InkaVote Plus Source Code Review

for

California Secretary of State
Debra Bowen

2-7 October 2007

The source code review for the InkaVote Plus system was conducted by:

atsec information security corporation
9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

for California Secretary of State Debra Bowen under contract with Freeman, Craft, & McGregor Group (FCMG). atsec is accredited as a Common Criteria Evaluation Lab, a Cryptographic Module Test Lab (FIPS 140-2), and provides other computer security testing services for commercial companies.

General Description of Equipment Under Test (EUT)

The InkaVote Plus system, marketed by Election Systems & Software (ES&S), consists of the InkaVote Precinct Ballot Counter (PBC) and Unisyn Election Management System (EMS). The PBC is based on a standalone lottery ticket machine design developed by the International Lottery & Totalizator Systems, Inc. (ILTS). The system supports the InkaVote ballot, which has been used in County of Los Angeles and City of Los Angeles elections for several years. The InkaVote ballot is a mark sense ballot based on the design of a Hollerith (IBM) punch card. Ballot identification data is pre-punched in the leading columns. To vote, the card is placed in a marking device, which has a ballot-voting booklet and template guide showing the location to mark a vote for each candidate in each contest. A special marking pen is used to mark the voter's choices. The InkaVote Plus PBC unit may be equipped with an optional component called the Audio Ballot unit, which provides support to assist visually blind as well as other voters who need an audio ballot. The Audio Ballot unit consists of a keypad, earphones, and printer, and does not include a visual display for the voter of the ballot. This unit uses an audio ballot script, which guides the voter through voting and prints a marked InkaVote ballot. The voter may then insert the marked ballot into the PBC unit, which checks for overvotes and blank ballots. Voters who mark their ballots manually or with the ballot booklet template may also use the PBC unit to check the ballots for overvotes and blank ballots. If an overvoted or blank ballot is detected, the system returns the ballot to the voter, giving the voter an opportunity to remake the ballot. This error checking is a Help

America Vote Act (HAVA) requirement. Although the PBC unit is capable of tallying the ballots and producing a machine report of the results when the polls close, the City of Los Angeles and County of Los Angeles only use the system for the audio ballot and error checking functions, without using the ballot tally and reporting functions. The InkaVote ballots in the City and County of Los Angeles are tallied and reports are generated by a central counting system used for all the ballots, including both the polling place and absentee ballots.

The Unisyn EMS suite of applications is a set of Java-based software applications which allows the user to create election definitions for the PBC, and load the election definition into one or more PBCs (multiple units may be programmed using an Ethernet link). The suite design includes the option to load compatible XML formatted election definitions from other election management systems. Once the polls close, the tally results may be transferred back to the EMS suite for accumulation of multiple PBCs' results and reporting. The Unisys EMS suite of applications operates on Windows XP-supported workstations. EMS component applications operate independently and may be installed on separate workstations as needed. The component applications include:

- an election database, using MySQL;
- the application to modify and define the election for each election, which is identified in the manuals as the "EMS" application or "Election Generator";
- an Election Converter which converts an XML description of an election and produces an encrypted Election CD;
- an Election Loader, which supports the installation of the election provided by the Election CD in each PBC using a local Ethernet network;
- a Vote Converter to transfer the voting results from the PBC using a USB memory media device as a carrier, a.k.a. Transfer Device; and
- a Vote Tabulation module to tabulate, consolidate, and generate election reports on the voting results.

The County of Los Angeles provides the XML election definition from their legacy election system to the Election Converter component and uses the Election Loader component to load the election into the PBC. Because the City and County of Los Angeles do not use the tabulation and reporting capabilities of the system, the other components of EMS are not used.

Scope Limitations

The City and County of Los Angeles only use InkaVote Plus PBC for the specific purposes of:

- detecting and preventing the casting of ballots which are blank;
- detecting and preventing the casting of ballots which have at least one overvoted race; and
- providing the Audio Ballot interface which marks ballots for voters requiring the audio ballot.

