SOFTWARE VULNERABILITIES IN THE BRAZILIAN VOTING MACHINE

Diego F. Aranha, ***Department of Computer Science - University of Brasília, Brazil***
Marcelo M. Karam, ***Center for Informatics - University of Brasília, Brazil***
André de Miranda, ***Center for Informatics - University of Brasília, Brazil***
Felipe B. Scarel, ***Center for Informatics - University of Brasília, Brazil***

Abstract

This work presents a security analysis of the Brazilian voting machine software based on the the experience of the authors while participating of the 2nd Public Security Tests of the Electronic Voting System organized by the Superior Electoral Court (SEC), the national electoral authority. During the event, vulnerabilities in the software were detected and explored to allow recovery of the ballots in the order they were cast. We present scenarios where these vulnerabilities allow electoral fraud and suggestions to restore the security of the affected mechanisms. Additionally, other flaws in the software and its development process are discussed in detail.

INTRODUCTION

The Brazilian Superior Electoral Court (SEC) has been increasingly adopting electronic elections since 1996, culminating in the current scenario where nearly all votes are collected by voting machines and a considerable fraction of the machines have fingerprinting devices for voter identification. Important milestones in the history of the initiative were the first purely electronic elections in 2000, the transfer of full responsibility for software development to the SEC in 2006 and the migration to the GNU/Linux operating system in 2008. Although security testing by independent parties should be a part of the process from the start, as a natural way to improve reliability of elections and reassure that the system provides sufficient ballot secrecy and integrity, it only received significant attention after the software components and human procedures for electronic voting became stable. An important movement in this direction has been the public and periodic testing of the voting systems organized by the SEC since 2009. Despite some undesirable restrictions, these tests allow teams of specialists from industry and academia to independently evaluate the security mechanisms adopted by the Brazilian voting system.

The main goal of this work is to present the observations collected by the authors during their participation in the 2nd iteration of the Public Security Tests organized by the SEC in 2012. Our previous official report of the event was jointly written with the SEC and does not contain sufficient information regarding other security issues not directly attacked by the authors during the event. Our intention is to point out several limitations of the Brazilian electronic voting system and to contribute to its security process. Following standard practices in the security field, we present self-contained descriptions of the observed software and development process flaws with multiple suggestions for correction or mitigation. This way, the interested parties are in an adequate position to implement effective countermeasures. In particular, the main design and implementation problems detected on the security mechanisms of the voting machine software are detailed. An overview of such issues can be found below:

- **Inadequate protection of ballot secrecy:** votes are stored out of order, but it is trivial to recover them in order only from the public data produced by a voting machine and superficial knowledge of the software source code, which is also made public to the political parties. This vulnerability fully compromises ballot secrecy when associated to a partial or complete ordered list of electors.
- **Inadequate use of encryption:** the same encryption key is shared among all voting machines for encrypting the critical portions of their memory cards. These include the voting machine

software and other cryptographic keys required for authenticating election results. Using the classical abstraction of a locker as an encryption technique, this is equivalent to using half a million lockers with exactly the same key, since this is the approximate number of voting machines in operation. This encryption key is also stored in the plain text portion of the memory cards. Using the same analogy, this is compatible to hiding the locker key under the carpet and trusting the secrecy of this location to protect the confidentiality of the key.

