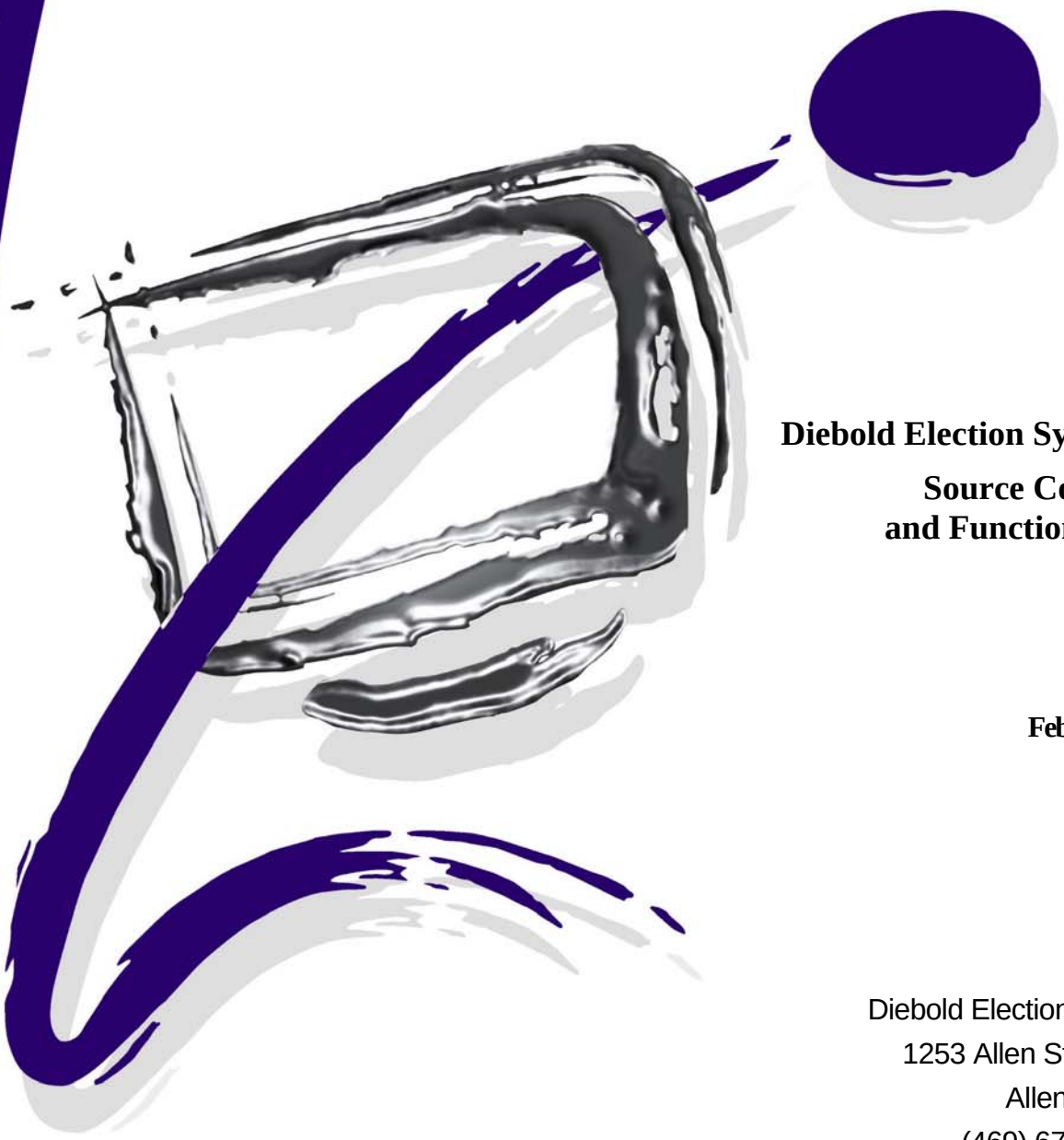

Diebold Election Systems, Inc.

