12
Open Source Exam

This chapter delves into unavoidably technical areas. This presents a challenge to the reader if, like me, you don't have a computer background. Even if you don't understand the specifics of the flaws uncovered, the gist of the problem is apparent. You will see our evolution from curiosity, to concern, to alarm as we unravel computer programming that runs the Diebold voting system.

Aside from looking at file names, I wasn't much help in analyzing what was in the FTP files. But in June 2003, Diebold voting files began to be examined at a forum called DemocraticUnderground.com, and we learned that people are deeply interested in how their votes are counted.

"This is dangerous," someone explained, to everyone's surprise. "Bad things could happen. Very bad things."

Can someone please explain to me how our "democracy" turned into something where ordinary citizens can get arrested just for looking at how their votes are counted? No, I'm not asking you to explain the "Digital Millennium Copyright Act" (DMCA),[1] which in Internet circles is almost as controversial as the Patriot Act. The DMCA was designed to clamp down on music swapping, but somehow it turned into a tool that can eliminate free speech without due process. It may punish copyright violations with jail time. Some people say the DMCA might be used against anyone who studies the software that counts his votes.

What I want to know is this: How can we call ourselves a democracy if we are so afraid of the consequences that we don't dare to inspect our own vote-counting system? What I'm looking for is an explanation of how *scaring people* who simply want to make sure their votes are counted properly can possibly be the right approach to a robust democracy.

Apparently, this looking at how we count votes is dangerous and (possibly) forbidden — but no one seems to know for sure. Lawyers confess to uncertainty as to whether looking at vote-counting files found on an open Web site can be permitted.

For several months, I considered this issue. As of the writing of this book, I've not yet been able to get a straight answer out of anyone. Here is what *I* came to believe, after much thought: I think that examining our voting machine software is not only a legitimate activity, but it is also our civic duty. For queasier souls, I offer these statements in defense of this endeavor:

1) These files were publicly available.

2) Examining them is in the public interest.

3) Our objective is study and review, not copying and selling voting systems.

4) In a democracy, vote-counting should not be secret in the first place.

The Internet is alive with message boards, chat rooms and forums. People go to these Web sites to meet and converse with each other, using "screen names" so that they can feel free to express any opinion they like. DemocraticUnderground.com (DU) is a rapid-fire political discussion board with more than 35,000 participants. Because this kind of venue provides a feeling of safety and anonymity, citizens used it to examine our voting system.

I perused more than 5,000 comments about voting systems from DU, and I think you'll agree that the excerpts from this body of work (screen names changed) show a remarkable picture of democracy in action.

"I haven't seen the Diebold machines or how they operate," commented "Cleaver" on DU, "but in my precinct, we have a numbered ballot we fill out that is scanned into a machine. In case of a questionable result, the numbered paper ballots can be used to verify results by a hand count. The Diebold machines should have something similar."

Three months later, Cleaver got a rude awakening. He learned that he has indeed been voting on Diebold machines and that a security breach was discovered right in his home county.

After sitting on the files for four and a half months, I was dying to know what was in them.

"What could this thing possibly be doing to need so much source code?" asked "Romeo," a computer programmer. "I have built systems ten times more complex than any imaginable voting machine in one-hundredth the source code space. Sometimes when programmers don't know what they are doing this is the result – lots of cut and pasted functions that are almost the same, tons of obsolete but not removed code ... Ugh."

Another programmer did not find the quantity of code unusual.

"Given that professional programming is complex by its nature and professional programmers are often messy tasteless people by 'normal' social standards," said "mortal," "I'd be surprised if it didn't look like this. In fact, while the sample in question is small, it looks like at least half of the source is visual C++ generated from templates by click&drag, by virtue of its unpleasant-to-type words.

"Once the compiler gets hold of it, chops logicals and optimizes loops, you'll never know how crappy the source looked anyway ... there are actually contests (such as the infamous 'obfuscated C contest') to write the most convoluted and inscrutable programs possible."

A participant called "BettaWatchYerVote" didn't think we'd find evidence of tampering in the computer code.

"I don't think it's likely that you can prove anything with the source code. You won't find a function called "double_GOP_Votes" that does fake counting ... nevertheless, we could very well find back doors, which aren't that uncommon, that would allow tampering."

Some participants argued about the discussion process itself.

"The thought struck me after reading the third or fourth message that this dialogue should not be on a public forum," said "ErgoWeAre."

"Why not? This is the very underpinning of democracy we're discussing here. If there was ever a need-to-know issue for the general public, this is it," replied mortal.

Others suggested the most efficient ways to hunt for vote fraud.

"Have any empirical tests been done?" A citizen we'll call "Ovaltina" often defended Diebold, and provided an alternate point of view. Here, Ovaltina suggested ways to test the code.

"Meaning, generate a large amount of output with the code, and analyze that output, looking for anything the least bit funny, then going back and then focusing on those funny results to look for foul play."

A forum participant called "Bibbidi-Bobbidi-Boo" had a different approach.

"OK, so you've got your haystack and you're looking for the needle ... Here's how I'd approach this problem. ... I'd begin by doing a bit of analysis on how the system is structured. Isolate the important data types (that voter info one is a good example) that someone might be interested in modifying...

"After that, I'd go a few levels deeper with the functions that are doing the data modifications (look at the functions that are called by those functions.) I'd begin to chart out the 'life of a vote' in the system...

"...[I'd look for] code that does not appear to do what it's comments say it's supposed to do; code that is completely undocumented; any code that seems to be manipulating memory in 'weird' or unnecessary ways. God help you because this is in C++."

One participant pondered new DMCA legal issues.