- **Obsolete cryptographic algorithms:** the SHA-1 cryptographic hash function used for computing digital signatures and integrity verification is demonstrably not collision-resistant. These specific applications of the chosen hash function have been deprecated for 6 years already. A sophisticated collision in this hash function would allow an insider attacker to construct fake voting software capable of producing election results indistinguishable from the correct outcome.
- **Inappropriate attacker model:** significant emphasis is put on the design of security features resistant only to outsider attackers, when insider threats present a much higher risk.
- **Faulty software development process:** bad engineering practices allow the accidental or malicious insertion of software vulnerabilities, clearly attesting that the software development process is immature from a security point of view.
- **Insufficient integrity verification:** the voting software verifies its own integrity during its initialization process, but all of the information needed to subvert this verification is contained inside the voting machines, with different attack surfaces depending on the presence of a hardware security module. In the older voting machine models without this module, the problem of software authentication is reduced to itself, with no external source of trust. In this case, digital signature- based software self-verification (Janino, Balcão Filho, Montes Filho, Lima-Marques, & Dahab, 2009) is equivalent to trusting the authenticity of a document based only on the allegations of the author, who is free to impersonate anyone. It is also important to emphasize that an authentic signature attests only to the processing of the protected object at a point in time and space where the signing private key was also present. Even when the integrity verification mechanisms are not circumvented, digital signature techniques cannot attest that software is in fact correct or secure. Digitally signing vulnerable software also has the opposite effect of providing mathematical certainty that all of the voting machines have the same exploitable flaws. The version of the source code studied by the authors also had commented out a function call to perform integrity verification of a significant portion of the voting software, further illustrating the intrinsic limitations of the technique.

Detailed descriptions of the problems mentioned above are presented in the rest of this document, but it can be noted that many of the protection features implemented in the voting machine software aim to achieve *obfuscation* instead of *security*, not resistance to insider attacks or advanced persistent threats. Several of these problems are the result of architectural flaws or inappropriate design assumptions. Fixing the underlying causes will require more than *ad hoc* localized interventions in the source code. A complete review of the software development process is needed to establish good engineering practices and avoid the intentional or accidental insertion of new vulnerabilities by internal or external attackers. Since the Direct Recording Electronic (DRE) voting machines adopted in Brazil require software integrity to provide integrity of results, the problems discussed in this report achieve a critical status and require the introduction of software-independent auditability measures (Rivest, 2008). Only with periodic scientific evaluation, it is possible for the Brazilian voting system to satisfy minimal and plausible security and transparency requirements.

This document discusses only aspects of the voting machine software, omitting physical or hardware aspects of the equipment in order to respect the authors' fields of expertise. The information provided only pertains to a small – yet strategic – fraction of the software source code. It excludes other software components that constitute the complete voting system, because the rules of the event and time restrictions imposed on the investigators did not allow for a full evaluation. The content is entirely the responsibility of the authors and does not necessarily represent the position of University of Brasília or any other institutions where the authors have worked or will work in the future.

BACKGROUND

The Brazilian voting machine is a classical Direct Recording Electronic (DRE) device without a Voter-Verified Paper Audit Trail (VVPAT). It consists of an election officer terminal used to authenticate electors by their registration number or fingerprint and a voter terminal where votes are cast. Both terminals are connected by a cable, as shown in Figure 1. The cable provides access to the elector data stored in the voter terminal. In general terms, an election using the voting machine follows the preparation steps below:

1. Development of the software components and distribution of memory cards containing the voting software across the country.
2. Installation of the software stored in the memory cards on the voting machines.
3. Distribution of the machines to the corresponding polling places.

Figure 1. Brazilian voting machine and its two terminals.
The election officer terminal is on the left and the voter terminal is on the right.

On the election day, a well-defined procedure is executed at each polling place:

1. Printing of the zero tape, an official public document which supposedly attests that no votes were computed for any candidates before the start of the elections.
2. Opening of the voting session by the election officials.
3. Granting of access for electors to cast their votes in the voting machines.
4. Closing of the voting session by the election officials.
5. Printing of the Partial Summation (PS) by each voting machine, containing per-machine totals for each candidate.
6. Recording of authenticated public products of the election by each voting machine. They consist of a digital version of the PS, a chronological record of events registered by the

machine (LOG) and the Digital Record of the Vote (DRV), an electronic shuffled list of the actual votes.

7. Authorized breaking of the protecting seal by the election officials and retrieval of the Memory of Results (MR), an orange USB drive containing the public products of the election.
8. Transmission through a private network of the public products of the election to the centralized tabulation system. This is performed by the election officials at the polling places using computers provided by the SEC. The digital partial summations are made available on the Internet afterwards.

