Taking Total Control of Voting Systems: Firmware Manipulations on an Optical Scan Voting Terminal

Seda Davtyan Sotiris Kentros Aggelos Kiayias Laurent Michel Nicolas Nicolaou
Alexander Russell Andrew See Narasimha Shashidhar Alexander A. Shvartsman

Voting Technology Research Center and Computer Science and Engineering Department
University of Connecticut, Storrs, CT 06269, USA

{seda,skentros,aggelos,ldm,nicolas,acr,andysee,karpoor,aas}@cse.uconn.edu

ABSTRACT

The firmware of an electronic voting machine is typically treated as a "trusted" component of the system. Consequently, it is misconstrued to be vulnerable only to an insider attack by someone with an in-depth knowledge of the system and access to the source code. This case study focuses on the Diebold/Premier AccuVote Optical Scan voting terminal (AV-OS) that is widely used in the USA elections. We present three low level manipulations of the above voting terminal's firmware resulting in divergence from its prescribed operation: (i) the first bestows the terminal with a powerful memory card dumping functionality, (ii) the second enables the terminal to leak the ballot details through its serial port thus violating voter privacy during the election, (iii) the final third firmware manipulation is a proof of concept attack that swaps the votes of two candidates thus permanently destroying the election outcome in an undetectable fashion. This demonstrates the extent to which the firmware of the AV-OS can be modified with no insider knowledge or access to the source code.

Our results underscore the importance of verifying the integrity of the firmware of electronic voting terminals accompanied by sound auditing procedures to maintain the candor of the electoral process. We also note that this work is performed solely with the purpose of security analysis of AV-OS, and the first and the second firmware manipulations we describe serve a dual purpose in assisting the technological audits of actual voting procedures conducted using AV-OS systems.

1. INTRODUCTION

Frequently, the firmware component of a proprietary computing system is misconstrued to be vulnerable only to a technical insider attack by someone with extensive resources, an in-depth knowledge of the targeted system, and access to source code and/or hardware specifications. In this work, we challenge this assumption in the domain of e-voting machines by presenting a detailed case study focusing on the AccuVote Optical Scan Terminal (AV-OS) that is widely used in the US elections. Our investigation demonstrates the possibility of an array of firmware manipulation attacks against the AV-OS that are characterized by their low cost (<$300) and the fact that they were developed without any access to hardware specifications or the source code of any software running in this system. By virtue of being a firmware modification, our attacks are totally immune to cryptographic integrity checks or any other type of auditing procedure (aside from an auditing procedure that inspects the firmware itself, which is a type of audit whose importance is strongly underscored by our work).

Our results include three firmware manipulations applied to the AV-OS. Our first manipulation equips the voting terminal with a powerful memory card dumping functionality. The second firmware manipulation leaks the ballot contents through the serial line when a voter scans his/her ballot. Finally, we present a proof of concept malicious attack to illustrate the dangers of trusting un-audited firmware. The third firmware alteration is capable of swapping two election counters thus permanently destroying the outcome of any election that relies solely on the machine counts.

We note that this work is performed solely with the purpose of security analysis of AV-OS and to assist in the audits of elections conducted using AV-OS systems. The various firmware manipulations find a dual purpose in the auditing process and the basic techniques developed herein are useful tools that can expedite the auditing procedures for voting terminals.

Optical Scan Security Vulnerabilities: There are two major types of electronic voting equipment: Direct recording electronic (DRE) machines and optical-scan (OS) machines. An important benefit of the optical scan technology is that it naturally yields a voter-verified paper audit trail (VVPAT)—the actual "bubble sheet" ballots marked by the voters that enables hand-counted audits and recounts. Despite the paper trail benefit of OS terminals, three major issues serve as the motivation of this paper: First, OS machines, such as the AV-OS terminal, have a loadable software component containing information about the election and candidates. Such software presents opportunities for attacks [5, 2, 8] and should be audited. Second, OS machines may occasionally make errors when reading ballot sheets, as indicated by hand-counted audits [10, 9]. Such audits are performed randomly in some states and may be triggered by a tightly contested race. Unfortunately, such audits are

SAC'09 March 8-12, 2009, Honolulu, Hawaii, U.S.A.

time consuming and may themselves be error prone [10, 11]. The third issue concerns the firmware, the software component that implements the basic functionality and typically resides in the internal memory of the terminal. Generally, the firmware of voting machines is trusted implicitly. In some cases the firmware source code is inspected by independent experts [8, 6] to ensure that it behaves as expected, however, to the best of our knowledge, the actual firmware binary code is currently not audited.