"Discussion cannot be considered illegal under the DMCA," said "Clark Kent," a programmer who had noticed that Diebold passed around other people's proprietary code on its site. "... By making this third party code available freely, Diebold was violating the DMCA...It's unfortunate that Diebold allowed Microsoft source code to be publicly available on one of their FTP servers."

Participants debated whether the curious phrasing in some of the user manuals indicated security weaknesses, or simply imprecise writing.

"Look at this sentence," said "Jolio," — "*When you have finished entering the totals for a precinct, all Check values must be zero in order for you to proceed to the next precinct. If necessary, you can make up the difference by putting the number in the Check tally in the Times Blank field if the race is a Vote For One race. If not, you may have to perform some additional calculations to make the Check value equal zero.*"

Several technical writers participated in the analysis. One, who called himself "Crapper Dan!" couldn't decide whether the previous passage was badly written or contained instructions on how to fudge the numbers.

"I'm a technical writer, and even *I* can't figure out if that says what we think it says or not. Enter that one in the STC's 'Worst Manual of the Year' contest," he said.

Citizens examined the built-in "manual entry" feature, wondering what it was used for and what controls were in place to prevent its abuse.

"Why are they entering manual votes?" asked Jolio. "If we have optical scanners reading absentee and touch screens reading polling votes (and the touch screens also read the challenge votes) — what is the purpose of manual entry?"

"My guess," replied "K3Park," "[is that] the optical scan machines may not be integrated into the same computer system as they are using to run the GEMS software. So (I am guessing) the data has to be entered manually. Even [if] the optical scan machines WERE on the same computer, it might be necessary to enter the data manually if there is no standard protocol for transferring the data from the "optical scan" app to the GEMS software. Another possibility is write-in votes or provisional ballots."

This seemed like a good explanation, though an examination of internal memos, which surfaced later, indicated that both touch-screens and optical scans are integrated into the same GEMS program.

Jolio was still concerned. "That could be the reason for it, but if so...what security measures should it have, at a minimum? Because, manual entry might have a legitimate purpose for entering absentee votes, yet provide a back-door for tampering also."

Clark Kent went looking for the source code which controls the manual entry function.

"Unfortunately, a key piece is missing, manualentry.cpp," he said. "It's documented, but is not there."

The discussions began to attract more programmers. One, who we'll call "Rummage," was particularly interested in the central count "GEMS" system, recognizing that it could provide a key attack point.

"That's right," he said. "The code for the GEMS Server is the key and it ain't here."

Concern soon turned into criticism, as citizens noticed ommissions and weak auditing procedures.

"It took 'em three years to log manual entries, said "Lucille Goldman," a programmer whose criticism stung all the more because she often defended Diebold. "Sheesh!"

"I see the section on manual entry." said Jolio, after reviewing the user manuals. "Not a word in it on who is allowed to do it — presumably, must be someone with admin privileges, but I note this manual also has a section for remote access to the database (why does any election supervisor need remote access to their computer for voting program tasks?) And uh — wouldn't you say that a key event to log [in the audit] after launching the election would be to log the closing of the election? Not a peep, they just go on and open another election."

"You call that an audit log?" asked Lucille Goldman. "Everybody's [logged in as] 'admin.'"

"Topper," who works with government procurement and computer programming, was concerned about holes in the documentation. "More damning ... is that there doesn't seem to be a document detailing policies and procedures for security both at the user/institutional level and the hardware/software level. There needs to be a document detailing who is entitled to do what with the system."

A programmer who I'll call "BlueMac" pointed out a series of agonized comments by Diebold programmers, found in the source code for card readers and touch-screens.

"They have had one hell of a time with standard magnetic card readers. Programmer frustration comments are rampant in this series of modules."

"The thing that disturbs me," said a participant we'll call "OutofTouch," "is the comment saying 'add this after it get backs from certification' (or however it's worded). While it's not necessarily nefarious doings — it could be they modified a function, and the mod was crashing, so they didn't want to insert the update until it was 'stable' — the note does imply that there may be a non-certified build in use."

Of course, anonymous participants on an Internet message board are of no help at all if you want to document problems in a formal way. With the Internet, you never really know whom you are dealing with; a fellow who joins a singles forum may think he's chatting up a

buxom blonde named Inga from Denmark while he's actually charming a 400-pound farmer from Iowa named Ralph.

Among the advantages of this informal review format was the perception of protected freedom of speech, facilitated by anonymity. A disadvantage of doing an open source investigation using a public forum was that we knew very little about these people's credentials, except what they volunteered.

This public "open-source investigation" had many drawbacks, but it did attract intellectual talent and ultimately led to the first public evaluations of the software outside the voting industry itself. One of the contributors, whom I'll call "Goody Two-Shoes," explains how he came to be concerned about the Diebold software:

"I'm the poor schmuck who configures brand new, untested, computer systems designed by teams of highly educated hardware engineers and loads brand new untested software designed by highly educated teams of software engineers and then performs the 'debug' to make them work together. The systems rarely, if ever, work the first time. It's been my job to be the final arbiter of the finger pointing battles between the two engineering groups who each claim the others product is at fault."

He goes on to describe how he can quickly locate problem areas in the source code.

"...Programmers tend to be extremely logical thinkers. They exhibit that logical thinking in the way they write their comments into the source code. Each section of code produced by a 'good' programmer has a 'plain english' explanation of what that section does. You might call it a 'professional courtesy' to other programmers who have to work with their code downstream. It's [looking at the comments] a shortcut that quickly lets you know where to focus your attention rather than study every line of code to find what you're looking for."

