DIEBOLD
ELECTION SYSTEMS

July 30, 2003

Checks and balances in elections equipment and procedures prevent alleged fraud scenarios

Diebold Election Systems has concluded that the recent voting systems report conducted by Associate Professor Aviel Rubin alleges scenarios that could not occur within an actual election process due to the checks and balances within the actual equipment and those found in accepted election procedures.

On July 23, 2003, a team of students, directed by Rubin and Assistant Professor Dan Wallach, issued a report to the media alleging serious security flaws in Diebold Election Systems' AccuVote®-TS.

Following this executive summary is a detailed rebuttal of the flaws and inaccuracies of the Rubin report. The problems with the report fall into four major categories:

- Lack of presentation and/or understanding of the full electoral process
- False technical assumptions
- Inadequate research methodology
- Insufficient input from election experts

The Diebold AccuVote-TS touch screen voting station is just one element of the electoral process that is highly regimented and that has evolved and been honed over hundreds of years in the United States. When developing their various election fraud scenarios, the authors of the report focused solely on the part the AccuVote-TS software plays in the voting process, while ignoring the other critical checks and balances present in our electoral system. The authors' expressed assumptions include fundamental misunderstandings of the overall election process - whether making use of modern electronic voting systems or traditional papers ballots.

An election is a very public event. Every critical process throughout an election cycle is verified by third parties to ensure the integrity of the electoral system.
These checks and balances include:

- Equipment Certification – Voting equipment is certified by the National Association of State Election Directors and generally by the state certification expert, as well.

- Equipment Purchase – Through an independent committee or an appointed elected board, a county, state or municipality evaluates and chooses election equipment.

- Equipment Receipt - Contracts authorized by local election boards for the purchase of election equipment generally include mandatory acceptance testing, to confirm that all the voting equipment meets their standards.

- Ballot Preparation - Programmed ballots are reviewed by county officials to confirm that the correct ballot with the proper candidates will be presented to each voter.

- Equipment Preparation - Each piece of equipment is prepared for the election by election staff. Dependent on local statutes, a public test is held to verify this process. Before this process and after the public test is completed, all equipment is sealed and secured until delivery to the polling location.

- Election Day - Each voter is checked-in by election staff to ensure that they are a registered voter and that they are eligible to vote. The check-in judge records their intention to vote, and acts as the safeguard to ensure that the person only votes once. A ballot or machine judge activates the voter access card, which authorizes and allows one vote per voter. The machine judge monitors the voting process, and identifies and removes any voter who attempts to tamper with the voting equipment.

- Election Results - Election results are either electronically transmitted by the election judges or hand carried by the election judges to election headquarters. Electronically transmitted results are never deemed "official" election results.

- Election Canvass - After the election, a canvass is conducted to review accumulated votes. Results from individual voting units are uploaded into a new election and compared with the totals on election night.

In addition to expressing an incomplete understanding of the full scope of the electoral process, the report is explicitly based in large part on false assumptions. One such assumption the authors made is that the system is somehow connected to the Internet during the voting process. This is absolutely not the case, and had the authors consulted Diebold Election System programmers or technical support personnel, they could have easily corrected this inaccurate assumption.

The researchers reached their conclusions after reviewing an inadequate, incomplete sample of Diebold Election Systems' voting system software. Furthermore, the report was released directly to the press, rather than submitted to standard peer review by presentation to open academic forum.

The researchers also did not take advantage of the vast knowledge of election oversight organizations, such as the Federal Election Commission, The Election Center, or the Independent Testing Authorities. Doing so would have helped the authors to understand and present the rigorous standards voting systems are required to meet before they are put to use in real-world elections.

This report, which delivers a multitude of false conclusions based on such inadequate information, is damaging to the community of people working to improve the efficiency and security of the electoral process.

Diebold Election Systems welcomes a healthy debate on electronic voting. The following responses to many of the points raised in the report have been prepared in order to bring clarity and balance to this issue.

Detailed technical analysis of recent voting system report

1.1 Electronic Voting Systems

Allegation #1 (p. 2):

> *"The most fundamental problem with such a voting system is that the entire election hinges on the correctness, robustness, and security of the software within the voting terminal."*

Response:

This is fundamentally incorrect. The software is only one part of a voting process. The totality of the software, hardware and the electoral process and procedures, which include certification and testing by election officials, is what safeguards the integrity of election results. Cryptography, in particular, is only a small part of the equation.

Allegation #2 (p. 2):

> *"Should that code have security relevant flaws, they might be exploitable either by unscrupulous voters or by malevolent insiders. Such insiders include election officials, the developers of the voting system, and the developers of the embedded operating system on which the voting system runs."*

Response:

To be true, this claim would require a *conspiracy* of unscrupulous voters or malevolent insiders, or a combination of the two. The electoral process is designed in such a way that no single individual, or even a small group of individuals, can tamper with the election results. It is also important to note that such a conspiracy would not necessarily require any "security relevant flaws" in the code to accomplish its aims. Fraud of this degree would have the potential to undermine *any* voting system.

Allegation #3 (p. 2):

> *"Using this 'Mercuri method'* [Voter Verifiable Receipt], *the tally of the paper ballots takes precedence over any electronic tallies. As a result, the correctness of the voting terminal software no longer matters; either a voting terminal prints correct ballots or it is taken out of service."*

Response:

Voter verifiable receipts do have the advantages cited by the authors, though this solution essentially reduces an electronic system to a paper system, which has risks of its own. For example: Unscrupulous election officials can replace the paper ballot receipts with their own tampered copies ("ballot stuffing" in the classic sense). There are other pros and cons associated with such a system, but a full discussion of voter verifiable paper receipts is beyond the scope of this response.

1.2 "Certified" DRE systems

Allegation #4 (p. 3):

> *"Many government entities have adopted paperless DRE systems without appearing to have critically questioned the security claims made by the systems' vendors. Until recently, such systems have been dubiously 'certified' for use without any public release of the analyses behind these certifications, much less any release of the source code that might allow independent third parties to perform their own analyses."*

Response:

The certification process is both rigorous and arduous. It is by no means dubious. The certification and testing bodies *are* the third parties that perform their own analysis. Many states require a voting system to go through several levels of analysis before being accepted for production. The Voting System Standards are available to the public on the Federal Election Commission website.