In [5], the authors describe a general architecture that characterizes special purpose trusted computing devices such as the AV-OS voting terminal. Recall that a system like the AV-OS is composed of both software and hardware components. From the security viewpoint, the key features are the components that are accessible or easily replaceable from outside the system, namely, the software components. The main theme in [5] (as well as in most previous research) centered around the vulnerabilities and security issues of the AV-OS voting terminal that relate to the machine's proprietary language, called `AccuBasic`, used for reporting election results. For the most part, previous work assumes that the firmware of the AV-OS terminal is a trusted component of the system, a point from which we depart in the present work.

Contributions: We present three manipulations of — in essence attacks on — the AV-OS firmware. First we implement a new diagnostic memory card dumping function of the terminal; in particular, we develop a firmware manipulation that expedites the memory card dumping functionality of the AV-OS terminal; our modified firmware exhibits a sixfold speedup of the total dumping time. This firmware manipulation was performed to assist in the auditing of the contents of memory cards but can also be instrumental in launching an attack against the voting terminal since the memory card stores all the sensitive information regarding the election and expedient access to this information can be essential to an attacker. The second manipulation targets vote privacy: our modified firmware "leaks" through the serial port the actual contents of a voters' ballot. The dual purpose of this firmware manipulation is assisting in the hand-counted audits of the election outcomes. Indeed, in the post-election stage this modification of the firmware allows auditors to visually inspect (on screen) the interpretation of the ballots by the AV-OS terminal. Such function can be used to enhance an optical scan terminal with some of the functionality of a DRE voting terminal that provides visual cues on the machine's interpretation of the voters' input (such affirmation is missing from optical scan voting terminals but is present in DRE machines). Lastly, we demonstrate a malicious proof-of-concept attack based on firmware manipulations. The modified firmware tests to see whether a set of right conditions are met and then swaps the vote counters of two candidates – something that permanently destroys the election results (to the degree they rely on the output of the machine). This attack, being at the firmware level, is immune to any cryptographic integrity checking that is employed at the memory card level (recall that the firmware should in fact be capable of modifying the contents of the memory card and is ultimately the component responsible for enforcing integrity checks).

At no time did we have access to source code or internal documentation, showing that the above attacks can be carried out by anyone with a reasonable computer science background. The proof of concept attacks illustrate the dangers of trusting un-audited firmware and show the importance of verifying the integrity of the firmware of EVMs accompanied by sound auditing procedures to maintain the candor of the electoral process.

2. THE AV-OS ELECTION SYSTEM

The AV-OS election system is comprised of two components [3]: the AccuVote Optical Scan voting terminal (AV-OS) and Global Election Management System (GEMS).

The AV-OS terminal is a computing device responsible for accepting ballots and tabulating the results of the election. The functionality of the terminal is determined by the firmware loaded into a single Erasable Programmable Read Only Memory (EPROM) residing inside the terminal. The AV-OS voting terminal described in this paper contains the firmware version 1.96.6. The major hardware components include an 8MHz microcontroler which emulates an Intel 80186 processor, 128K RAM chips, an optical scanner, a dot-matrix printer, LCD display, serial communication port, built-in modem and a removable 40 pin EPSON memory card. An analysis and discussion of the contents of the memory card is presented in this paper.

GEMS is the ballot design and central tabulation system. It is installed on and operated from a conventional PC. GEMS consists of several databases that include the data and ballot layout corresponding to the precincts participating in the election, and the bytecode that determines how the tabulated results are printed. This data is transferred via the serial communication port to the memory card kept in the AV-OS terminal.

2.1 AV-OS Software Components

The behavior of each AV-OS terminal is determined by the *firmware* and the *memory card* contents.