The role of the central tabulator is to combine all the partial summations to obtain and declare the overall result of the elections.

Organization

The document is structured as follows. The next section briefly describes the format and the results obtained in the Public Security Tests. Afterwards, we detail the progression of vulnerabilities which provided a method to defeat the sole mechanism implemented in the voting machines to protect ballot secrecy. Multiple alternatives for correcting the vulnerabilities are described and realistic scenarios are discussed where voter privacy is threatened if the vulnerabilities are not fixed. The following section presents another collection of flaws detected in the voting software and its development process. The final section concludes the document with perspectives on how to improve transparency and auditability of the Brazilian electronic voting system.

PUBLIC SECURITY TESTS

The 2nd Public Security Tests of the Electronic Voting System organized by the Superior Electoral Court (SEC) were held in March 2012. The organization involved a Disciplinary Committee, responsible for creating and enforcing the competition rules, and an Evaluation Committee, responsible for evaluating the performance of each competing team. Formally, the Public Security Tests began with the publication of a call for participation and team registration. According to the official announcement (SEC, 2012a), only the teams approved by the SEC would have the opportunity to participate in the trials. The major difference between the second and first iterations of the trials was access to the source code of the voting software. The first iteration of the event was held in 2009 and consisted exclusively of "black box" testing.

Format

The 9 approved teams were composed of 24 Brazilian professionals from industry, universities and government institutions. The investigators participated in two stages spanning 3 days with 10 hours a day of activities: (i) a preparation phase, March 6–8, when the teams could study the voting software source code and ask technical questions to formulate hypotheses and testing plans to evaluate the quality of security features implemented in the voting machine; (ii) a testing phase, between March 20–22, when teams could no longer study the source code but could exercise their methodologies to validate hypotheses and obtain results and conclusions.

Concrete activities of the 2nd Public Security Tests started on March 6, with an opening talk (Azevedo, 2012) where the format and rules of the event were presented, together with an overview of the voting procedures and security measures implemented in the voting machine. The goal of the opening talk was to level the amount of information available to the participants. The team composed by the authors,

identified as "Group 1", attended the opening talk to familiarize themselves with technical aspects of the system and to detect promising points of attack.

During the 12-day period between the two phases, teams were required to submit testing plans formulated from the information collected during the preparation phase. Only testing plants approved by the Disciplinary Committee of the event (appointed by the SEC) could be put into practice in the following phase. The restriction on source code access during the testing phase was waived on the second day of the testing phase. The authors did not take advantage of this possibility.

Objectives

The call for participation explicitly divided the objectives of the trials into two distinct classes, directly translated from the official announcement (SEC, 2012a):

- **Failure:** event when a system violates its specification after entering an inconsistent state of execution caused by a fault or imperfection in the software or hardware components, and improper functioning does not have any interference on the integrity or anonymity of the votes.
- **Fraud:** intentional act of modifying information or causing damage with impact on the integrity or anonymity of the votes, preferably without leaving apparent traces.

The first class comprises *denial of service* attacks, where an attacker aims only to make the voting equipment unavailable to the electors. The second class captures attempts at electoral fraud.

Our team formulated and submitted two testing plans, titled "*Untraceable attempt at compromising ballot secrecy*" (Aranha, Karam, Miranda, & Scarel, 2012a) and "*Untraceable attempt at corrupting election results*" (Aranha, Karam, Miranda, & Scarel, 2012b), both clearly directed to cause fraud in a simulated election using official procedures. Due to time restrictions, only the first testing plan was put into practice.