The ballot tabulation and reporting features of the InkaVote Plus system are not being used in this venue. Accordingly, the examiners were asked to limit their examination, where possible, to the modules of the system which are being used by the County and City of Los Angeles and to vulnerabilities that affect:

- the integrity of the election definition needed to support the error detecting and Audio Ballot functions;
- security audit logs and the log reporting services; and
- the basic operation of the PBC (i.e., denial of service attacks).

The full system was supplied as a testing resource, and all technical documentation was provided for reference. Documents and source code not in the scope of testing were available to the Source Code Review Team as a resource, if needed. For example, during the Red Team test, the tally and report generation features within the PBC were used to document and demonstrate the effectiveness of one of the demonstrated exploits. If the Source Code Review Team did notice that an identified vulnerability could affect vote tallies or reports, they were encouraged to report it, although it was not a primary focus.

The County of Los Angeles procedures and programs to generate the XML were outside the scope of testing.

For the purpose of the test, the test team was asked to consider four classes of attackers:

- **Voter**: Usually has low knowledge of the voting system machine design and configuration. Some may have more advanced knowledge. May carry out attacks designed by others. They have access to the machine for less than one day.
- **Poll worker**: Usually has a low knowledge of the voting machine design and configuration. Some may have more advanced knowledge. May carry out attacks designed by others. They have access to the machine for less than one day.
- **Election official insider**: Has a wide range of knowledge of the voting machine design and configuration. They may have restricted access for long periods of time. Their designated activities include:
    - Set up and pre-election procedures.
    - Election operation.
    - Post election processing of results, and
    - Archiving and storage operations.
- **Vendor insider**: Has a great knowledge of the voting system design and configuration. They have unlimited access to the machine 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.

atsec added one other category on FCMG recommendation: the **storage or warehouse worker** with virtually unlimited access between elections.

The team was not limited to these attackers, and their direction included direction from Resolution # 17-05 of the Technical Guidelines Development Committee (hereafter “TGDC”) of the U.S. Election Assistance Commission, adopted at the TGDC plenary meeting on January 18 and 19, 2005, which calls for:

> "... testing of voting systems that includes a significant amount of open-ended research for vulnerabilities by an analysis team supplied with complete source code and system documentation and operational voting system hardware. The vulnerabilities sought should not exclude those involving collusion between multiple parties (including vendor insiders) and should not exclude those involving adversaries with significant financial and technical resources."

The specific tasking, as presented in the Statement of Work for the Source Code Review Team, was:

> *"The review places emphasis on security and integrity of the system and should identify any security vulnerabilities that could be exploited to alter vote recording, vote results, critical election data such as audit logs, or to conduct a "denial of service" attack on the voting system.*
>
> *The review will include, but not be limited to:*
>
> - *Adherence to the applicable standards in sections: 4 of Volume I [*Software Standards*], 7 of Volume I* [Quality Assurance], *and 5*[Software Testing] *of Volume II of the 2002 Voluntary Voting System Standards.*
> - *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 reviewers find 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 which are 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.*
  - *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.*

….

The review is to provide a "Vulnerability Assessment", based upon the model provided in ISO/IEC WD 18045:2006(E) Information Technology-Security Techniques-Methodology for IT Security Evaluation, App B documenting and categorizing vulnerabilities, if any, to any tampering or errors that could cause incorrect recording, tabulation, tallying or reporting of votes or that could alter critical election data such as election definition or system audit data."

Operation of the Review

The review was conducted 2-14 October 2007 at the atsec offices in Austin, TX. The team consisted of two experts from atsec (Stephan Műller and Klaus Weidner) and was supported by meetings with FCMG (Steve Freeman).