"When you find comments [in the source code] that say things like: *'this is baloney, you don't have to do this, this function is already built in to XXXXXXX, just use the XXXXXX command'* or *'the (insert critical flag here) flag is broken so I did this and that to get around it'* and even things like *'I don't know why you want me to do this, it will let this and that happen....unless that's what you want to happen then I guess it's OK'*!

"Comments of this type naturally lead a good programmer looking for problems to investigate what is going on in those routines."

The contributor known here under the screen name "Rummage," studied computer science under a Nobel laureate at Carnegie-Mellon University. In real life and under his normal name, he designs databases for critical applications in the medical field.

"So far, that's the story of the last few days," he wrote. "From databases with no foreign keys (read no referential integrity), unprotected transmission code, ample opportunity for buffer overruns right to PCMCIA slots for wireless modems. Not so much nefarious code as a system with so much opportunity for hacking/fraud as to invite cheating. "

"...as for structure and understanding the DB [database], there are no relationships and the Primary keys are not defined as Access Primary keys. This will make reconstructing the schema a little harder. I don't think a DBA [database analyst] designed this.

"No referential integrity — no autonumber primary keys. Bad for maintaining a reliable database — good for adding and deleting data at will."

I've spoken to many of the participants of the voting machine examination who seemed especially insightful, and they often have impressive credentials, but to most of the world they are anonymous so you can't really know. These informal forum discussions are more akin to casual conversation in the cafeteria than to academic research.

Here are comments from "t_device," a European participant who concurs with "Rummage" about weaknesses in the database design.

"The fact that they're using Access disallows relationality ... When using a decent database, SQL Server Sybase etc, for example, constraints, triggers, stored procedures, packages, relationships, views, etc are all maintained inside the database — that's where all the business logic resides in a well crafted modern application.

"With Access, however, you're dealing with basically a toy database, and since all of the above are missing, it is common to join tables on the fly using the data connection and SQL code embedded into the program itself...

"... I could be wrong, but in Access, if you have write capability, you have delete capability...the security features are very limited.

"Security is not something I would consider claiming to have for *any* Access-based application since about any user can gain access fairly easily ... and if you'd ever tried to upsize from Access you wouldn't be touting it as a good thing. Data types get changed, boolean fields don't translate ... it certainly shouldn't be used in a mission critical voting application."

On forums, people are free to make opinionated, dogmatic and sometimes mistaken statements, just as we do in casual conversation on the subway or in a bar. The Internet culture uses forums and message boards to consider perspectives and ideas, but never for a definitive answer. One reason: It all depends who's chatting that day.

Lucille Goldman took issue with criticisms posted by t_device.

"Let's not get into a pissing match," she wrote. "My upsized applications run very nicely to this day. Yes, it's not perfect, but I've used ERwin for documentation and Access is much easier for smaller projects. You get the application running, produce the relational schema and put it on the server. You may choose to develop on the target system. I prefer my method. I hope we can treat each other respectfully."

"I believe we have been civil," said t_device. "If that's not the case, let me know. Apparently we have a difference of opinion. That's healthy. I have upsized a few Access apps and I've developed in it, so I'm not speaking off the top of my head ... Anyway, let's drop the Access better/worse convo and stick to the voting application."

Most programmers concurred that Diebold's use of the Microsoft Access program indicated weaker security than desired.

"Go over to slashdot," said "abcxyz." Slashdot.org is a forum for computer people. "Try talking about 'security' and 'Access' in the same breath and see how seriously they take you over there — they won't even dignify you with a response, they'll just laugh at you and spray you with onomatopoetic responses like this:
choke
wheeze
bwahahahahahahahahahahah
gasp
Wait, these things are already in use?!?
thud
...because all programmers know there is no security in Access."

"If you want to know why Access is a bad idea," said Goody Two-Shoes, "just do a Google search for 'Access, vulnerability' and browse through the 951,000 hits!"

"Now THAT is a legitimate beef re: Access," agreed Rummage. "And the lack of referential integrity (which could have been done, but wasn't) only fuels my suspicions."

By now, many people have read criticisms about the Diebold voting system, but back in June, 2003 no such reports had yet surfaced. I found myself staying up late into the night, just to see what else the programmers would find. They were especially critical of the audit log, cited by Dr. Brit Williams as a key component to the security of Georgia's voting system.

"Good point about database audit log tables," said a programmer we'll call "gandalf." He pointed out that the Diebold audit log was not constructed properly. "Very easy to delete any entries. Though there should be some sort of audit ID (in any good database design) that records the sequence of audit log entries which would indicate that a log entry had been deleted."

The audit log. The more people looked at it, the greater the dismay. What citizens were finding simply did not match claims made by Diebold and its regulators. From Dr. Brit Williams:

> "Overall security of any computer-based system is obtained by a combination of three factors working in concert with each other: "First, the computer system must provide audit data that is sufficient to track the sequence of events that occur on the system and,to the extent possible, identify the person(s) that initiated the events." [2]

The following statement, taken from the Diebold document used to sell its system to the state of Georgia, refers to a touch-screen audit trail:

> "Generated entries on the audit log cannot be terminated or interfered with by program control or by human intervention." [3]

Not quite. The server at the county that accumulates all the incoming votes (GEMS) is an attractive tampering target, and altering the critically important GEMS audit log is quite easy.

"Bev, in what way is it significant that the audit log can be rewritten?" asked a programmer we'll call "Mae West." "I'm puzzled by

that," she said, "because as several people said (I among them) early on, physical control of a machine always means you can overwrite whatever you like. The trick is to keep the bad guys from gaining physical control."