Source Code Review and Functional Testing

February 23, 2006

Prepared For:

Diebold Election Systems, Inc.
1253 Allen Station Parkway
Allen, Texas 75002
(469) 675-8990 (office)
(214) 383-1596 (fax)

ciber
ALWAYS ABLE

Prepared By:
CIBER Huntsville and
CIBER's Global Security Practice

CIBER, Inc.
7501 South Memorial Pkwy
Suite 107
Huntsville, AL 35802
256.882.6900

Table of Contents

REVISION HISTORY

Date	Version	Description	Author
02/23/2006	V1.0	Document Creation	CIBER Huntsville and CIBER's Global Security Practice

INTRODUCTION

BACKGROUND

Diebold

Diebold Election Systems, Inc. retained the services of CIBER Huntsville and CIBER's Global Security Practice professionals in order to determine the degree to which the Diebold Election Systems compiler, interpreter, and script source code and functionality comply with security industry and coding best practices and are not vulnerable to compromise per the following order:

"The Diebold AccuVote-OS (AV-OS) optical scan and AccuVote-TSX (TSX) touchscreen voting systems contain a report writing facility in which scripts, written in a proprietary high-level AccuBasic programming language, are first compiled into a low-level tokenized language, and then the tokenized code is interpreted using an interpreter module in the firmware of the AV-OS or TSX unit. The safety of the interpreted code is a vital security issue, so it is important to verify that it is not possible to compromise an election in any way through the (mis)use of AccuBasic, *including an unintentional error or malicious AccuBasic script*." [Reference: Request for ITA Review of Diebold AccuBasic Interpreter, Compiler, and Scripts.pdf]

CIBER's Global Security Practice

CIBER® (NYSE: CBR) is a leading international system integration consultancy with superior value-priced services for both private and government sector clients. CIBER's services are offered on a project or strategic staffing basis, in both custom and enterprise resource planning (ERP) package environments, and across all technology platforms, operating systems and infrastructures.

CIBER's Global Security Practice, focuses exclusively on information security. Its professional staff designs, implements, and manages security solutions for critical information systems in a wide range of commercial and Federal environments. Since 1992, organizations desiring superior security engineering and consulting services have turned to CIBER Security to fulfill their information security needs.

OVERVIEW AND APPROACH

The CIBER Huntsville and CIBER Global Security teams were tasked with performing a combination of testing and analysis of the Diebold Election System's Source Code to identify security and functionality vulnerabilities. The testing was structured to identify and evaluate as much potential vulnerability as possible within a reasonable/controlled level of effort.

The engagement was comprised of four phases that were collectively designed to provide a comprehensive inspection of the source code using primary and secondary development requirements as the evaluation metric. The phases were:

Phase 1: Portions of the code specific to the interpreter were assessed. Manual reviews broke up interpreter code into that which is specific to the AV-OS, and that which is specific to the TSX. For each interpreter code set, nine (9) primary requirements, three (3) of which contained a total of 12 sub-requirements, were reviewed.

Phase 2: Portions of the code specific to the compiler were reviewed. Four (4) primary requirements were reviewed for this purpose.

Phase 3: CIBER's team conducted manual testing of the AccuBasic scripts, for which two (2) primary requirements were relevant.

Phase 4: Code was reviewed for other forms of potential risks, hazards, or vulnerabilities *not* relative to any of the specified primary or secondary requirements.

REFERENCE MATERIALS

The following reference materials were used for the source code review:

1. ABasic 2-1-9 to 2-1-9-1 Change Release Summary
2. ACCU-BASIC User Guide Release 1.92
3. TSX AccuBasic Interpreter Source Code
4. AV-OS AccuBasic Interpreter Source Code
5. AccuBasic Compiler
6. Provided Scripts written in AccuBasic

EXECUTIVE SUMMARY

The TSX interpreter inspected appears to be ready for an election. The AV-OS interpreter inspected appears to be sufficiently secure to run an election if the recommended corrective measures are applied to the interpreter and rechecked. If trusted chain-of-custody were established to prevent tampering with memory cards between the GEMS system and the AV-OS voting machines, then the existing units would be safe for an election.

The fact that the programs appear to provide adequate security shall not be interpreted to mean that the programs are without security vulnerabilities or are impenetrable. It does mean that the programs appear to provide reasonable assurance that it can protect the confidentiality, integrity, and availability of the information it processes, stores, and communicates.

It is standard practice at CIBER to provide recommendations in addition to review findings. In addition to the recommendations that will be placed throughout this report, one high-level recommendation is provided:

- Certain vulnerabilities in this report *may* require a portion of the code to be modified in order to correct the vulnerabilities identified. To ensure that the efforts to correct vulnerabilities do not introduce new vulnerabilities, CIBER strongly recommends retesting of the remediated code prior to its migration to a production environment.

The interpreter had three security vulnerabilities and a small number of requirement violations that were not capable of being exploited by malicious code or operators. Of the three serious problems, they can be fixed with minor code changes.