The review (consisting of documentation review and source review) examined the ES&S Technical Data Package (TDP) and the source code. The TDP and source code used were verified copies of the TDP and source code, which were sent from the National Association of Election State Election Directors (NASED) Independent Test Authority (ITA) lab. The chain of custody followed the files from the lab, to the Secretary of State, to the Source Code Review and Red Teams at atsec. The integrity of the delivered documents was verified from electronic file signature hashes provided by FCMG from the trusted sources original disks.

atsec divided the documentation review (based only on the TDP, with no reference to the source code) into two categories for reporting:

5.1 Sufficiency to Enable Review of Source Code
5.2 Sufficiency to Design and Conduct Tests

The source code review (based on the TDP, in addition to the source code) used a combination of manual review and automated data collection and analysis methodologies to identify potential areas for exploitation. The source code review was divided into the following categories for reporting:

6.1 Adherence to applicable standards
6.2 Adherence to other coding format conventions and standards
6.3 Program logic and branching structure
6.4 Commonly exploited vulnerabilities
6.5 Cryptography and key management
6.6 Error and exception handling
6.7 Likelihood of security failures being detected
6.8 Privilege escalation
6.9 Best practices / defensive coding
6.10 System amenability to analysis
6.11 Dynamic memory access features
6.12 Runtime scripts / instructions / control data

Because of the limited time (12 days) and broad scope (assessment of documents and quality of the code, along with source code review), the team concentrated on surveying a breadth of categories of vulnerabilities that they could identify, and only reviewed in depth enough samples of each of the categories to determine how that vulnerability was being handled. For all the categories, no attempt was made to enumerate how many instances existed. Other source code review projects would be likely to find more instances, but those findings should be within the listed categories.

Test tools used included lexical scanners and special code review tools from open sources, commercially available search and analysis tools, and in-house developed scripts. Details specifying tools and sources, as well as the scripts used for the tools are provided in the confidential reports.

<u>Results Summary</u>

Full details will be found in the confidential source code review report, including the detailed work papers. A vulnerability summary table is found at the end of this report, as well as a description of the rating system used. The vulnerability rating assessment is based on the Common Methodology for Information Technology Security Evaluation (CEM v3.1) Rev 1 and Rev 2, App B. The use of this terminology is for convenience in characterizing the potential vulnerability of the system to the identified attack, but is not necessarily compliant with and should not be taken as representing a full, formal finding under Common Criteria evaluations. The document review and compliance check against the VSS (2002) are not applicable to the vulnerability assessment and are rated accordingly.

The document reviews and assessments are listed in the summary table, but do not carry a vulnerability assessment unless vulnerabilities were detailed in the worksheet product.

Unless otherwise indicated, the potential vulnerabilities found in the source review have not been confirmed to be exploitable in the full-deployed environment due to time constraints, and it is possible that technical measures outside of the specific module being examined may prevent an exploit. For the purposes of the vulnerability rating, only assumptions, checks, and protective measures which are clearly identified in the relevant code comments or documentation are considered to be in place. For example, if a function implicitly assumes that parameters are checked or sanitized in a different code location, but no documentation exists for this assumption, the reviewer did not attempt to trace code paths to check if this implicit assumption is appropriate. For security critical sections, the reviewer's expectation was that either explicit checks or clear and verifiable documentation about assumptions should exist.

Document Assessment

5.1 Sufficiency to Enable Review of Source Code

The documentation provided by the vendor states the system design specifications in very general terms. There is no detailed description of software components and algorithms that could be directly compared to specific software modules in the source code. This means that the documents are of very limited value to conduct a design assessment that allows searching for vulnerabilities (A.1 of the Summary Table below). No specific vulnerabilities were identified so there is no vulnerability assessment on this finding.

The documentation provided by the vendor does not contain any test procedure description; rather, it provides only a very abstract description of areas to be tested. The document mentions test cases and test tools, but these have not been submitted as part of the TDP and could not be considered for this review. The provided documentation does not show evidence of "conducting of tests at every level of the software structure". The TDP and source code did not contain unit tests, or any evidence that the modules were developed in such a way that program components were tested in isolation.
The vendor documentation contains a description of cryptographic algorithms that is inconsistent with standard practices and represented a serious vulnerability. No vulnerability assessment was made as part of the documentation review because the attack approach could not be identified based on the documentation alone. (The source review identified additional specific vulnerabilities related to encryption).