Most of the previous systems that election officials seek to replace either have not been subject to third party testing or have not been tested to the same rigorous standards that are now required for new systems, including the current DRE systems.

The source code for ballot tabulation systems is generally required by statute or regulation to be placed in a third party escrow facility, to be examined only upon court order or the vendor's failure to support the code.

1.3 Summary of Results

Allegation #5 (p. 4):

> *"Furthermore, the protocols used when the voting terminals communicate with their home base, both to fetch election configuration information and to report final election results, do not use cryptographic techniques to authenticate the remote end of the connection nor do they check the integrity of the data in transit."*

Response:

The Ballot Station software doesn't receive information from the central site from the polling place. This activity is done over private, disconnected networks at election central under the auspices of election officials. At the polls, the system operates offline until polls close, and is then only optionally connected to upload *unofficial* election results to the central server.

Allegation #6 (p. 4):

> *"Given that these voting terminals could communicate over insecure phone lines or even wireless Internet connections, even unsophisticated attackers can perform untraceable 'man-in-the-middle' attacks."*

Response:

Uploading unofficial election results is done over a private point-to-point network and not through the Internet or dial-up Internet services.

Allegation #7 (p. 4):

"As part of our analysis, we considered both the specific ways that the code uses cryptographic techniques and the general software engineering quality of its construction. Neither provides us with any confidence of the system's correctness."

Response:

The authors' analysis provides no evidence whatsoever of the system's *incorrectness,* and was undertaken without full knowledge of the election systems.

Allegation #8 (p. 4):

"Cryptography, when used at all, is used incorrectly."

Response:

This statement is based on the presumption that there is a single correct means of using cryptography. This is not accurate. The software is designed with the realization that subsequent versions will be released to address any needed improvements or requested changes; but the cryptography in the software is used as the developers intended, taking into account additional security measures, and the possibility of future development.

Allegation #9 (p. 4):

"In many places where cryptography would seem obvious and necessary, none is used. More generally, we see no evidence of rigorous software engineering discipline. Comments in the code and the revision change logs indicate the engineers were aware of areas in the system that needed improvement, though these comments only address specific problems with the code and not with the design itself."

Response:

First, the software examined in the report was an older version. Second, *all* software has areas that can be improved. It is a sign of good engineering practice to see that those areas are identified in the code. Third, the software has passed rigorous functional tests and reviews. Fourth, none of the areas that are identified in the code as targets for future improvement have any affect on the accuracy of results or any other critical component. Lastly, because the authors didn't share their results with us prior to publication, they had no understanding of our engineering discipline.

Allegation #10 (p. 4):

"We also saw no evidence of any change control process that might restrict a developer's ability to insert arbitrary patches to the code. Absent such processes, a malevolent developer could easily make changes to the code that would create vulnerabilities to be later exploited on Election Day."

Response:

It would be realistically impossible for a malevolent software developer within the company to successfully insert suspect code. Diebold Election Systems has multiple procedural checks against just such a possibility. Making such changes would require a conspiracy among nearly all the developers in the company, in addition to members of the quality assurance group. Even granting the strictly theoretical possibility of such a conspiracy, which would be a criminal act, any malicious code would not survive the code review process conducted by the independent testing authority. Even if missed by the ITA, logic and accuracy tests conducted by the local and state entities would almost certainly expose any malevolent behavior.

Furthermore, once malevolent code was inserted, it would be discoverable in perpetuity. It simply would not be possible to install software on tens of thousands of units undetected. A checksum of the FEC-certified binary is made, and this checksum is available for comparison with installed units.

Suggesting the *possibility* that a developer *could* insert code, without any evidence of this ever happening, is pure speculation.

Allegation #11 (p. 4):

"We also note that the software is written entirely in C++. When programming in an unsafe language like C++, programmers must exercise tight discipline to prevent their programs from being vulnerable to buffer overflow attacks and other weaknesses."

Response:

The assertion that C++ is "unsafe" is an oversimplification. Programming in any language can be safe or unsafe, depending on how it is used. C++ is an object-oriented language that is totally appropriate for this application.

Unlike a Web server or other Internet enabled applications, the code is not vulnerable to most "buffer overflow attacks" to which the authors refer. This form of attack is almost entirely inapplicable to our application. In the limited number of cases in which it would apply, we have taken the steps necessary to ensure correctness.

We could find no evidence of a buffer overflow in the cited reference, and the report itself indicates that the researchers found evidence that the developers properly addressed buffer overflow concerns.

2 System overview

Allegation #12 (p. 4):

"Although the Diebold code is designed to run on a DRE, one can run it on a regular Microsoft Windows computer (during our experiments we compiled and ran the code on a Windows 2000 PC)."

Response:

The software will run on a personal computer, but this is not a supported configuration and is not used in Diebold's election systems during elections.

Allegation #13 (p. 4):

"In the following we describe the process for setting up and running an election using the Diebold system. Although we know exactly how the code works from our analysis, we must still make some assumptions about the external processes at election sites. In all such cases, our assumptions are based on the way the Diebold code works, and we believe that our assumptions are reasonable."

Response:

In many cases these assumptions are *not* reasonable. The election process is highly formalized, and the authors clearly are unaware of many of the safeguards these processes put in place. Any election official the authors had consulted would have had sufficient knowledge of these safeguards to point out the obvious complementary nature of the software and procedural safeguards.

Allegation #14 (p. 7):

"Shortly prior to the election, the voting terminals must be installed at each voting location. In common usage, we believe the voting terminals will be distributed without a ballot definition pre-installed."

Response:

This is a false assumption. The ballot definition is pre-installed and distributed with the AccuVote-TS unit. The memory card is sealed in the unit behind a tamper-proof label.

Allegation #15 (p. 7):

"They may be distributed using removable media, such as floppy disks or storage cards."