Firmware. The main software component of the AV-OS is its *firmware*, executable code kept in a 128K 32 pin EPROM chip (M27C1001) and responsible for all the functions provided by the machine. The EPROM is electronically programmable and UV (ultra-violet) light erasable. Using appropriate (inexpensive) tools, we were able to extract the binary code of the firmware and disassembled that into a friendlier assembly-level[1] representation for examination and with the purpose of security analysis.

Our analysis reveals that the firmware consists of *two segments*. The first is at the beginning of the EPROM's memory space. "Far" calls lead to the starting address of the second segment and facilitate the interaction between the two segments.

Thereafter, utilization of I/O strings, program flow controls, procedural structure and register operations, allowed the identification and modification of the firmware. Interestingly, deployment of the *customized* firmware, required nothing more but burning the modified binary code onto a programmable EPROM and installing the new EPROM into the AV-OS; this procedure is accomplished in minutes given physical access to the machine.

Memory Card. The AV-OS terminals contain a 40-pin 128KB Epson memory card. It is installed into the slot via

[1]Dissasembly according to the 80186 architecture since the actual processor in the AV-OS (part number NEC D70320L-8) emulates an Intel 80186 processor.

a J40 connector found in the right front side of the terminal. Worth mentioning is that Epson discontinued the production of this memory card, and reader/writers for this memory card are not readily available.

The data on the card includes status information, an audit log, ballot description, and counters, described in more detail in Section 3. This information was extracted in part by the systematic analysis of the firmware's binary code and in part by "eavesdropping" on the communication between GEMS and AV-OS. Note that the analysis was performed without any technical documentation from the vendor and in the absence of the firmware source code. The memory card format is shown in Figure 1.

3. GAINING ACCESS CONTROL

In the process of this work it became apparent that we could affect, replace, or modify all aspects of the firmware. A first step of our security analysis presented below was to gain control over the following components: (1) the memory card access and analysis, and (2) the serial port access and one way communication from the AV-OS terminal to GEMS (and, in effect, any external PC). With this new control it becomes possible to design and embed software (benign and malicious) on the AV-OS hardware.

Memory Card Access Control. Perhaps the most important step in the security analysis of the AV-OS terminal was understanding and interpreting the contents of the 40-pin EPSON memory card. Since the election data and the election flow control are stored in the memory card, it is imperative to unveil the importance and use of every single byte on the card by the terminal. At first we identified the memory card address from the firmware. This helped us to trace the conditions under which the terminal accesses the memory card and identify the purpose of most of the card's bytes. The organization of the card appears in Figure 1. Below we summarize the data found in each section.

Header: The header contains information about the organization of the contents of the card, description of the election and flow control flags. The header has a fixed length, totaling 576 bytes.

Log: This segment is a fixed-size "circular" buffer in which the firmware logs certain timestamped events. It can hold at most 512 entries—any log additions beyond this limit overwrite entries in an earliest-first fashion. Each entry consists of 2 bytes, totaling 1024 bytes of log. Further analysis and parsing of the log reveals information about the history and the actions taken on the AV-OS terminal that hosted the analyzed memory card.

Election Data: The election data obtained from the GEMS database is kept several bytes below the log end. The election data segment has a variable length and can be broken down into three subsections: a) *Ballot Data* that reveal information about the ballot layout, b) *Race Data* which include the details about the participating offices, and c) *Candidate Data* which contains the details about the election candidates.

Bytecode: The AccuBasic (AB) bytecode present in the programmed memory cards is solely responsible for the reporting procedures. The code is written in a proprietary symbolic language that was decompiled and analyzed.

Election Counters: The election counters are located below the bytecode on the memory card as illustrated in the schematic in Figure 1. Here all the election results and statistics are stored. This section can be divided into two broad subsections: a) *Race Counters* and b) *Candidate Counters*.

Serial Port Access. We then aimed to control the communication between the AV-OS terminal and an external PC. This would significantly enhance the analysis and processing of the terminal's data, utilizing the power and tools developed on an external PC. For this purpose we identified the firmware's specialized procedure for transmitting bytes to the serial port. Then, guided by the general reference on the Intel-derived processor chip of the AV-OS [7], we first identified key memory-mapped registers used in the procedure and responsible for the interaction between the processor and the serial line. According to the use of those registers, the main idea of the transmission is to unmask and trigger the transmission interrupt which in turn will place the byte to be sent from the sending buffer on the wire. Due to space limitations we omit the technical details in this paper and we summarize the AV-OS transmission methodology in the following four steps: (1) Discovery and reservation of the sending buffer, (2) Byte to be sent is uploaded to the sending buffer, (3) Address of the Serial interface is loaded, and (4) Unmask the transmit completion interrupt to transmit the byte from the send buffer on the wire.