"The significance is that in letters from certifiers and in documentation provided to certifiers and to the public, they took the curious position that the 'audit log' was a primary means of security protection," I replied, referencing Dr. Williams' report.

"Hmmm...did they say in what way?" she asked. "Because if they said it as you implied here (i.e., the existence of an audit file is enough), that would actually be hilariously funny if it weren't so serious. Nerds the world 'round would be cleaning their keyboards and monitors after failing to laugh and swallow at the same time."

Looking at the Microsoft Access database used in the county vote tabulation system led to concerns about the integrity of the GEMS program as a whole. Interest in the GEMS program began to take on a life of its own on the forums.

"Here's the best part," said BlueMac, "With GEMS (server) installed on my computer, I was able to create a user name ("me") with a password of my choosing ("mac") and assign myself ADMIN capabilities. This was without ever signing into GEMS....all I had to do was create a new database and I was in like Flynn."

Diebold was not without its supporters. "Ovaltina" pointed out that a database maintenance application might provide the security that GEMS was found to lack.

"The votes end up in a database. Whenever there's a database, it makes sense that there would be a database maintenance application. Always preferable to have such an application controlling data entry, to control access and make sure everything agrees, catch entry errors, log activity, etc.

"Without this data entry procedure, what would stop someone from going directly into the database and committing fraud that way? I think you said before that it's an Access database? So open up the database with Access and put your phony votes in. So what I'm saying is the mere ability to edit votes isn't all that menacing to me, because it doesn't say that there are no procedures to prevent it from being abused. Maybe elsewhere in the system, or maybe completely outside the system."

The GEMS program at the county, which pulls in all the polling-place votes, would not be as vulnerable if a report was run directly from the voting machines themselves before any data was sent to the county tabulator. That way, if someone tampered with GEMS, (even if they also tampered with the incoming data from the polling place), the numbers wouldn't match. A forum member called "DanglingChad" who had election experience weighed in:

"Full precinct reports are required by California state law as well as others. The Diebold system better be complying with the requirement ... California Code 19370 states... 'At the close of polls... at the precinct...one copy of the statement of return of votes cast for each machine shall be posted upon the outside wall of the precinct for all to see. The return of votes includes each candidate's name and their vote totals at the precinct. During certification of voting machines, the Voting Systems Panel requires evidence that the procedures of each vendor include this process...'"

If someone tries to hack the GEMS program, posted reports at each precinct (as long as they were printed before any upload of data) would make fraud at the central tabulation stage significantly more difficult, though a clever insider could get around this safeguard. Unfortunately, as you'll learn in the next chapter, this procedure was not followed in the 2003 California gubernatorial recall.

* * * * *

Most of us are given some amount of common sense (as long as sex or money isn't involved), and when we meet up in a group and bring our experiences into the picture, we can make some good, solid decisions. People familiar with accounting and bookkeeping began to weigh in on the online voting system examinations, and they sometimes took software engineers to task for their failure to understand basic accounting principles.

At issue in this conversation were statements by computer scientists that it was sometimes permissible to design tabulation systems in which totals could be manually overwritten.

No way, said a citizen who went by the moniker "ItAllAddsUp." "Each and every vote should exist as a distinct and unadulterated record of one citizen's transaction, probably one or more copies should be generated simultaneously, and everything should be 'journalled' ...

"Since voters are not allowed to recast votes, no possible set of circumstances can possibly exist to justify changing those records. ... Every change, every addition or subtraction to votes, has got to be a separate transaction. As a matter of fact, what reason should ever exist to make a change that has an intrinsic value of more than one?

"If a fifty vote change has to be made, then you had better show fifty transactions ... If you need to cancel fifty votes, then you had better show which fifty votes that you are cancelling. Damn and double damn. There is absolutely no technical reason in the world why this cannot be done.

"One vote today is the same as one vote in 1776, which is the same as one vote in 1876, which is the same as one vote in 1976, which should be the same as one vote in 2076.

"What is so hard to understand about that for these computer geeks?"

"Cleaver" pointed out that counting votes was a form of bookkeeping, and explained why the same kinds of safeguards should be used.

"Accounting practices are double entry, not only because of mistakes, but also fraud. Two sources are better than one. So there should be an accounting trail to verify results, especially when there is a question of accuracy ... It doesn't have to be paper but it should be a traceable source document."

Most of all, citizens weighed in with demands for transparency. They chafed at corporate claims to privacy for votes that belong to all of us:

"Government has no business hiding behind proprietary computer code in proprietary voting machines," said ItAllAddsUp. "If the government wants us to use a number 2 lead pencil to mark the ballot, then we damn well better be able to examine that number 2 lead pencil ourselves. We should be able to buy a box of those very same, identical, number 2 lead pencils if we so desire. The paper used for the ballots has got to be paper that can be examined by any who wish. The boxes where the ballots are stuffed need to be able to be examined ..."

As citizens become more concerned about the security of the Diebold voting system, they began to look for remedies, and found that state law often lacks adequate protections.

"States like Georgia have written provisions into their laws that make it impossible to get a machine in dispute adequately inspected," said Goody Two-Shoes. "The Georgia law stipulates that three people, a patent attorney and two *mechanics*, be appointed by law to look at the computerized machines! This is tantamount to appointing two blind men and an attack dog to inspect the machine. If either of the 'mechanics' asks about how the machine works the attorney is there to tell them 'it's proprietary information', you're not allowed to know!'"