Response:

The election data is stored on memory cards only (not floppy disks), which are locked inside the physical Ballot Station and continuously controlled by local election officials.

Allegation #16 (p. 7):

"They may also be transmitted over the Internet or a dial-up connection."

Response:

Although theoretically possible, in practice the election data is not transferred over the Internet or a dial-up connection. The election data (election.edb) is transferred to memory cards over disconnected local area networks at election central.

Allegation #17 (p. 7):

> *"We do not know exactly how the voter gets his voter card. It could be sent in the mail with information about where to vote, or it could be given out at the voting site on the day of the election."*

Response:

> The voter cards are created by election officials and given to the voter at the polls – identical to what has occurred in locations that have previously used Diebold Election Systems' voting systems. Voter cards are never mailed to voters. The voter cards are returned to the election officials prior to the voter leaving the polling site.

Allegation #18 (p. 7):

> *"We are unable to verify that there are checks to ensure, for example, that there are no more votes collected than people who are registered at or have entered any given polling location."*

Response:

> Various checks are performed using reports available from the software and as a matter of electoral process. This has been validated previously by tests and during actual elections.

3.1 Homebrew smartcards

Allegation #19 (p. 6):

> *"Upon reviewing the Diebold code, we observed that the smartcards do not perform any cryptographic operations. For example, authentication of the terminal to the smartcard is done 'the old-fashioned way': the terminal sends a cleartext (i.e., unencrypted) 8-byte password to the card and, if the password is correct, the card believes that it is talking to a legitimate voting terminal. Unfortunately, this method of authentication is insecure: an attacker can easily learn the 8-byte password used to authenticate the terminal to the card (see Section 3.3), and thereby communicate with a legitimate smartcard using his own smartcard reader."*

Response:

> Hypothetically, it would be possible to reverse engineer the password using the means described. But to describe the process as "easy" is an exaggeration.

Allegation #20 (p. 9):

> *"First, we note that many smartcard vendors sell cards that can be programmed by the user."*

Response:

> Diebold Election Systems orders smartcards under contract with a single vendor. The cards are not generic cards, but are created specifically for Diebold Election Systems, with a distinct configuration. While an attacker could purchase smartcards from the same vendor, such an attacker would also need to know and

request the specific cards in a specific configuration, without raising the suspicions of the vendor.

Allegation #21 (p. 9):

"Again, given the privacy of voting booths, an attacker using such a card reader would be unlikely to be noticed. Given the ease with which an attacker can interact with legitimate smartcards, plus the weak password-based authentication scheme (see Section 3.3), an attacker could quickly gain enough insight to create homebrew voting cards, perhaps quickly enough to be able to use such homebrew cards during the same election day."

Response:

This is possible in theory, but it is an entirely ineffective way to alter the outcome of an election.

Since the perpetrators would need to be at the polls, and sign in, this would seem to be a high personal risk for the few fraudulent votes that could be cast.

Even if a voter manages to record a handful (at most) of fraudulent votes, this would be discovered during the process of *roster reconciliation*. Each voter, in order to vote, must sign a roster when they enter the polls. Post election, before results are made official, the signatures are reconciled with the number of ballots cast on the voting machines. If the totals do not reconcile, an investigation is launched.

Note also that the more fraudulent votes cast, the more glaring the roster reconciliation error. It is virtually impossible to cast enough votes to sway an election in this manner.

3.1.1 Multiple voting

Allegation #22 (p. 10):

"More simply, instead of bringing multiple cards to the voting booth, the adversary could program a smartcard to ignore the voting terminal's deactivation command. Such an adversary could use one card to vote multiple times."

Response:

This is incorrect. We are not aware of a way to program the smartcard in such a way that the card will not be canceled. The authors provide no explanation as to how they believe this could be accomplished.

Allegation #23 (p. 10):

"Will the adversary's multiple-votes be detected by the voting system? To answer this question, we must first consider what information is encoded on the voter cards on a per-voter basis. The only per-voter information is a 'voter serial number' (`m_VoterSN` in the `CVoterInfo` class). Because of the way the Diebold system works, `m_VoterSN` is only recorded by the voting terminal if the voter decides not to place a vote (as noted in the comments in `TSElection/Results.cpp`, this field is recorded for uncounted votes for

backward compatibility reasons). It is important to note that if a voter decides to cancel his or her vote, the voter will have the opportunity to vote again using that same card (and, after the vote has been cast, `m_VoterSN` *will not be recorded).“*

Response:

The authors were viewing code in development and as such misinterpreted the purposes. Once a card is voted, the ballot information is in fact stored when the ballot is cast, not canceled.

As a secondary issue, the authors also misunderstand the usage of `m_VoterSN`. This field is used only for “provisional” or so called “challenged” voters, whose identity needs to be verified after the election. We do not store per-ballot voter identifiers for reasons of voter anonymity. This is an FEC requirement to ensure voter privacy.

Allegation #24 (p. 10):

“Can the back-end tabulation system detect multiple-vote casting?“

Response:

No, for the voter anonymity reasons cited above. Roster reconciliation does, however, detect this.

Allegation #25 (p. 10):

“If we assume the number of collected votes becomes greater than the number of people who showed up to vote, and if the polling locations keep accurate counts of the number of people who show up to vote, then the back-end system, if designed properly, should be able to detect the existence of counterfeit votes. However, because `m_VoterSN` *is only stored for those who did not vote, there will be no way for the tabulating system to count the true number of voters or distinguish the real votes from the counterfeit votes.”*

Response:

Ultimately, *no* system can do that practically, without sacrificing voter anonymity, regardless of the use of `m_VoterSN`. It is possible to use various techniques to obscure the codes used to identify a voter and a ballot, but at the end of the day, an insider can always then match the voter to the vote, violating FEC rules. A system, through roster reconciliation, can detect that there were more votes cast than allowed.

