitation. Any vulnerability theories developed by the source code review team members are, to the extent possible, provided to the Secretary of State staff herein.

Potential Vulnerabilities Found

For each of the items below there is reason to believe that if somebody (poll worker, elections official insider, vendor insider) were to get unauthorized access to these sections of code, data corruption or loss could occur. Other, less likely uses for some of these findings would be using these instabilities or code flaws to cause system shutdown/reboot, allowing malicious code to be placed onto the system without specific attention being drawn to it.

The following potential vulnerabilities were found during the source code review of the **KNOWiNK Poll Pad 3 v2.4.8.01** electronic poll book system.

- A superfluous code comment was observed. This vulnerability would require access by a poll worker, elections official insider, or vendor insider.
- A to-do comment was observed that states there is a possible probe. This vulnerability would require access by a poll worker, elections official insider, or a vendor insider.
- Some superfluous to-do comments were observed. This vulnerability would require access by a vendor insider.

- Superfluous to-do code comments were observed regarding a database check-in and check-out process. This vulnerability would require access by an elections official insider or a vendor insider.
- To-do comments were observed that state the code is breaking, and specific defined items aren't persisting in the database. This vulnerability would require access by an elections official insider or a vendor insider.
- A general to-do comment was observed stating servers are being run in development mode. This is a potential vulnerability if not changed before the software is officially in use. This vulnerability would require access by a vendor insider.
- Comments were observed about application instability after running an import function. This vulnerability would require access by a poll worker, elections official insider, or a vendor insider.
- As a general comment, while it looks like encryption is part of the iOS/Swift code package, for P2P it must be used to be secure. No evidence was found of use of encryption. This vulnerability would require access by a poll worker, elections official insider, or a vendor insider.
- It appears that the syncable exporter can be restarted, but there is no checking to see if the input may be modified between runs. This vulnerability would require access by a poll worker, elections official insider, or a vendor insider.
- The software has the ability to delete log files and that activity is not itself logged, violating standard security practice. This vulnerability would require access by a poll worker, elections official insider, or a vendor insider.
- Controllers should not throw model windows. That should be done by the view to maintain proper architecture. This vulnerability would require access by a vendor insider.
- As a general comment, severe errors should log (not display) traceback for forensic purposes (e.g. i.backtrace.inspect) for all fatal conditions. This vulnerability would require access by a poll worker, elections official insider, or a vendor insider.
- A call to generate a unique ID should be checked against current database before acceptance; otherwise, a duplicate may occur. This vulnerability would require access by an elections official insider or a vendor insider.

Conclusion

During the source code review of the **KNOWiNK Poll Pad 3 v2.4.8.01** electronic poll book system, 32 unique discrepancies were noted during the software review. Of the discrepancies noted, 13 are considered potential vulnerabilities.

End of Test Report