Every now and then someone still pops up to tell us that the voting system topic has no legs, or that people just don't care about it. Then explain this: Voting system analysis at DemocraticUnderground.com became kind of an attraction. More and more people tuned in, but at the same time, the subject matter became increasingly technical, while the tone of discussions reflected more urgent concerns. Occasionally someone would sigh and raise their hand:

"Can anyone explain what is happening here in simple language for those of us who are non-techies?" asked a citizen called "SkiBob." "I can't make heads or tails about what you may have found here."

Well, we're talking about the systems used to count our votes.

"But have you guys found anything? Everybody seems to be talking in very excited tones using terms I can't understand."

(Sorry). Yes, people were finding things. Many of the things they found were eventually found also by researchers at Johns Hopkins and Rice universities,[4] in a report that ended up in *The New York Times*. It was the "increasingly excited tones" that directly led to the events that produced that report.

"Attn: Bev Harris... look at the cryptographic routines of the voting system. I've just started to go through this system and have a few little snide remarks to make," said the computer professional who went by the name "Topper." She was concerned about the possible use of a free, open-source cryptography program which is no longer supported.

"The problem with using open source with no support is getting a timely answer to your question," she said. "Ergo, if there is a security problem during an election, you are stuck with fixing it — which you may not be able to do yourself in a timely fashion."

"Actually it's not so bad," countered a programmer called "MidniteMunchies." "I'm a programmer and have used that code before.

It isn't very well documented and the code is very confusing due to some funky overuse of C++ templates.

"... However, I would have to agree that any kind of election software encryption should be based on a standard commercial or government supported encryption solution rather than someone's hobby encryption project."

Encryption is important because we don't want people intercepting and changing the votes as we record, transfer and tally them up. Used correctly, encryption can prevent some kinds of unauthorized access. Discussions about encryption were about as opaque as it gets on a political discussion board, but even those of us who are not techies could tell that Diebold's encryption was causing significant concern.

"I'm not sure any of the encryption is actually used anywhere," said "PoodieToot." This was not reassuring.

"Since you brought it up, I thought I'd see what algorithm they ended up using. The problem is, I've grepped all over the files, and I don't find any header file inclusions from the crypto library *anywhere*, other than the crypto library. I can't see where the other CVS modules call any of this stuff at all.

"You know," PoodieToot continued, "they *could* have gone with OpenSSL — it's free, and supported by far, far more users (and corporate users, such as Apple and IBM for example). But, then again, it doesn't look like they are using any of it anyway..."

Uh-oh. When answers about Diebold's cryptographic methods were found, they weren't the right ones.

"Mystery solved," said PoodieToot, "but ... oh, no ... I found what they are actually doing for encryption. They have their own implementation of DES in Des.h. Here's the bad news...it looks like the DES encryption key is *hard coded as a macro*!!!!!

"AAAAIIIIIIIEEEEEEEHHHHHHHH!!!!!!!!!!!

"I'll leave discovery of aforementioned key as an exercise for the reader... Good God....."

PoodieToot's discovery brought the Internet board alive with the forum equivalent of shrieks and moans.

"Ooorah!!!!!!! Yeah," said Topper. "I've found the DES.h file...and will start trolling through this... If you've hard coded your key and

left it just like the public implementation, then it would not be that hard for a hacker to figure out how to get into your system."

Programmers were beside themselves upon viewing the blatant security flaws, and soon they were finishing each others' sentences.

"—It would end up as a static string in the executable file," said PoodieToot. "And you can tear the static strings out of an executable to view them faster than you can blink your eyes."

"In your best 50s announcer voice," said Romeo sarcastically, "now *that's* real data security! (cough, cough.)"

The more people learned, the more alarmed they became.

"These things actually use PCMCIA cards?" asked Clark Kent in dismay. "Huge potential security breaches! Think of the new stuff out there. This is Windows CE-based code. Couldn't the existence of these drivers open up any one of these machines having a PCMCIA based wireless network card installed surreptitiously, allowing remote access via airwaves?

"They're using simple PCMCIA ATA disks These things are basically notepad PC's and the security is almost non-existent. How many local governments will be up on the sophistication required to implement WEP with encryption and hiding SSID's for wireless networks? Heck, you wouldn't even have to hack the wireless network to get around these things, all that is necessary is to pop out one hard drive of results and pop in another with new results preconfigured."

A tech who went by the name "Razmataz" was shocked at finding evidence of wireless communications in the voting system.

"Wireless programming required? Are they nuts? I thought I'd been following all the 'electronic voting machine' strategies but that's one I missed. I'm a techie, 36 years in the business, some of it with reading punch card votes and optical votes. Wireless programming capability is just plain nuts. That's a security hole the size of a 747.

"That would mean somebody could walk near the voting area (even outside the building), connect to the voting machines via wireless network, and make changes to the voting programs and/or the vote counts"

"I think we've found a potential hole where somebody could alter results remotely with nothing going over any wire," said Clark Kent. "Somebody needs to seriously wardrive elections sites using these things."

"Ah... That is serious bad news if they are running these terminals wirelessly and only relying on WEP for security," said "RescueRanger." "That is enough to fail a security audit at any fortune 1000 company.

Yet, RescueRanger held onto hope for a bit of good news.

"On the other hand, wireless can be extremely secure, more secure in fact than most wired communication if done properly and with the right equipment and design. To do it securely, would require fairly recent (and proprietary) technology...certainly not anything that is anywhere near five years old."

Perhaps we should all calm down, intoned a forum participant who went by the name "spock."

"You are assuming no encryption. Because this is wireless does not mean no encryption is being used. WEP anyone? Proprietary encryption perhaps? But then again it could be none is."