No issues were discovered with the compiler that impacts the security of the system. There were no findings in the inspection of the AccuBasic Scripts that would materially impact the security of the system.

INTERPRETER ASSESSMENT

OVERVIEW

On December 28th, Diebold Elections Systems, Inc. requested that CIBER perform a source code review of Diebold's AccuBasic Interpreter. From the statement of work, the purpose of this task was to identify the validity of the following statement:

"The safety of the interpreted code is a vital security issue, so it is important to verify that it is not possible to compromise an election in any way through the (mis)use of ABasic, including an unintentional error or malicious ABasic script."

Nine primary and twelve secondary requirements were identified as essential. Along with the presented requirements, CIBER was encouraged to provide additional, experience-related concerns or comments as appropriate.

To minimize assumptions made in the code review, the review was performed solely on the provided materials and their ability to handle both normal and malicious use, even if the malicious behavior came from any point outside of the code base.

The AccuVote-OS Interpreter was provided in the form of two files. The TSX Interpreter was provided in the form of three files. Microsoft Visual Studio 2003 was used to examine the software but was used only for text viewing and for reference searching. CIBER inspected the source code on a function-by-function basis, as well as used Internet references as needed for validation of the functionality of commands from external libraries.

The Interpreter source code that was provided was not a stand-alone application and the programs that call the interpreter were not present. The calling programs were deemed "not trusted" and capable of providing invalid input in the course of this review. A series of requirements were given that were used to evaluate the security and functioning of the code.

GENERAL INTERPRETER REQUIREMENTS

General requirements for review of the interpreter necessitate that the interpreter address those properties defined in the following sub-sections.

Report Writing Functionality

Requirement: The language and interpreter are used only for the report-writing functions of the AV-OS and TSX.

AV-OS Finding: No violations were found. A total of six locations were found in the AV-OS source code where the interpreter is called, all for the purpose of printing a report.

TSX Finding: No violations were found. One location was found in the TSX source code for calling the interpreter that was written for the sole purpose of printing reports.

Special Purpose Functionality

Requirement: The language and interpreter do not constitute a general purpose embedded programming system, but are instead extremely limited and special-purpose in their capability and linkage to the rest of the firmware environment, and possess essentially no more capability than is required for their report-writing function.

AV-OS and TSX Finding: No violations were found. The compiler is separate from the interpreters and not included on the Diebold voting units. The interpreters have extremely limited functionality for the creation of a report and the interpreter cannot change its own parameters of execution. The Interpreter cannot create or write to any files and only sends information to the printer and the display.

Program Behavior

Requirement: No program in the compiled, tokenized form (whether or not generated by an AccuBasic compiler) can ever, directly or indirectly,

a. Modify any votes or vote counters,

b. Modify the electronic audit trail, or

c. Cause any other behavior that might compromise the integrity of an election, even if the code is maliciously engineered, and even if election procedures have not been properly followed.

AV-OS and TSX Finding: Three violations exist that allow manipulation and reading of data in global space. Three different types of modified tokens used to index data outside of their intended memory range cause the vulnerabilities, each with slightly different effects. These can only be exploited by a modified AccuBasic object file.

It is quite possible that these exploits can be used in conjunction with each other in a way to produce an escalation of privileges, depending on the operating environment and the compiler settings. The evaluation team confirmed the flaws are present and considered dangerous, but proof-positive exploit for an escalation was not possible without access to a working development environment and appropriate development software.

The TSX environment contains a check to validate the AccuBasic object files, so if a file is tampered, the tampering will be detected. Therefore, this problem is more severe for AV-OS than it is for TSX. TSX can still be considered election ready because such tampering will be detected.

SPECIFIC INTERPRETER REQUIREMENTS

Specific requirements for review of each of the two interpreters necessitate that each interpreter address those properties defined in the following sub-sections.

Report Printing

Requirement: Verify the interpreter is not called or executed when printing a report or the sequence of question and response selecting the options and sequence of the report(s).

AV-OS and TSX Finding: No violations discovered. The interpreter itself does not have an ability to detect multiple instances of itself being run, however the rest of the source code indicates that the voting software does not support multi-threading per code instance, and only one instance of a running program is allowed at a time.