3.1.2 Administrator and ender card

Allegation #26 (p. 10):

“Using a homebrew administrator card, a poll worker, who might not otherwise have access to the administrator functions of the Diebold system but who does have access to the voting machines before and after the elections, could gain access to the administrator controls. If a malicious voter entered an administrator or ender card into the voting device instead of the normal voter card, then the voter would be able to terminate the election and, if the card is an administrator

card, gain access to additional administrative controls. The use of administrator or ender cards prior to the completion of the actual election represents an interesting denial-of-service attack. Once 'ended,' the voting terminal will no longer accept new voters (see `CVoteDlg::OnCardIn()`) until the terminal is somehow reset. Such an attack, if mounted simultaneously by multiple people, could shut down a polling place."

Response:

While possible in theory, such an attack would be detected instantly. The voter might have left the polling place, but he or she would have signed the roster to gain access to the voting machine. An investigation of all recent voters would be launched, making this a high risk attack.

Allegation #27 (p. 11):

"Even if the poll workers were later able to resurrect the systems, the attack might succeed in deterring a large number of potential voters from voting (e.g., if the attack was performed over the lunch hour)."

Response:

Administratively "closing" or even "deleting" an election, as the report authors theorize, is an entirely reversible operation. Because the votes can always be recovered, this attack would be entirely ineffective. There are multiple Ballot Stations in the polling place, and if one or more stations had to be reset, it would not significantly hamper the ability of others to vote. It is not a lengthy or complicated process.

3.3 Terminal-to-card authentication

Allegation #28 (p. 12):

"First, hard-coding passwords in C++ files is generally a poor design choice."

Response:

This issue has since been resolved in subsequent versions of the software.

Allegation #29 (p. 12):

"Recompiling on a per-election basis may also be a concern, since good software engineering practices would dictate additional testing and certification if the code were to be recompiled for each election."

Response:

This argument is based on a false assumption. The system does not recompile on a per-jurisdiction basis. Beyond being bad practice, this would violate FEC rules.

Allegation # 30 (p. 12):

"If the password chosen by the designers of the system (...) does not work, then `CCLXSmartCard:: Open()` uses the smartcard manufacturer's default password. One issue with this is that it implies that sometimes the system is used with un-initialized smartcards."

Response:

It is of academic interest to the arguments raised here, but the card is immediately cleared after authenticating against the secondary password here (called the manufacturer's default password). A cleared card is no threat to election integrity.

Allegation #31 (p. 12):

"This means that an attacker might not even need to figure out the system's password in order to be able to authenticate to the cards. As we noted in Section 3.1, some smartcards allow a user to get a listing of all the files on a card."

Response:

Diebold Election Systems' smartcards do not permit this.

Allegation #32 (p. 12):

"If the system uses such a card and also uses the manufacturer's default password of . . . , then an attacker, even without any knowledge of the source code and without the ability to intercept the connection between a legitimate card and a voting terminal, but with access to a legitimate voter card, will still be able to learn enough about the smartcards to be able to create counterfeit voter cards."

Response:

This is not true. With the factory password alone, it is impossible to determine anything about the file structure. It is also impossible to determine the encoding of the voter card data.

Allegation #33 (p. 12, footnote):

"Many smartcards are shipped with default passwords. Making homebrew cards this way is somewhat risky, as the attacker must make assumptions about the system. In particular, the attacker is assuming that his or her counterfeit cards would not be detected by the voting terminals. Without access to the source code, the attacker would never know, without testing, whether it was truly safe to attack a voting terminal."

Response:

This is correct. And furthermore, an attacker has no way of knowing, even with the source code in hand, whether it matches that being used in the election.

4 Data storage

Allegation #34 (p. 13):

"While the data stored internally on each voting terminal is not as accessible to an attacker as the voting system's smartcards, exploiting such information presents a powerful attack vector, especially for an election insider."

Response:

It is an attack vector *only* for an election insider at election central, or an individual with the active cooperation of an election insider at election central.

Voters have no access to the storage cards at all. Poll workers have access to the card *after polls close*, but (like a classic paper ballot box) only under the observation of their peer poll workers.

4.1 Data storage overview

Allegation #35 (p. 13):

> *"Unfortunately, under Windows CE, which we believe is used in commercial Diebold voting terminals, the existence of the removable storage device is not enforced properly. Unlike other versions of Windows, removable storage cards are mounted as subdirectories under CE. When the voting software wants to know if a storage card is inserted, it simply checks to see if the* `Storage Card` *subdirectory exists in the file system's root directory. While this is the default name for a mounted storage device, it is also a perfectly legitimate directory name for a directory in the main storage area. Thus, if such a directory exists, the terminal can be fooled into using the same storage device for all of the data. This would reduce the amount of redundancy in the voting system and would increase the chances that a hardware fault could cause recorded votes to be lost."*

Response:

This line of argument is not logical. Election results would still be stored in two locations, albeit on the same media. More questionable is the impracticality of setting up an election like this. Data is downloaded on to the storage (PC) cards using a few units, and then the stacks of cards are inserted into the thousands of terminals to be sent to the polling places. To load on to an internal directory, you would have to download to each of these units directly – a logistically impossible task. Such an undertaking, furthermore, even if it were not impossible, would almost certainly be discovered by auditors checking the election setup prior to the election.

4.2 System configuration

Allegation #36 (p. 13):

> *"Thus, all an adversary must do is modify the system registry to trick a given voting terminal into effectively impersonating any other voting terminal."*

Response:

There is nothing to impersonate here. The "identity" of the voting terminal is determined by the contents of the storage card, not the serial number of the machine. The serial number on the machine is used for informational purposes only. The machines are interchangeable.

Allegation #37 (p. 13):

> *"It is unclear how the tallying authority would deal with results from two different voting terminals with the same voting ID – at the very least human intervention to resolve the conflict would probably be required."*

Response:

Again, it is the storage card that stores the ID (actually multiple IDs). Duplicate IDs are managed through version fields (which the authors mention in other parts of their paper) and through other means such as audit records on both the storage media, internally to the unit, and on the central election server.

Allegation #38 (p. 14):

"By modifying this counter, an adversary could cast doubt on an election by creating a discrepancy between the number of votes cast on a given terminal and the number of votes that are tallied in the election."

Response:

Again, the authors misunderstand the purpose of the counter. The referenced statistic is called the "public counter" (in contrast to the "protective counter"). The public counter is stored on the storage card.