"The onus is on the local election administrators," said Clark Kent. "though I have my home wireless network locked down so tight most wardrivers will take one look at all of my security measures and drive on down the street to the guy who is advertising an SSID that is the default on the access point he installed and has never changed the admin password.

To most of us, this conversation might as well have been conducted in Greek, but we couldn't stop tuning in. Clark Kent explained the security flaws in language only a cryptologist could love.

"Even I know that with 128 bit encryption using WEP, no advertised SSID, and a MAC Address list can still be cracked. MAC addresses can be spoofed relatively easily and brute force can break the 128 bit encryption if you've got the processor power. Even with encryption, it can be cracked. Now tell me how many of the local election boards you've had experience with are sophisticated enough to implement WEP, let alone MAC Address access lists? Add to that the fact that there is a ton of code that could hold back door access and this thing is rife with potential abuse.

"Nope, this doesn't even compare to the potential for pushing out chads on hundreds of cards with a pin so they register as double votes and thus are spoiled ballots. The potential for abuse is magnitudes above this. If the government does not require an independent code review by at least three different companies, it's not doing its job."

Remain calm, spock suggested: "I trust you are aware... The chances of breaking 128 bit encryption with a brute force approach could very well take centuries with just about any computer on the planet?"

No, no, no. "A 128 bit encrypted file and the encryption level on WEP are two different things," said Clark Kent. "I assure you, WEP is crackable. A PGP file with 128 bit encryption is, as you stated, not easily crackable. And when database files have passwords that are the name of the county where votes are counted, how secure is this system?"

It got worse.

"Perhaps this programmer's comment in the Results Transfer Dialog file [TransferResultDlg.cpp] will answer that question for you," said BlueMac. "'Changed the election.dbd file to only store ASCII code not unicode to make it compatible between windowsNT/95/98 and WinCE. The conversion from ASCII to unicode, if required, is done when the data is retrieved from the database. Note: This does not affect RTF data since it is always stored in ASCII.'"

Though many of us didn't exactly understand it, this last news, apparently, was pretty bad.

"Straight ASCII????????" wrote Clark Kent. "For compatibility with Windows 95/98/NT???? On February 15, 2001?????"

A typographical wink was spock's response: "Why not? ;o"

"That's some encryption there! Straight ASCII for backwards compatibility on operating systems that are obsolete," said Clark Kent. "This makes a lot of sense for a system we are supposed to trust the future of the world to."

He wasn't sure it was a disaster, but spock ceased to be at all reassuring.

"I believe it is talking about the unencrypted values for backwards compatibility when being viewed. But then again that's another problem with leaked source that may or may not be final, you can't be sure."

"And that's the problem with computer voting systems, isn't it," said PoodieToot. "You can't be sure."

But why not use widely accepted encryption techniques?

"If I were the guys doing openssl, I'd be real pissed off right now," said mortal. "That blows chunks. I guess assigning a public/private key pair to each networked voting machine is too difficult for the people

entrusted with the lifeblood of democracy?"

"Seems a Congressional investigation should be next," said "SPacific."

If anything should have a congressional investigation in full view of TV cameras, the voting industry should, but as of the writing of this book, it hasn't happened.

* * * * *

What came next was a quiet phone call on a Sunday morning.

Over the course of a year, I had consulted with about two dozen computer techs. Several are not on DemocraticUnderground.com because they are Republicans. I met one on Free Republic, a conservative forum. Voting-system integrity is a truly nonpartisan subject. Democrats, Republicans, Libertarians and Greens — everyone but the Charlatan Party, I guess — all respond the same way when someone says, *By the way, we won't be auditing the vote, thank you.*

Among my sources is a computer programmer I'll call "Cape Cod." He rarely calls me and has always been irritatingly discreet about his examinations of the Diebold files. When he calls, his clipped, East Coast voice provides no unnecessary words and gives very tidy explanations.

The best programmers explain things in a very concise way. I'll keep asking questions until I understand the answer or the other person starts shouting at me, whichever comes first. But highly skilled programmers are extremely organized thinkers, and it is easy to follow their explanations. Cape Cod is such a person. His explanations of complex computer concepts follow this simple, linear fashion: *Here is A, and I'm going to take you to B. Take hold of A, and walk just this way, and I'll describe the scenery as we go. Now, here we have arrived at B; did you enjoy it?*

He never calls unless he has something to say. He made one efficient, four-minute call to explain how a voting system might be able to cheat with "zero reports," for example:

"It's quite simple, really; your goal is to stuff the electronic ballot box while at the same time generating a report at the beginning of the election which tells you that zero votes have been cast, proving the ballot box has *not* been stuffed.

"Here's what you do: You stuff the ballot box by entering two vote totals that cancel each other out: 'plus 50 for Truman, minus 50 for

Dewey.' You have thus created a spread of 100 votes between the candidates before the election begins — yet because +50 and -50 sum to zero, you have added no extra voters.

"To make the report read zero when you start the election, simply instruct the code to put a string of zeroes into the 'zero report' if there are any negative numbers in the ballot-stuffing area, but it must only do this if there are no other votes in the system. And by designing a database without referential integrity, you can arrange for the evidence of this ballot-stuffing area to fall off the radar."

One Sunday morning while I was still in my bathrobe, I received one of Cape Cod's rare phone calls.

"Go to your computer. I want to show you something."

He proceeded to walk me through the process of rigging an election using a real Diebold "GEMS" program, with a version used in a real election, with a vote database for Cobb County, Georgia.