Bounded Stack and Memory

Requirement: Verify that it is not possible for the interpreter to write to main memory outside the bounds of its own stack and memory. As part of this, verify that

a. The interpreter's stack (and heap if present) are bounded, so that they cannot overflow their fixed area without triggering an exception;

b. The AccuBasic language itself does not provide for dynamic storage allocation, nor any use of pointer-like variables or variable offsets and hence such language constructs cannot be misused to write main memory arbitrarily;

c. The only indexed data structures supported in the AccuBasic language are strings (not arrays), and they can only be *read* through the index, but not *written*, so that in particular it is not possible to overwrite main storage arbitrarily using an out of bounds index in a string assignment;

d. String and substring copy operations are protected in the interpreter so no buffer overflows can occur, as a result of copying a big string into a small buffer.

TSX Finding: The bounds checking on the stack and heap segments were not detected, although it may have been outside the scope of the assessment area for this team. Bounds checking is performed inside the code and can be used to prevent overflows. Also, a "stack guard" (or non-executable stack) software package may help lower the risk. The lack of non-executable stack software does not imply that a malicious overflow does exist, nor does the presence of one guarantee that an overflow attack cannot still work.

Both: A previously mentioned violation for "Program Behavior" demonstrates ability for a maliciously constructed file to access global memory.

A minor violation of the requirements exists in that arrays do exist in AccuBasic, but can only be seen and used through an external data record, and have bounds checking. This data cannot be written to. Therefore, the arrays are not exploitable for out-of-bounds memory attacks. Bounds-checking does exist at the I/O perimeters to prevent buffer overflows as per the requirements. All memory used by the interpreter is fixed in length, cannot be dynamically allocated, and is bounds-checked to prevent access to main memory.

Operating System Services

Requirement: Verify that the interpreter is extremely limited in the operating system services it requests (I.e., to the small number needed) to

a. Request a yes/no input from a user,
b. Write a report to the printer,
c. Print message to the LCD screen (or touchscreen)
d. Append audit trail entries, and
e. Retrieve date/time.

AV-OS and TSX Finding: The I/O is limited to the requirements as stated and has no violation of this requirement.

Method Invocation and Data Structures

Requirement: Verify that the interpreter does not invoke other methods or reference other data structures in the firmware codebase (outside the interpreter) except as required to perform its report-writing function, and that all such references are read-only. In particular, verify that the interpreter cannot

a. Write any variables outside its own memory,
b. Modify files on any file system (in the TSX),
c. Launch any application programs.

AV-OS and TSX Finding: No violations were found of this requirement. Both versions of the interpreter do not support the ability to write to external variables

or files and have no provision or function that allows for the launching of any application programs.

Exception Handling

Requirement 1: Verify that the interpreter properly fields and properly handles all possible exceptions that can be generated in the AccuBasic language, especially string bounds errors, zerodivides, integer overflows, stack overflows and underflows, etc.

AV-OS and TSX Finding: No violations discovered. Appropriate error checking was made to catch and handle all forms of standard interpreter errors that come from the provided scripts.

Requirement 2: Verify that any exceptions generated by the interpreter's own execution are all properly caught and handled as well, including such exceptions as heap allocation failures, end-of-file, file access errors, I/O errors, volume overflow (for the volume containing the audit trail), etc.

Both: No violations were discovered. The critical points of failure, such as the reading of the object files and opening I/O streams to the printer, are checked for errors. Writing typically isn't checked for failure but, because they are open-ended streams and buffered, an error won't trigger even if the code checks for them.

Script Operations

Requirement: Verify that the script operations terminate. That is, that all explicit and implicit looping or recursive operations have a definitive exit and the exit condition which will be achieved in a limited number of passes and that any exception handling or conditions result in an exit and return to the system operations and are not hung up in a wait state unless the wait is conditioned by an alert and visible response request to the control screen.

AV-OS and TSX Finding: No violations were discovered. A definitive exit condition and exit exists for all functions that exist in both versions of the interpreter as well as for all functions called by the interpreters.

Halting

Requirement: Verify that the interpreter does not halt except at the appropriate time.

AV-OS and TSX Finding: No violations of this requirement were discovered. All detected failures will result in error codes being propagated to the calling point of the interpreter and passed to the calling program. There were no clear bugs discovered in the code where the system would suddenly halt based on normal usage.

Infinite Loops

Requirement: Verify that the interpreter does not infinitely loop (unless interpreting an AccuBasic script that contains an infinite loop at that level).

AV-OS and TSX Finding: No violations of this requirement were found, all loops in both interpreters have a clearly defined beginning and end point with an appropriate index that increments so that an end-point is always reached.

INTERPRETER – OTHER FINDINGS

Requirement: While the above requirements are intended to give explicit guidance for the conduct of the review, the ITA (I.e., CIBER) is also expected to apply due diligence in reporting other potential risks, hazards, or vulnerabilities in the review.