4. ACCURATE MEMORY CARD AUDITING

The firmware in the AV-OS terminals provides a function to export the contents of its 40-pin memory card through the serial line. We refer to this function as *memory card dumping*. Our election audit procedure uses this capability of the machine to test the contents of the memory card (containing the election data) against election data provided by a trusted party (for more details see [9]). For reasons given below, a new memory dumping function was essential to ensure the reliability, accuracy, integrity and efficiency of an election audit procedure. The firmware-resident memory dumping function has several deficiencies:

1. There is no indication on whether AV-OS faithfully dumps the contents of the memory card.
2. Using the native firmware may cause unintended changes to the contents of the memory card, interfering with the ability to recover the original contents of the card.
3. The memory card dumping selectively filters out some characters from the contents of the card affecting their reliable extraction and reproduction.
4. The dumping of the card takes a relatively long time, making it unfeasible to examine hundreds or even thousands of cards considered in a typical audit.

Ultimately, our objective was to enable reliable delivery of the data from the card to an external receiver via the serial port with no side-effects. Armed with the knowledge presented in Section 3 we were able to access the memory card, extract its content and transmit it unmodified through the serial line. To hasten the data extraction process we implemented a very simple form of data compression based on *run-length encoding (RLE)*.

Delivery of the Memory Card data: A clear understanding of the address space layout and content as well as how to access the memory card enabled us to write a simple routine that reads the data from the memory card.

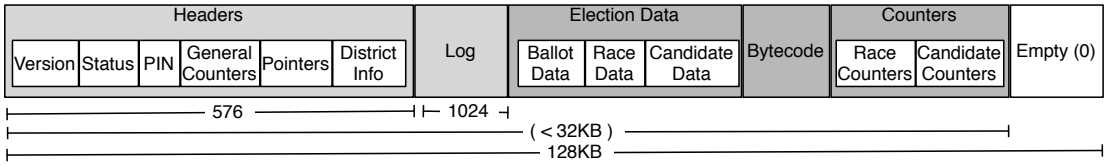


Figure 1: Format of the AV-OS Memory Card.

The next task was to ensure that the bytes read from the memory card were faithfully transmitted using the serial line one-way communication from the terminal to an external serial line receiver (Section 3). The new procedure simply accepts the byte to be send on the wire, places that byte in the sending buffer and unmask the transmission interrupt which triggers the transmission over the wire. No other control, meta-data or arbitrary bytes are transmitted on the wire. The new firmware was tested on the transmission of memory cards with known content to confirm the fidelity of the transmission.

Speeding up memory card dumping: Because of the low transmission rate of the serial port (9600bps = 1KB/s) of AV-OS, dumping the contents of the 128KB memory card takes a significant amount of time (> 2 *min*), burdening the inspection of a large number of cards (i.e., many hundreds of cards). To solve this problem we enhanced the simple transmission protocol with a straightforward *run length encoding* algorithm to compress the bytestream (and decompress it on the receiving end). This compression reduced the dump time to 20 *sec* per memory card. (Simple run-length encoding works well here because several large parts of the memory card are sequences of identical values.)

Avoid unintended alteration of the memory cards: The final task was to ensure that no information is added to the memory card or altered during the dumping process – such changes would be unacceptable in an audit procedure. Since our software now has control over the dumping process and communication from the end of the machine boot sequence, we can make sure that the card contents are not altered. Thorough testing also verified that the boot/initialization code alters neither the log nor any other data on the memory card. Thus the entire procedure gets a faithful image of the card and does not alter its contents.