Bypassing the supervisor password

If you install GEMS and make a new "test election," the manual tells you to use the password "GEMSUSER." Close your test election and open the same file in Microsoft Access, and you will find an encrypted version of the "GEMSUSER" password. Copy the encrypted password and paste it into any election database. You don't really need Microsoft Access; a simple text editor can also be used. By doing this, you can bypass the password in any GEMS vote database.

You can grant yourself supervisor privileges by making yourself an "admin."

You can add as many friends as you want. (I added 50 of mine and gave them all the same password, which was "password.")

It gets worse: If you go in the back door, you don't even need a password.

A triple set of books

The GEMS program looks and feels very secure when you work with it. However, running behind the GEMS program is a database using Microsoft Access. When you open an election in GEMS, it places

an election database in a folder on your computer. Anyone who can get at the computer, either with physical access or by hacking in, can open this election file; right-click it, open it with a text editor or with Microsoft Access, then just go right in the back door. This technique is not certified or authorized, but it can be done anyway. You don't need any special computer skills. At the time we examined the files, if you could right-click a file and type, you could alter the votes in GEMS.

Back to Cape Cod.

"Here's what we're going to do," he said. "We'll go in and run a totals report, so you can see what the election supervisor sees. Then I'll show you something unusual."

I opened the GEMS program and ran a totals report, showing the overall election results. Then I ran a detail report showing the results in each precinct.

"Now, open the file in Microsoft Access."

"Close out of GEMS?"

"No, Access is configured for multiple users."

OK, I didn't know that. Two people can wander around in the vote database at the same time without bumping into each other.

Remember that there are two programs: the GEMS program, which the election supervisor sees, and the Microsoft Access database (the back door) that stores the votes, which she does not see.

You can click a table called CandidateCounter, which will show you how many votes the candidate has accumulated for each polling place.

Cape Cod showed me another table in the vote database, called SumCandidateCounter. This table had the same information as CandidateCounter, but we observed that it had two complete sets of the same information. One set was marked by a flag, the number "-1." Notice that this gives us three sets of votes.

"Change some of the vote totals in SumCandidateCounter."

"Now go into GEMS and run the totals report."

The totals report showed my new numbers, proving I could alter the report by going in the back door and replacing vote totals with my own.

"Now go back and look at that detail report."

The detail report had the original votes, not the ones I changed. In accounting, this is called having two sets of books. (Or in this case, three. I never heard what the third set of books does. Cape Cod called it the "Lord only knows" table.)

"Why would it be good to have the detail report show the real votes while the summary shows the ones I changed?"

"Because it would allow a manipulated system to pass a spot check."

Altering the audit log

Any time you open the GEMS program, it will show up in the GEMS audit log. (If you go in the back door using Microsoft Access, however, your work will not show up in the audit log.) But suppose you need to erase your activities in GEMS?

In the Diebold system, it seems that everyone uses the same name when they go into GEMS (they all call themselves "admin"), but I wanted to see whether I could become someone new, play around in GEMS and then erase myself from the audit log.

I created a new user by the name of "Evildoer." Evildoer performed various functions, including running reports to check his vote-rigging work, but only some of his activities showed up on the audit log. For some reason, a few of his activities omitted themselves from the audit log even before I tampered with it. But I wanted to erase *all* evidence that Evildoer had existed.

I went in the back door. I expected the audit-log entries to be numbered automatically with something I could not edit. That way, if I erased some Evildoer activities, the numbers would still be there, marking an activity that had disappeared. I was surprised to find that I could just type new numbers over any of the GEMS audit-log numbers, and I could also erase events altogether.

In every version of GEMS that I examined, the autonumbering feature was disabled, allowing anyone to add, change and delete items from the audit without leaving a trace. I simply erased Evildoer.

Going back into GEMS, I ran an audit report to see if Evildoer had indeed disappeared. *Poof!* Gone. As Verbal Kint, in the movie *The Usual Suspects* (1995) said, "The greatest trick the devil ever pulled was convincing the world he didn't exist."

Another thing that seemed improper in the GEMS program is this: You can enter *negative* votes. It is a simple matter to program the software so that it will never accept a negative number. Why should it? A vote total that is less than zero can only be illicit.

The entire process — bypassing the password, changing the vote totals, cleaning up the audit log — took less than 10 minutes.

* * * * *

Scoop Media's publisher knew from my communications on the forums that we had something big.

"Hi, Bev. (New Zealand pronunciation: 'Bivv'). Alastair here. (New Zealand pronunciation 'Alasteh'). What's up?"

"Well, we have a story. With the GEMS program, using one of the databases found on the FTP site, we were able to rig it," I said.

"Hmm!"

"I'm writing it up. I'm not sure which outlet I'm taking it to, though."

"You know, I rather thought this might be a good time to publish the link," said Thompson.

"What link?"

"Oh you know. To the files."

"The files from the FTP site?"

"It seems like a good time, don't you think? I think we should come out with your story at the same time. Get people to it, right?"

"Alastair, that set of files is huge. Do you have the bandwidth?"

"Oh, I think we'll be all right.They have bandwidth to burn."

The story went out on *Scoop Media* on July 8; [5] Thompson ran my story about the hackability of GEMS, along with his own editorial which he titled "Bigger than Watergate!" He has since been roundly criticized for that choice of title, but remember: Watergate took two years to get as "big as Watergate."

Just 16 days later, *The New York Times* ran a scathing report on the Diebold voting-system software by computer security experts from Johns Hopkins and Rice universities. They had downloaded the files, originally from the Diebold FTP site, from *Scoop Media*. At least one new story came out in a major media outlet every day for the next two months. In September, a report written by Pentagon contractor

Scientific Applications International Corp. (SAIC) was published that detailed 328 security flaws in the Diebold voting system, 26 of which it deemed "critical."