Even if the purpose of the counter was to store the number of ballots cast in the election, there is no way for a voter or poll worker to modify this counter, even with an administrator card.

Allegation #39 (p. 14):

"While the current method of implementing the counter is totally insecure, even a cryptographic checksum would not be enough to protect the counter; an adversary with the ability to modify and view the counter would still be able to roll it back to a previous state. In fact, the only solution that would work would be to implement the protective counter in tamper-resistant hardware token, requiring modifications to the physical voting terminal hardware."

Response:

A write-once hardware device would be gross overkill for the intended purpose of the protective counter. In fact, we are not aware of a hardware device that could serve this purpose, even if one were warranted. Even so-called digital rights management chips are ultimately read-write, and hence modifiable.

4.3 Ballot definition

Allegation #40 (p. 14):

"The 'ballot definition' for each election contains everything from the background color of the screen to the PPP username and password to use when reporting the results. This data is not encrypted or checksummed (cryptographically or otherwise) and so can be easily modified by any attacker with physical access to the file."

Response:

The attacker would need physical access to the file. Preventing this is a matter of process control at election central during the programming of the election.

Allegation #41 (p. 14):

> *"While many attacks, such as changing the party affiliation of a candidate would be noticed by some voters, many more subtle attacks are possible. By simply changing the order of the candidates as they appear in the ballot definition, the results file will change accordingly. However, the candidate information itself is not stored in the results file. The file merely tracks that candidate 1 got so many votes and candidate 2 got so many other votes. If an attacker reordered the candidates on the ballot definition, voters would unwittingly cast their ballots for the wrong candidate. As with denial-of-service attacks (see Section 3.1.2), ballot reordering attacks would be particularly effective in polling locations known to be heavily partisan."*

Response:

> This is incorrect. Changing the order of the candidates in the ballot definition would result in a change in the order of the results as well. Candidates would simply be listed in the wrong order on the ballot. It does not matter that the candidates are not stored in the results file. In reality, it is the very fact that the candidate keys *are not* stored with the results that makes the system immune to such tampering.

Allegation #42 (p. 14):

> *"Even without modifying the ballot definition, an attacker can gain almost enough information to impersonate the voting terminal to the back-end server. The terminal's voting center ID, PPP dial-in number, username, password and the IP address of the back-end server are all available in the clear (these are parsed into a* `CElectionHeaderItem` *in* `TSElection\TSElectionObj.cpp`*). Assuming an attacker is able to guess or create a voting terminal ID, he would be able to transmit fraudulent vote reports to the backend server by dialing in from his own computer. While both the paper trail and data stored on legitimate terminals could be used to compensate for this attack after the fact, it could, at the very least, delay the election results."*

Response:

> Such an attempt would result in an "already uploaded" message. If poll workers receive an "already uploaded" message, they are instructed to contact election central to immediately check the results for that precinct. If the results are suspicious (for example, have wildly weighted results), that precinct's totals would be quarantined until the issue is resolved.

4.4 Votes and audit logs

Allegation #43 (p. 15):

> *"Unlike the other data stored on the voting terminal, both the vote records and the audit logs are encrypted and checksummed before being written to the storage device. Unfortunately, neither the encrypting nor the checksumming is done securely. All of the data on a storage device is encrypted using a single, hardcoded DES [NBS77] key:* `#define DESKEY ((des_key*)"F2654hD4")` *Note that this value is not a hex representation of a key. Instead, the bytes in the string '*`F2654hD4`*' are fed directly into the*

DES key scheduler. If the same binary is used on every voting terminal, an attacker with access to the source code, or even to a single binary image, could learn the key, and thus read and modify voting and auditing records."

Response:

An attacker would need access to *both* the source code *and* the physical storage.

Allegation #44 (p. 15):

"Even if proper key management were to be implemented, many problems would still remain. First, DES keys can be recovered by brute force in a very short time period [Gil98]. DES should be replaced with either triple-DES [Sch96] or, preferably, AES [DJ02]."

Response:

There are stronger forms of compression than DES, but the authors' implication that the keys can be recovered "in a short time" is deliberately misleading. To mount such an attack, poll workers would have to collectively bring to bear the resources outlined in [Gil98] at the polling place before printing out the results tape – an incredibly unrealistic task.

Allegation #45 (p. 15):

"Jones reports that the vendor may have been aware of this design flaw in their code for several years [Jon01, Jon03]. We see no evidence that this design flaw was ever addressed."

Response:

We were not able to find such a claim in the Jones paper.

Allegation #46 (p. 15):

"At the time that the logging occurs, the log can also be printed to an attached printer. If the printer is unplugged, off, or malfunctioning, however, no record will be stored elsewhere to indicate that the failure occurred."

Response:

The poll workers would take corrective action if the printer was malfunctioning.

Allegation #47 (p. 15):

"The following code from `TSElection/Audit.cpp` *demonstrates that the designers failed to consider these issues:*

```
if (m_Print && print) {
CPrinter printer;
// If failed to open printer then just return."
```

Response:

This is incorrect. This comment shows that the designers acknowledged the behavior.

Allegation #48 (p. 16):

> *"If the cable attaching the printer to the terminal is exposed, an attacker could create discrepancies between the printed log and the log stored on the terminal by unplugging the printer (or, by simply cutting the cable)."*

Response:

> The log would still be stored on the smart card and on the internal storage. The printer cable to the terminal is not exposed.

Allegation #49 (p. 16):

> *"An attacker's most likely target will be the voting records themselves. Each voter's votes are stored as a bit array based on the ordering in the ballot definition file along with other information such as the precinct the voter was in, although no information that can be linked to a voter's identity is included."*

Response:

> The analysis fails to explain how such an attack would take place.

Allegation #50 (p. 16):

> *"If the voter has chosen a write-in candidate, this information is also included as an ASCII string. An attacker given access to this file would be able to generate as many fake votes as he or she pleased, and such votes would be indistinguishable from the true votes cast on the terminal."*

Response:

> This claim is based on a false assumption, as access to this file is not given to anyone, and this file is generally not accessible.