Methodology

The method proposed by the testing plan required the team to split into two parts, here identified by A and B, who alternated their presence in the testing room to avoid any kind of internal communication. The experiments followed the procedures below:

1. Generation by the SEC of a secret list of fictional votes for city councilor and mayor.
2. Receipt of the secret list of votes by part A of the team.
3. Software installation of the voting machine using an official memory card and printing of the zero tape.
4. Casting of votes in the voting machine by part A of the team, following the list order and under supervision of SEC officials.
5. Breaking of the protecting seal and delivery of the Media of Results (MR) to part B of the team.
6. Execution of a customized program to analyze the Digital Record of the Vote (DRV) stored in the MR and to produce a list of votes in an order supposedly corresponding to the votes cast on the voting machine.
7. Comparison of the list of votes kept secret from part B and the list of votes produced by the customized program.

The success criterion for the attack is naturally the correspondence between the two lists. Observe that, inside the testing room, part B of the team had to break a seal and retrieve the MR to complete the simulation, since this was the only way to obtain the DRV matching the simulated election. In real elections, the DRV is public by law (Presidency of Brazil, 2003). Part A also needed physical access to the voting machine, but only to cast the prescribed votes, according to the protocol described above.

Results

As stated in the report jointly written by the authors and the SEC (Aranha, Karam, Miranda, & Scarel, 2012a), the ballot secrecy attack method obtained absolute success in recovering the votes in the order they were cast during simulated elections with 10, 16, 21 and 475 electors (20, 32, 42 and 950 votes, respectively). The latter reproduced the proof-of-concept results with a realistic amount of data, a requirement made by the SEC to match the 82% participation rate from the previous election in the universe of 580 fictional electors composing the training set of the event. Voting in Brazil is mandatory, thus the high participation rate. Since the attack method only consisted of analyzing public products of an election, no modification in any component of the voting machine, or invasion of its security perimeter was needed. For this reason, the method is essentially untraceable.

Storing the votes in an order different than the order they were cast is a critical procedure for protecting ballot secrecy. It is clear that the authors' methodology defeated the sole security mechanism employed by the voting machine to protect ballot secrecy. It was not possible, however, to recover the ordered list of elector identities from the public products of an election. This information must be obtained externally in order to relate the ordered votes with the ordered identities, making possible an exact correspondence between each elector and his or her vote. To the extent the authors could investigate, public products only store the registration number of missing electors in lexicographic order. Later, we describe how recovering the ordered votes allow electoral fraud in realistic scenarios.

There was not sufficient time to execute the second testing plan, which aimed to evaluate the security measures that protect the integrity of results. Priority was given to the first testing plan because of its simplicity and almost complete independence from any significant collaboration with the SEC. Attacking the integrity of results during the trials would require active collaboration from the electoral authority to at least attest to the authenticity of the corrupted results with the existing detection measures.

Scoring

Scoring criteria were devised by the SEC to quantitatively compare the performance of the teams (SEC, 2012b) using the formula:

$$N = \frac{1}{\Delta t} \cdot \frac{1}{p^2} \cdot A \cdot E,$$

where Δt ranged from 1 to 15 depending on the time in minutes until the attack presented the first relevant results, p was the number of intervention points required for the attack to be successful, value A was 1 or 10 depending on the attack type (failure or fraud, as discussed previously), and value E ranged from 1 to 20 depending on the geographical extent of the attack (from polling place to nationwide). The final score would be doubled if the investigators provided a suitable solution for correcting the vulnerabilities found.

Without detailed justification and even with the absolute success during the execution of the testing plan, the authors received the negligible score of 0,0313 on a 0–400 scale (Lima-Marques, Montes Filho, Imamura, Barbar, & Cardoso, 2012). The Evaluation Committee of the event (also appointed by the SEC) considered that our team took 176 minutes to successfully attack the system ($\Delta t = 4$), required 4 intervention points ($p = 4$), aimed at only causing a failure ($A = 1$) and impacted single voting machines or polling places ($E = 1$).