The reviewer found inconsistencies, wrong references, and a lack of technical details on the Linux hardening1 procedures to be used (A.3). The Red Team reported encountering

[1] Hardening is a technique which has been supported by the publication of standard guidelines through Microsoft, NIST, and the Center for Internet Security for reducing the services and features from a default installation of the operating system. This practice supports Software Engineering security principles such

some good hardening practices on the test machines that prevented many common attacks, but these were apparently done by the ES&S/ILTS installation crew that set the system up for Red Team testing and may not be documented. The vendor-supplied material does not provide assurance that this is the standard procedure for all installed systems. The Source Code Review Team did note that the version(s) of the Linux Operating System described was an older version that is no longer being maintained by the operating system distributor. As such, the lack of updated security patches and releases suggests that there are documented vulnerabilities available through the Web. The Red Team was successful in several attacks using openly known vulnerabilities. The System Security Specification identified a file as being generated "as part of the configuration process for the customer." The Red Team had found the file and determined it contained the Jurisdiction key, determined it is used to create encryption keys for the election, and used it plus some other information to open all the files, including encrypted files on the Election CD. The problem the Review Team identified was that there is no description of how or when the file is created and how it was handled (A.4). As it is a significant factor in the creation of the encryption keys used by EMS and the PBC, secure handling and management is necessary but undocumented.

Source Code Assessment

Vulnerabilities

The source review identified potential or actual vulnerabilities as listed in the appendix of this report, and detailed in the confidential report.

In the area of cryptography and key management, multiple potential and actual vulnerabilities were identified, including inappropriate use of symmetric cryptography for authenticity checking (A.8), use of a very weak homebrewed cipher for the master key algorithm (A.7), and key generation with artificially low entropy which facilitates brute force attacks (A.6). In addition, the code and comments indicated that a hash (checksum) method that is suitable only for detecting accidental corruption is used inappropriately with the claimed intent of detecting malicious tampering. The Red Team has demonstrated that due to the flawed encryption mechanisms a fake election definition CD can be produced that appears genuine, see Red Team report, section A.15.

106 instances were identified of SQL statements embedded in the code with no evidence of sanitation of the data before it is added to the SQL statement. It is considered a bad practice to build the SQL statements at runtime; the preferred method is to use predefined SQL statements using bound variables. A specific potential vulnerability was found and documented in A.10, SQL Injection. SQL injection attacks could potentially be used to modify any of the information stored in the database, bypassing the sanity checks and logging that the code would normally do. Note that in the intended deployment scenario, the software's capability of aggregating vote counts in the database is not used.

as 'Least privileges'. In application, the technique requires careful testing and implementation to avoid disabling applications (a potential 'denial of service' condition) or hamper application design features providing other security protections.

A potential vulnerability was found related to Zip File directory traversal (A.9) with the potential impact of creating or overwriting files on the system in attacker-specified locations outside of the intended storage directory.

The reviewer found no instances of deliberately inserted back doors or Easter eggs. However,

- the Zip File directory traversal (A.9),
- SQL injection (A.10), and
- the egregious use of cryptography (A.6, A.7, A.8) could be exploited as a back door.

<u>Adherence to applicable standards</u>

The Voting Systems Performance and Test Standards [VSS (2002)] provide standards related to coding conventions and best practices. The reviewer examined the source code for applicable items as specified in the work plan and reported the result of this examination.

The confidential report lists specific instances where code constructs do not appear to match the requirements of this standard. Note that these instances are not automatically equivalent to actual or potential vulnerabilities. This summary report omits instances that do not appear noteworthy. The following items are considered significant due to being an obstacle for effective code analysis, potentially hiding other problems, or because they could contribute to introducing problems in later changes to the source code.