5. LEAKING THE BALLOT CONTENTS

As mentioned in Section 1, one issue with optical scan machines is that they may occasionally make errors, as suggested by hand count audits such as that described in [9, 10]. Hand counts are time consuming, and potentially error prone themselves. To address this, we designed a "semi-automated" process. The idea is to send the contents of each ballot through the serial line to a PC which will display the ballot *as interpreted by the* AV-OS. The poll worker performing the audit can then verify that the ballot was read correctly and correct errors. This will yield two benefits: First, the process of comparing the ballot sheets to the displayed counters should be faster for poll workers than the usual hand count process. Second, more generally this will allow the collection of data regarding the source of ballot read errors to improve accuracy in future elections.

In the modified firmware, an additional function call is added to the code that executes after each ballot is cast. This call sends all the counters, including the latest ballot, through the serial line. The program running on the connected PC tracks the counters as they are received and computes the difference in order to display the latest cast ballot. The poll worker using this interface can then make changes using the interface if the ballot scanner has made errors. The program on the PC keeps track of these changes in order to report on the reliability of the AV-OS and to compute the revised totals. Note that current audit procedures typically require multiple witnesses for the hand counts, and similar procedures must be used in this proposed auditing machine.

Needless to say a dual way of viewing this firmware modification is as an attack on voter privacy. Setting up a laptop in the receiving end of the serial port of the AV-OS terminal during election will retain the full record of all cast ballots in the exact order they were cast.

The design of the above auditing process suggests an improved approach to optical scan voting in which voters can verify the votes as recorded by the OS machine and reject the recorded ballot if there are errors. Voters may then have the option of fixing the ballot (e.g., filling in bubbles more completely) or placing the ballots in a separate box to be hand counted later. There are other combined electronic and paper systems, such as the Populex Digital Paper Ballot System (www.populex.com), that produce a paper ballot from voter input at a DRE type terminal. However, the proposed design differs in that the VVPAT is *voter generated*. This should dispel any doubts about the VVPAT itself. It also has the benefit of higher efficiency since many voters can fill out ballot sheets simultaneously at tables with privacy screens and share a single terminal for recording their ballots, as with ordinary OS machines. With DRE type machines, more terminals are required to achieve the same level of throughput, adding to the high costs of electronic voting.

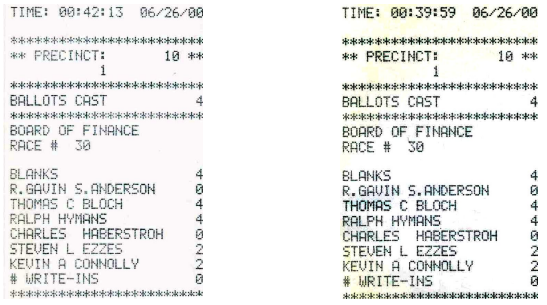
6. A MALICIOUS MANIPULATION

We test the possibility of implementing a malicious firmware that swaps the contents of two counters. The firmware is a trusted component and is not subject to auditing. We illustrate that a malicious attacker can reverse engineer the firmware and produce a malicious version which performs a counter-swap attack and cannot be traced by current pre-election auditing procedures.

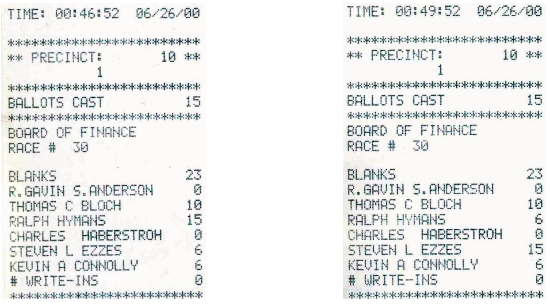
For the purpose of this attack, a malicious function was added to the firmware, in the free space that exists at the end of the EPROM. With the exception of this malicious function the firmware is identical with the AV-OS 1.96.6 firmware. The importance of this is that the "Malicious Firmware" behaves like the original AV-OS firmware version 1.96.6.