The Johns Hopkins/Rice universities report

On July 24, 2003 *The New York Times*[6] ran an exclusive story about "stunning, stunning security flaws" uncovered by four researchers at Johns Hopkins and Rice universities. The report, titled "Analysis of an Electronic Voting System," described many of the findings pointed out by the irreverent bunch at Democratic Underground, but these computer scientists — Avi Rubin, Dan Wallach, Adam Stubblefield and Yoshi Kohno — did a gutsy formal study and put their names on it.

The Johns Hopkins/Rice report was blistering. It quoted source code and delved into Diebold's smart-card security and its cryptographic weaknesses. The report also revealed that one flaw had been pointed out by voting examiners five years ago and still had not been corrected.

Diebold Election Systems came out swinging: The software was never used in any election! Well, it was used in some elections, another Diebold spokesman was reported by *WiredNews* reporter Louise Witt to have said.[7] I called her to ask how solid this quote was. Rock solid, she said, but the quote was pulled a day later in favor of this: "*A small part of the software may have been used in some elections.*"

We were told by Diebold that the problems had been fixed and also that they were never a problem in the first place, because the Diebold software is surrounded by election procedures and physical security, which have neutralized the problems all along. Diebold tells us this, but will not prove it to us.

There are weaknesses in the Hopkins/Rice report. Several sections seem to assume that touch-screen machines are connected to the Internet, but nothing I've seen indicates that to be the case. GEMS servers can connect to the Internet, and GEMS also connects to modems which, in turn, connect back to touch-screens.

The criticism that the Hopkins/Rice report doesn't take into account all the election procedures is, in many ways, correct. It doesn't appear that the authors read the user manuals that go with the software; they apparently did not interview any election officials.

Other areas of the report describe hacks that would be impractical or could not affect many votes at a time. The most publicized security flaw in the report has to do with making extra voter cards (or reprogramming one so that it can vote as many times as you want). These are valid concerns, but checking the number of voters signed in against the number of votes cast is a required safeguard in most states and would reveal such a ploy. This type of hack would also be difficult to achieve on a grand scale; you would have to make rigged smart cards and send people in to cast extra votes at hundreds of polling places at once, which gets into the crazy conspiracy realm.

The biggest taint applied to the Hopkins/Rice report is a conflict of interest on the part of one of its primary authors, Aviel Rubin. Lynn Landes, a freelance reporter, revealed that Rubin had been an advisory-board member for VoteHere, a company that claims its software solves many of the problems in the Hopkins/Rice report.[9] Rubin also held stock options in VoteHere; he resigned and gave back his stock options after Landes had published her article. Rubin told Landes that he had forgotten about this conflict of interest when he wrote the report.

Three more researchers — Dan Wallach, who is a full professor at Rice University, and Adam Stubblefield and Yoshi Kohno, of Johns Hopkins — also wrote the report, and none of them appears to have any conflicts of interest. It seems unlikely that all three would help Rubin slant a report just to help him sell VoteHere software.

The importance of the Hopkins/Rice report:

1) It correctly identifies weaknesses in Diebold's software-development process. The code seems cobbled together to fix and patch.

2) It identifies very real security flaws that can jeopardize vote data, especially during transmission to the county tabulator.

3) The Hopkins/Rice report pushed media coverage into the mainstream. When you are researching this story, you can't even sneeze without finding something new, so coverage of the integrity of our voting system will continue to gather momentum. The longest leap forward in a single day was attributable to the Hopkins/Rice report.

4) The report triggered another evaluation, this time by Science Applications International Corporation

.

SAIC report

In August 2003, the governor of Maryland, which had placed a $55 million order for Diebold touch-screen machines, ordered an evaluation by Scientific Applications International Corp.[10]

If Rubin is said to have a conflict of interest, then SAIC had a whopper: The vice chairman of SAIC, Admiral Bill Owens, was the chairman of VoteHere. Like the Johns Hopkins/Rice report, the SAIC report identifies areas for which VoteHere claims to have a solution.

The SAIC report did a bizarre thing: It redacted the version numbers for the software it studied. Now, the version numbers are what is certified by the ITAs, and it is the version number that gets approved by the states. Refusing to say what version number was studied pretty much eliminates the usefulness of this report.

The report also redacted the entire section on GEMS, stating simply that the program was unsatisfactory. All in all, 131 pages out of 200 were redacted, and if we are to believe Washington State Elections Director David Elliott, even state election officials are not privy to this information. Here is what Elliott wrote when I inquired about how he can oversee elections that use GEMS if he can't read the SAIC report:

"As to your questions about the SAIC report. I share your frustration about the redactions contained in that report. I have read what was published in its redacted form. I have not been able to secure an un-edited copy."

If there was ever an indictment of the concept of privatizing a public trust, this is it. Here we have a for-profit corporation asserting privacy over a report commissioned by the state of Maryland, and the state of Washington, which uses the voting program developed by this company, cannot even find out what this government-commissioned report says.

The SAIC report does validate important findings in the Hopkins/Rice report and identifies many new areas of concern. Because it is heavily redacted, we don't know the details on all of the flaws it found, and many are specific to Maryland. Still, these words, taken from the SAIC report, reverberate:

"The system, as implemented in policy, procedure, and technology, is at high risk of compromise."

Or, to put it succinctly: "328 security flaws, 26 deemed critical."

As this book went to print, another independent study was released, this time commissioned by the state of Ohio,[11] and this time, the study examined systems from Diebold, ES&S, Sequoia and Hart Intercivic.

The study found multiple, critical security flaws in all four systems.