Allegation #51 (p. 16):

> *"While the voter's identity is not stored with the votes, each vote is given a serial number. These serial numbers are generated by a linear congruential random number generator (LCG), seeded with static information about the election and voting terminal. No dynamic information, such as the current time, is used. While the code's authors apparently decided to use an LCG because it appeared in Applied Cryptography [Sch96], LCG's are far from secure."*

Response:

> The report is misleading in this section. There is no need for "security" here. The only intent of this code is to pseudo-randomize the order of the ballots for purposes of display and reporting, as is required in some states.

Allegation #52 (p. 16):

> *"However, attacking this random number generator is unnecessary for determining the order in which votes were cast: each vote is written to the file sequentially. Thus, if an attacker is able to determine the order in which voters cast their ballots, the results file has a nice list, in the order in which voters used the terminal. A malevolent poll worker, for example, could surreptitiously track the order in which voters use the voting terminals. Later, in collaboration with*

other attackers who might intercept the poorly encrypted voting records, the exact voting record of each voter could be reconstructed."

Response:

To succeed in this type of attack, the malicious attacker would have to (1) keep track of all machines all day (never miscounting one); (2) form a conspiracy to gain illicit access to the storage card; and (3) launch a cryptographic attack. The reward for this effort would be to learn how someone voted. This is unrealistic.

Allegation #53 (p. 16):

"Physical access to the voting results may not even be necessary to acquire the voting records, if they are transmitted across the Internet."

Response:

This argument is based on a false assumption. Results are not transmitted over the Internet.

5 Communicating with the outside world

Allegation #54 (p. 16):

"The Diebold voting machines cannot work in isolation. They must be able to both receive a ballot definition file as input and report voting results as output."

Response:

This is false. Diebold Election Systems' voting terminals operate in isolation.

Non-operational communication to and from the Diebold Election Systems' voting terminals is minimal and tightly controlled. The election database is downloaded to the storage card at the county election offices using either a "gang" programming device or one of several Ballot Stations dedicated to the task. The storage card is then inserted into and sealed in a Ballot Station to be sent to the polls performed on each unit's printer.

The primary form of output for the Ballot Station is the result tape. Again, this is an entirely isolated operation performed on each unit's printer.

Allegation #55 (p. 16):

"As described in Section 2, there are essentially two ways to load a voting terminal with an initial election configuration: via some removable media, such as a floppy disk, or over the Internet. In the latter case, the voting terminal could either be plugged directly into the Internet or could use a dial-up connection (the dial-up connection could be to a local ISP, or directly to the election authority's modem banks)."

Response:

Ballot Station voting terminals are *not* loaded over the Internet. This would not be practical, as the databases are large and high-speed Internet access is not available in a typical polling place.

After the election is over, election results are sent via a private, point-to-point network to a back-end post-processing server.

Allegation #56 (p. 16):

"Unfortunately, there are a number of attacks against this system that exploit the system's reliance on and communication with the outside world."

Response:

There is no reliance on communication with the outside world, and when necessary for initialization or reporting, outside communication is tightly controlled, as previously described.

5.1 Tampering with ballot definitions

Allegation #57 (p. 17):

"We first note that it is possible for an adversary to tamper with the voting terminals' ballot definition file (`election.edb`*). If the voting terminals load the ballot definition from a floppy or removable storage card, then an adversary, such as a poll worker, could tamper with the contents of the floppy before inserting it into the voting terminal."*

Response:

This argument is based on a false assumption. Ballot Stations arrive at the polls with the storage card locked in the Ballot Station, behind a numbered tamper-resistant seal.

Allegation #58 (p. 17):

"On a potentially much larger scale, if the voting terminals download the ballot definition from the Internet, then an adversary could tamper with the ballot definition file en-route from the back-end server to the voting terminal. With respect to the latter, we point out that the adversary need not be an election insider; the adversary could, for example, be someone working at the local ISP. If a wireless network is used, anybody within radio range becomes a potential adversary. With high-gain antennas, the adversary can be sufficiently distant to have little risk of detection. If the adversary knows the structure of the ballot definition, then the adversary can intercept and modify the ballot definition while it is being transmitted. Even if the adversary does not know the precise structure of the ballot definition, many of the fields inside are easy to identify and change, including the candidates' names, which appear as plain ASCII text."

Response:

All of these imagined attacks are inapplicable to our election system, as it does not utilize the Internet.

5.2 Preventing the start of an election

Allegation #59 (p. 17):

> *"Suppose that the election officials are planning to download the configuration files over the Internet and that they are running late and do not have much time before the election starts to distribute ballot definitions manually (i.e., they might not have enough time to distribute physical media with the ballot definition files from central office to every voting precinct)."*

Response:

> This argument is based on a false assumption. Election definitions are not downloaded over the Internet. Generally, the election calendar, from before candidate filing through to Election Day, is controlled by a statutorily mandated sequence of milestones. This structured progression makes virtually impossible the proposed scenario in which election officials "do not have time" to physically distribute ballot definitions.

5.3 Tampering with election results

Allegation #60 (p. 18):

> *"Just as it is possible for an adversary to tamper with the downloading of the ballot definition file (Section 5.1), it is also possible for an adversary to tamper with the uploading of the election results."*

Response:

> As we have discussed, it is not possible to tamper with the download of the election definition without a conspiracy of officials at election central. It is theoretically possible, though unlikely, for an adversary to tamper with unofficial election results uploaded after polls close. In any case, such an attack would be both detected and resolved without the loss of election result integrity.

Allegation #61 (p. 18):

> *"To make this task even easier for the adversary, we note that although the election results are stored 'encrypted' on the voting devices (Section 4.4), the results are sent from the voting devices to the back-end server over an unauthenticated and unencrypted channel. In particular,* `CTransferResultsDlg::OnTransfer()` *writes ballot results to an instance of* `CDL2Archive`*, which then writes the votes in cleartext to a socket without any cryptographic checksum. Sending election results in this way over the Internet is a bad idea."*

Response:

> This argument is based on a false assumption. The system does not upload results over the Internet.