- Missing validation of input parameters or otherwise inadequate specification of expected range values, including instances of mismatches between the documented and actual semantics of functions.
- Java usually handles abnormal conditions with exceptions and expected conditions with returned values. Multiple examples are provided in the confidential report where the constructs are used inconclusively or inappropriately, or where use of abnormal conditions handled through exceptions is hidden from source review.
- Vote counter/integer overflow. This was out of scope for the review since the system will not be processing vote counts, but the reviewer did note that the counters do not have an overflow or out-of-bounds check. The code assumes that the Java native variables will be large enough to handle integer computations without overflow. The VSS-2002 does not accept this justification.
- Two incidents of lines containing a conditional and an executable statement on the same line.
- Uses of numeric constants other than 0 and 1 should be explained by expressive variable names or code comments. Multiple cases were found where constants were used without adequate explanation. Most examples are where the constant value is named as a variable but the name does not indicate why that value is

significant, for example assigning the value “5” to a variable named “five” with no further explanation.
- One instance of nested use of the conditional “?:” operator in a complex multi-line expression.

Overall secure design and implementation

The applications all run at a privilege level that provides full read/write access to all security critical application data. The ‘least privilege’ principle is not exercised. The lack of privilege separation in the design does not support reliable detection of security failures.

Design documents and code comments do not provide any evidence that audit logs are protected from tampering. The code segments doing logging have sufficient privileges to modify or delete logs due to the lack of privilege separation. The design documents do not mention use of operating system features that support the integrity of the logs.

The system design does not depend on runtime scripts or instructions for its operation. It does depend on data provided at runtime, specifically the election definition file. The vulnerabilities related to the handling of this file (such as A.9 and the encryption related vulnerabilities) provide avenues for attack that can affect the integrity of data, including the integrity of installed software components.

System amenability to analysis

The reviewer noted the following items as impediments to an effective security analysis of the system:
- Lack of design documentation at appropriate levels of detail.
- Design does not use privilege separation, so all code in the entire application is potentially security critical.
- Unhelpful or misleading comments in the code.
- Potentially complex data flow due to exception handling.
- Subjectively, large amount of source code compared to the functionality implemented.

The code constructs used were generally straightforward and easy to follow on a local level. However, the lack of design documentation made it difficult to globally analyze the system.

SUMMARY TABLE OF SECURITY TESTING FINDINGS

			Attacker						Vulnerability Assessment					Total
Ref	Title	Component	Voter	Poll worker	Election official	Storage Personnel	Vendor	Scalability	Time	Expertise	Knowledge	Window of Opportunity	Equipment	Attack Resistance
	Source Code													
A.1	Evidence Documents	All		X	X	X	X	N/A	Document Review only					N/A
A.2	SOAP Encryption	PBC			X	X	X	N/A	Document Review only					N/A
A.3	Linux Hardening	PBC		X	X	X	X	High	0	3	0	4	0	Basic
A.4	Configuration Management	All					X	Undet	Document Review only					N/A
A.5	Audio Ballot Aid	EMS, Audio Ballot Aid		X	X	X		High	0	6	3	0	0	Basic
A.6	Low Key Entropy	All			X	X	X	High	0	3	3	1	0	Basic
A.7	Weak Master Key Algorithm	All			X	X	X	High	0	0	3	1	0	Basic
A.8	Symmetric Cryptography	All			X	X	X	High	0	3	3	1	0	Basic
A.9	Zip Directory Traversal	PBC			X	X		High	0	6	3	1	0	Enhanced
A.10	SQL Injection Password	EMS			X			High	0	3	3	1	0	Basic

<u>Legend for the Summary Table of Security Testing Findings</u>

Vulnerability Assessment Coding:

1. Time to Exploit. "…total amount of time taken by an attacker to identify that a particular potential vulnerability may exist in the TOE, to develop an attack method and to sustain effort required to mount the attack against the TOE. "[CEM v3.1, App B] "TOE" is the target of evaluation.