The injected function is invoked right after an election is closed and before the results of the election are printed. The function checks whether a threshold of "ballots cast"

has been reached. If enough ballots have been casted, which implies that we are not in a hand-counted test election, but in the actual election, it swaps the contents of two counters in the card. In our prototype the counters choice is hard coded, though in principle the selected counters could be determined arbitrarily. Note that like [5] the attack uses a "time bomb", since the firmware will perform the counter swap only during the election. Furthermore, the damage from the attack is permanent, since the contents of the counters are permanently altered in the card. Thus electronic reporting through GEMS and the printed election results will agree. To illustrate the results of this attack, Figure 2 shows the printed results with 4 and 15 votes, using the original and modified firmware. Note that the votes are swapped only when there has been sufficient votes cast to indicate a real election.



(a) Using original firmware (left) and modified (right). Note that the behavior is the same.



(b) Using original firmware (left) and modified (right). Note that two candidates have swapped votes.

Figure 2: Results printed after casting 4 ballots (a) and 15 ballots (b)

Finally the fact that the attack is packaged into the firmware makes it impossible to detect it through the verification of the bytecode found in the memory card; measures such as cryptographically signing the bytecode would be entirely ineffective against the type of attack presented in this section.

7. CONCLUSION

In this work we demonstrated a set of firmware manipulations for the AV-OS voting terminal that enable an attacker to violate voter privacy or permanently damage the reporting of the election results. Our implementations being at the firmware level are immune to any potential cryptographic integrity checks. A variation of other exploits may also be implemented in the firmware. One such exploit could be the alteration of the election results following a predefined voting pattern or button pressing sequence.

The firmware manipulation techniques we present raise some important questions: *How trusted should the hardware of an electronic voting terminal be and what means are required to improve trustworthiness of such systems?*

We stress that all the findings presented here were developed from first principles and at no time we had access to the vendor specifications of the system or the software source using inexpensive tools. Our results strongly underscore the importance of pre-election and post-election audits for any voting procedure, that should include an integrity check of the firmware code. Moreover, the incorporation of firmware cryptographic integrity checking in the architecture of an e-voting machine can further expedite the auditing process. While the subject of this case study was the AV-OS we have no reason to believe that our findings would have been any different had we focused on a different system.

8. REFERENCES

[1] Black Box Voting http://blackboxvoting.org.

[2] H. Hursti, Critical Security Issues with Diebold Optical Scan Design, Black Box Voting Project, July 4, 2005. www.blackboxvoting.org/BBVreport.pdf

[3] A. Kiayias, L. Michel, A. Russell, A. Shvartsman, M. Korman, A. See, N. Shashidhar and D. Walluck, Security Assessment of the Diebold Optical Scan Voting Terminal, voter.engr.uconn.edu/voter/Report-OS.html

[4] A. Kiayias, L. Michel, A. Russell, N. Shashidhar, A. See, and A. Shvartsman, An Authentication and Ballot Layout Attack Against an Optical Scan Voting Terminal. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 07), August, 2007, Boston, MA.

[5] A. Kiayias, L. Michel, A. Russel, N. Shashidhar, A. See, A. Shvartsman, S. Davtyan. Tampering with Special Purpose Trusted Computing Devices: A Case Study in Optical Scan E-Voting. Twenty-Third Annual Computer Security Applications Conference (ACSAC), December, 2007, Fl.

[6] D. Wagner, D. Jefferson and M. Bishop, Security Analysis of the Diebold AccuBasic Interpreter, Voting Systems Technology Assessment Advisory Board, University of California, Berkeley, February 14, 2006.

[7] V25+ and V35+ User's Manual, NEC Corporation, December, 1992.

[8] J. Calandrino, A. Feldman, J. Halderman, D. Wagner, H. Yu, W. Zeller, Source Code Review of the Diebold Voting System, July 20, 2007. www.sos.ca.gov/elections/elections_vsr.htm

[9] A. Kiayias, L. Michel, A. Russell, N. Shashidhar, A. See, A. Shvartsman, Pre-Election Testing and Post-Election Audit of Optical Scan Voting Terminal Memory Cards. USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 08), July 2008, San Jose, California.

[10] The Connecticut Citizen Election Audit Coalition, Report and Feedback February 2008 Connecticut Election Audit Observation, April 3, 2008 www.ctelectionaudit.org/Reports/ObservationReportFeb08.pdf

[11] S. Goggin and M. Byrne, An Examination of the Auditability of Voter Verified Paper Audit Trail (VVPAT) Ballots, 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 07), August, 2007, Boston, MA.