Allegation #62 (p. 18):

> *"Nothing prevents an attacker with access to the network traffic, such as workers at a local ISP, from modifying the data in transit. Such an attacker could, for example, decrease one candidates vote count by n and increase the another*

candidate's count by n. Of course, to introduce controlled changes to the votes, the attacker would require some knowledge of the structure of the messages sent from the voting terminals to the back-end server. If the voting terminals use a modem connection directly to the tabulating authority's network, rather than the Internet, then the risk of such an attack is less, although still not inconsequential. A sophisticated adversary (or employee of the local phone company) could tap the phone line and intercept the communication."

Response:

No ISP is involved in the uploading of unofficial results. The unofficial results are sent over a private point-to-point connection between the polling place (or another location) and election central. It would be difficult for an attacker to intercept and modify an analog modem connection, even by contravening federal wiretapping laws. Even in the event that such an attack could occur, the attacker would at best modify *unofficial* election results. See responses to section 4.3, above.

Allegation #63 (p. 18):

"All of these adversaries could be easily defeated by properly using standard encryption suites like SSL/TLS, used throughout the World Wide Web for e-commerce security. We are puzzled why such a widely accepted and studied technology is not used by the voting terminals to safely communicate across potentially hostile networks."

Response:

This argument is based on a false assumption. We do not transmit results over publicly accessible networks.

5.4 Attacking the voting terminals directly

Allegation #64 (p. 18):

"In some configurations, where the voting terminals are directly connected to the Internet, it may be possible for an adversary to attack them directly, perhaps using an operating system exploit or buffer overflow attack of some kind. Ideally the voting devices and their associated firewalls would be configured to accept no incoming connections [CBR03]. This concern would apply to any voting terminal, from any vendor, with a direct Internet connection."

Response:

Since Diebold Election Systems' voting machines are not attached to the Internet, this is a false assumption.

6 Software engineering

Allegation #65 (p. 19):

"Of course, the opposite is also true, perhaps even more so: it is very difficult to produce a secure system by building on an insecure foundation."

Response:

All the hypothetical insecurities identified in this paper do not affect the core operation of the software. The authors imply as much at the end of section 5.3.

Allegation #66 (p. 19):

"Of course, reading the source code to a product gives only an incomplete view into the actions and intentions of the developers who created that code."

Response:

The authors did not, to our knowledge, attempt to contact the developers as part of their research. As such, their view of our software engineering practices is speculation.

6.1 Code Legacy

Allegation #67 (p. 19):

"Indeed, Diebold acquired Global Election Systems in September, 2001. Some of the code, such as the functions to compute CRCs and DES, dates back to 1996, when Global Election Systems was called 'I-Mark Systems.' "

Response:

Global Election Systems bought IMARK and inherited this code. Global Election Systems was formed in the early 1990s, and never operated under the name IMARK.

Allegation #68 (p. 19):

"This legacy is apparent in the code itself as there are portions of the `AVTSCE` *code, including entire classes, that are either simply not used or removed through the use of* `#ifdef` *statements. Many of these functions are either incomplete or, worse, do not perform the function that they imply as is the case with* `CompareFiles` in `Utilities/FileUtil.cpp`:

```
BOOL CompareFiles(const CString& file1, const CString&
file2)
{
/* XXX use a CRC or something similar */
```

Currently the code will declare any two files to be the same that have the same size. The author's comment to use a CRC doesn't make much sense, as a byte-by-byte comparison would be more efficient. If this code were ever used, its inaccuracies could lead to wide variety of subsequent errors."

Response:

The authors reviewed an incomplete snapshot of source code in the process of development. Furthermore, it is common engineering practice to "comment out" code that is incomplete and/or not in use, but which may become useful at a later date.

Allegation #69 (p. 19):

"While most of the preprocessor directives that remove code correctly use `#if 0` *as their condition, some use* `#ifdef XXX`*. There is no reason that a later*

programmer should realize that defining `XXX` *will cause blocks of code to be reincluded in the system (causing unpredictable results, at best).“*

Response:

`XXX` is used to identify portions of code that require further attention before release. This is part of common software engineering practice. This symbol is never defined, and no such “unpredictable results” are factual.

Allegation #70 (p. 19):

“We also noticed 13http://dallas.bizjournals.com/dallas/stories/2001/09/10/daily2.html 19 `#ifdef LOUISIANA` *in the code. Prudent software engineering would recommend a single implementation of the voting software, where individual states or municipalities could have their desired custom features expressed in configuration files.”*

Response:

The software for a given official release is the same for all jurisdictions. This is an FEC requirement. The `#ifdef LOUISIANA` was used to comment out code that was used in a sales demonstration. The software was never brought up to “production” standards, so the code in question was commented out. The code could, however, form the basis for a complete implementation should market requirements dictate. For this reason it was not deleted. Again, this is common software development practice.

6.2 Coding style

Allegation #71 (p. 20):

“While the system is implemented in an unsafe language (C++), the code reflects an awareness of avoiding such common hazards as buffer overflows.“

Response:

Any programming language may be safe or unsafe, depending on how it is used. Again, C++ is *not* an inherently unsafe language.

Allegation #72 (p. 20):

“Of course, a better solution would have been to write the entire system in a safe language, such as Java or C#.“

Response:

We respectfully disagree with this observation on what is an appropriate language for an electronic voting terminal application. We believe many qualified software engineers would draw the same conclusion as we have.

Allegation #73 (p. 20):

“The core concepts of object oriented programming such as encapsulation are well represented, though in some places C++’s non-typesafe nature is exploited with casts that could conceivably fail.“

Response:

The analysis does not, however, offer any evidence of such an exploit or failure.

Allegation #74 (p. 20):

"This could cause problems in the future as these locations are not well documented. Overall, the code is rather unevenly commented. While most files have a description of their overall function, the meanings of individual functions, their arguments, and the algorithms within are more often than not undocumented. An extreme example of a complex but completely undocumented function is the

`CBallotRelSet::Open` function from `TSElection/TSElectionSet.cpp`:

```
void CBallotRelSet::Open(const CDistrict* district,
const CBaseunit* baseunit,
const CVGroup* vgroup1, const CVGroup* vgroup2)
{
```

Nothing about this code makes its purpose readily apparent. Certainly, it has two nested loops and is doing all kinds of comparisons. Beyond that, most readers of the code would need to invest significant time to learn the meaning of the various names shown here."