2. Expertise. "…the level of generic knowledge of the underlying principles, product type or attack methods "[ibid]

3. Knowledge of Target of Evaluation (TOE). "…specific expertise in relation to the TOE."[ibid]

4. Window of Opportunity. "…equate to the number of samples of the TOE that the attacker can obtain. This is particularly relevant where attempts to penetrate the TOE and undermine the SFR may result in the destruction of the TOE preventing use of that TOE sample for further testing, e.g. hardware devices"[ibid]. For this test, the Window of Opportunity includes limitations on accessing a specific feature which has significance to the security of the system. "SFR" is "Security Functional Requirement" which is a member of set of formally, predefined security requirement in the Common Criteria standards that are used as a basis for interpreting and testing security requirements for a TOE.

5. Equipment, hardware/software or other. "…the equipment required to identify or exploit a vulnerability "[ibid]

"Table 3, Calculation of Attack Factor" [ibid]

Factor	Value
Elapsed Time	
≤ one day	0
≤ one week	1
≤ two weeks	2
≤ one month	4
≤ two months	7
≤ three months	10
≤ four months	13
≤ five months	15
≤ six months	17
> six months	19
Expertise	
Layman	0
Proficient	3*(1)
Expert	6
Multiple experts	8
Knowledge of TOE	
Public	0
Restricted	3
Sensitive	7
Critical	11
Window of Opportunity	
Unnecessary / unlimited access	0
Easy	1
Moderate	4
Difficult	10
None	**(2)
Equipment	
Standard	0
Specialized	4(3)
Bespoke	7
Multiple bespoke	9

(1) When several proficient persons are required to complete the attack path, the resulting level of expertise still remains "proficient" (which leads to a 3 rating).

(2) Indicates that the attack path is not exploitable due to other measures in the intended operational environment of the TOE.

(3) If clearly different test benches consisting of specialized equipment are required for distinct steps of an attack, this should be rated as bespoke"

"bespoke" is specified when "…clearly different test benches consisting of specialised equipment are required for distinct steps of an attack"[ibid].

"Table 4, Ratings of vulnerabilities and TOE resistance"[ibid]

Values	Attack potential required to exploit scenario:	TOE resistant to attackers with attack potential of:
0-9	Basic	No rating
10-13	Enhanced-Basic	Basic

14-19	Moderate	Enhanced-Basic
20-24	High	Moderate
≥ 25	Beyond High	

As an example, the PBS Physical Access attack described in the Red Team Report Item A.1 can be done by anyone with access to the PBS (poll worker, election official, storage worker, or vendor) in less than 20 minutes (≤ one day). The attack requires some skill with the locks and seals (Proficient) and knowledge of what the seals protect (Restricted) to be effective. Windows of Opportunity are somewhat limited (Moderate) because other observers would be expected to respond but the tools were common to home and office use (Standard). The resulting vulnerability to access (total of the factors=10) barely qualifies as Enhanced-Basic, which implies that the attack would require more than a casual event.

In contrast, the CD clear text attack (Red Team Report Item A.8) requires information gained through experience with the system and system documentation (Knowledge of TOE=3) and some common software to review the file contents but does not require additional time (≤ one day) or special tools (Standard). Some knowledge of the system to recognize the files and clear text contents are needed (Restricted) but the clear text may be read by a layman (Layman). The Window of Opportunity requires getting a copy of the CD (Easy). This gives a total vulnerability risk of Basic (Total of factors = 4).

These two examples are both foundation attacks that support other attacks by opening accesses and acquiring Knowledge of TOE that may be used in other attacks.

Public Report prepared by:
Steven V. Freeman, Senior Partner, Freeman, Craft, McGregor Group

atsec reviewers:
Klaus Weidner, Principal Consultant, atsec information security [Lead Reviewer]
Stephan Műller, Principal Consultant, atsec information security