Both: Error handling appears to be adequate for a system that executes in a perfect running environment. However, the interpreters do not have the proper degree of error checking to identify or recover from key failures in a damaged, altered or dysfunctional environment.

Often it is difficult to tell the difference between a 'hack' and a system failure for an operator. Programs that have high-assurance capabilities will provide additional information about sources of errors that can be useful to avoid negative speculation and recover faster in the advent of a catastrophic failure.

Our reasoning for increasing the security on the code is because the object code traverses potentially untrustworthy hands in the process of its distribution of scripts from the GEMS to the interpreter. Since the object code is on the memory cards being distributed, it is a prime target for potential tampering.

AV-OS Finding: The error codes returned by the interpreter to the AV-OS system are ignored. Although this isn't a security violation, it would assist as being a validation of the procedures for if a problem does occur.

COMPILER ASSESSMENT

OVERVIEW

Four source code files and a Microsoft Visual C++ 6.0 Project file were submitted for code review of the AccuBasic Compiler. Microsoft Visual Studio 2005 was used to perform this source code review. The AccuBasic Compiler was compiled and built in a Visual C++ environment.

Note: *The User Guide is Release 1.92, and was revised on August 31, 1994. However, the source codes currently in review are Version 1.95.2, and are dated as 2005/12/18.*

SPECIFIC COMPILER REQUIREMENTS

Requirements for review of the compiler necessitate that the compiler address those properties defined in the following sub-sections.

Commands, Operators, Functions, and Files

Requirement: Verify that the compiler does not support additional AccuBasic commands and access methods not documented in the AccuBasic programming language manual.

Finding: No security violations were discovered. There were some inconsistencies between the manual and the provided functionality, although the differences are not malignant. Our recommendation is to update the manual to reflect the improvements.

Additional Code

Requirement: Verify that the compiler does not interject additional code beyond what is specified in the AccuBasic report file source code scripts.

Finding: No security violations were discovered. The AccuBasic Compiler parses source code scripts. During the compilation, the compiled information was stored in the internal memory buffer. At the end of compilation, the Compiler writes the internal buffer into the output file to generate the object file.

There are 79 instances in which the Compiler writes the internal memory buffer. Besides the code (commands, functions, files, and etc.) specified in source code scripts, the Compiler also writes the series number into an object file. The Compiler also inserts internal flags into an Object file.

This extra information is necessary to maintain the integrity of the object files. Since they are very short strings and single-character flags, it is very unlikely that this kind of information will be interpreted as meaningful binary code when loaded by the firmware.

COMPILER – OTHER FINDINGS

Requirement: While the above requirements are intended to give explicit guidance for the conduct of the review, the ITA (I.e., CIBER) is also expected to apply due diligence in reporting other potential risks, hazards, or vulnerabilities in the review.

Findings: No security violations were discovered. The compiler is largely immune to security issues because it is a stand-alone system and never leaves the development environment. Undocumented features discovered were all non-malicious and did not extend the capabilities of AccuBasic in any way.

ACCUBASIC SCRIPTS ASSESSMENT

OVERVIEW

There were 19 AccuBasic script files subjected to review. Microsoft Visual Studio 2005 was used to perform this source code review, but only served as a text file reader due to the nature of the files.

SPECIFIC SCRIPT FILE REQUIREMENTS

Requirements for review of the scripts necessitate that the scripts address those properties defined in the following sub-sections.

Infinite Loops

Requirement: Verify, to the extent possible, that none of the scripts will infinitely loop under any conditions.

Finding: No infinite loops are contained within any of the scripts when the environment is running as designed.

Binary Code

Requirement: Verify that none of the scripts contain character string constants or other data that the compiler would include in the AccuBasic object files, but that might be interpretable as binary code when loaded by the firmware.

Finding: While certain values may be defined as binary, they are very short strings and single-character flags that are very unlikely to be interpreted by the firmware as meaningful binary code.

SCRIPT FILES – OTHER FINDINGS

Requirement: While the above requirements are intended to give explicit guidance for the conduct of the review, the ITA (I.e., CIBER) is also expected to apply due diligence in reporting other potential risks, hazards, or vulnerabilities in the review.

Finding: No additional security problems were discovered. However, inconsistencies were found in coding style and documentation. These include default values that were not found in the user's guide, default values defined in the source code and malign undocumented functions. We recommend updating the documentation to keep configuration control.