Response:

Most external readers of the code are not qualified to make this assessment. The code cited is by no means trivial. The current software includes a comment in the code cited. However, to a developer who is familiar to the data types in question (they are all common types in our application), the function of this example is actually quite obvious.

As a side note, the new 2002 FEC voting standards require a comment header on all functions and a comment on all variables at the point of declaration. The code has been modified to meet these guidelines.

6.3 Coding process

Allegation #75 (p. 21):

"An important point to consider is how code is added to the system. From the CVS logs, we can see that most code updates are in response to specific bugs that needed to be fixed. There are numerous authors who have committed changes to the CVS tree, and the only evidence that we have found that the code undergoes any sort of review process comes from a single log comment:'"Modify code to avoid multiple exit points to meet Wyle requirements.' This could refer to Wyle Laboratories whose website claims that they provide all manner of testing services."

Response:

Wyle is the independent testing authority responsible for adjudicating conformance with FEC voting system standards. Election vendors must submit to rigorous third party testing to achieve certification. Indeed, only a handful of

companies, Diebold Election Systems included, have successfully concluded this process.

Allegation #76 (p. 21):

"There are also pieces of the voting system that come from third parties. Most obviously is the operating system, either Windows 2000 or Windows CE. Both of these OSes have had numerous security vulnerabilities and their source code is not available for examination to help rule out the possibility of future attacks."

Response:

The report fails to identify any possible exploits to the Ballot Station among the apparently "numerous" security vulnerabilities. The application is, for the most part, self-contained. Outside of the input of the touch screen and the smartcard reader, there is simply no vector for an OS attack.

Allegation #77 (p. 21):

"Besides the operating system, an audio library called 'fmod' is used. While the source to fmod is available with commercial licenses, unless the code is fully audited there is no proof that fmod itself does not contain a backdoor."

Response:

Fmod is merely a sound library. The possibility of tampering with encrypted results record through the playing of an audio sound is, at best, "academic."

Allegation #78 (p. 21):

"Due to the lack of comments, the legacy nature of the code, and the use of third-party code and operating systems, we believe that any sort of comprehensive, top-to-bottom code review would be nearly impossible. Not only does this increase the chances that bugs exist in the code, but it also implies that any of the coders could insert a malicious backdoor into the system. The current design deficiencies provide enough other attack vectors that such an explicit backdoor is not required to successfully attack the system. Regardless, even if the design problems are eventually rectified, the problems with the coding process may well remain intact."

Response:

The correctness of the software has been proven though an extensive testing process, both within the company and by independent testing authorities, and ultimately though logic and accuracy tests by the election officials themselves. A "comprehensive top-to-bottom review" is undertaken regularly. Furthermore, such a review should (and does) concentrate on those parts of the code that tabulate vote results. These modules make up only fraction of the code. The assertion that there are *any* exploitable attack vectors is false. The implication that malicious code could be inserted into the system is baseless.

6.4 Code completeness and correctness

Allegation #79 (p. 21):

> *"It is also unclear whether later versions of AVTSCE were subsequently created. (Modification dates and locations are easily visible from the CVS logs.)"*

Response:

> Diebold Election Systems' software is under constant development and improvement to meet customer needs and ever-changing statutory requirements.

Allegation #80 (p. 21):

> *"These comments come in a number of varieties. For illustrative purposes, we have chosen to show a few such comments from the subsystem that plays audio prompts to visually-impaired voters.*
> [2] Notes on bugs that need fixing:
> `/* need to work on exception *caused by audio*. I think they will`
> `currently result in double-fault. */`
> *There are, however, no comments that would suggest that the design will radically change from a security perspective."*

Response:

> It is common practice to identify areas of software that require attention. The code reviewed for this analysis was in development and incomplete.

7 Conclusions

Allegation #81 (p. 22):

> *"We found significant security flaws: voters can trivially cast multiple ballots with no built-in traceability."*

Response:

> This is false. See responses to section 3.1.1, above.

Allegation #82 (p. 22):

> *"[A]dministrative functions can be performed by regular voters…"*

Response:

> This is false. See responses to section 3.1.2, above.

Allegation #83 (p. 22):

> *"[A]nd the threats posed by insiders such as poll workers, software developers, and even janitors, is even greater."*

Response:

> No credible vectors of attack by janitors or anyone else were provided by the authors.

Allegation #84 (p. 22):

> *"Based on our analysis of the development environment, including change logs and comments, we believe that an appropriate level of programming discipline for a project such as this was not maintained."*

Response:

> We respectfully disagree with the authors' assessment of our development capabilities. They never contacted Diebold Election Systems to understand our software development processes and safeguards.

Allegation #85 (p. 22):

> *"In fact, there appears to have been little quality control in the process."*

Response:

> It is not possible for the authors to access or speculate about our quality control processes based on an outdated cvs log.

Allegation #86 (p. 22):

> *"For quite some time, voting equipment vendors have maintained that their systems are secure, and that the closed-source nature makes them even more secure. Our glimpse into the code of such a system reveals that there is little difference in the way code is developed for voting machines relative to other commercial endeavors."*

Response:

> In fact, the development process followed common professional software engineering practice.

Allegation #87 (p. 22):

> *"In fact, we believe that an open process would result in more careful development, as more scientists, software engineers, political activists, and others who value their democracy would be paying attention to the quality of the software that is used for their elections. (Of course, open source would not solve all of the problems with electronic elections.)"*

Response:

> The difficulties with an open source model as applied to the election industry are touched on in section 1.2, but it is a complicated topic that is outside the scope of this response. It is worth pointing out, however, that scientists, software engineers, political activists and others who value their democracy have never been prevented from developing an open source voting solution to compete against the current closed-source vendors in the marketplace.