The penalties applied to the team score were questionable at best. It was not clear, for example, why penalties caused by intervention points required at the testing environment were applied even if they would not be needed during a real instantiation of the attack. The Evaluation Committee cited the following intervention points: physical access to the voting machine, protection seal, and memory cards and access to the source code. It would be impossible to simulate any election without physical access to the voting machine and it would be impossible to analyze the public products of a simulated election without breaking the seal to retrieve the Media of Results. The attack did not require access to the voting machine beyond what is allowed to electors during the voting process or mandated to election officials at the end of the voting session. Political parties receive the contents of the Media of Results without physical access to the voting machine. Is is also incoherent to penalize the team for reviewing the voting software source code, when the objective of the event was to evaluate the quality of security features implemented in that source code. The team still does not understand why their methodology was considered to be an attempt to cause failure instead of a fraud attempt on a simulated election, since no apparent failure was perceived in the voting equipment during the whole trials. Despite the scoring issues, the team won the competition after providing the most significant contribution to improve the security of the electronic voting system. There are two possible hypotheses for the negligible team score: either the Evaluation Committee did not understand the severity of the vulnerability exploited or this was a deliberate attempt to mischaracterize and quantitatively minimize the results. Both hypotheses are equally worrisome.

Improvement

During their participation, the authors collected several recommendations to improve the format of the event. Even if these recommendations are mostly of interest only to the event organizers, they can provide some insight on how the event was coordinated and how to set up rules for similar events in other countries:

- **Minimize intervention from the event staff:** the necessity to monitor the investigators during the execution of their testing plans is understandable, but the lack of privacy and constant intervention disrupted the efficiency of the team.
- **Minimize bureaucracy:** again, the necessity of keeping track of all the procedures executed by the investigators is perfectly justifiable, but satisfying bureaucratic requirements consumed an amount of time which could be dedicated to the execution of additional testing plans.
- **Minimize the time restriction:** 30 hours are absolutely insufficient to analyze a significant portion of the voting machine source code, which has in total a few million lines. Mission-critical software should be considered security software in its entirety, since a vulnerability in non-critical code can trigger a vulnerability in critical code.
- **Increase the source code availability:** a sealed room with only 4 computers was specifically dedicated for studying the source code. Since many teams had to share these 4 computers, the lack of capacity severely reduced the amount of exposure of the source code. In particular, our team only obtained access to the source code at 11 AM of the second day of the preparation phase, since another team obtained exclusive access to the sealed room on the first day. In total, our team spent only 5 hours of the preparation phase studying critical portions of the source code. On a positive note, the current availability of simple text processing utilities (`grep`, `vi`, `cat`, etc.) was paramount for the efficient detection of which code sections presented higher interest.

- **Enlarge the testing scope:** the event focused exclusively on the security mechanisms implemented in the voting machine, not pertaining to the central tabulator. The SEC provided the justification that any entity can perform a parallel tabulation of the results after all the partial numbers are published on the Internet. This way, any attack directed at the tabulator would only delay the publication of the official results, not change them. However, in our opinion, successful attacks directed to the centralized tabulation could create ambiguity or corruption of the official results. These can be detected and neutralized afterwards, but only when the respective guarantees, that the correct results obtained by each voting machine correspond to the ones published in the Internet, are available to any potentially damaged candidates. A successful attack of this type would still call into question the reputation and capacity of the electoral authority in executing the elections or even the validity of the election outcome.
- **Improve the scoring criteria:** the formula above for evaluating the performance of the teams was ill-conceived and had too much focus on applying penalties. The official report written by the Evaluation Committee did not justify their decisions and only listed the intervention points and final scores.
- **Change the nature of the event:** the competition format creates disincentives for information sharing among the teams and emphasizes cost-benefit metrics. Teams are led to prioritize attacks that would be fast to execute and demonstrate within the restrictions of the event, rather than those that might pose the most danger to real elections in practice. These characteristics clearly model a portion of potential attackers, but only a careful collaborative evaluation of security mechanisms allow the modeling of well-informed attackers with considerable resources to represent more dangerous threats.

The complete and careful evaluation of the voting machine software requires enormous amounts of effort and time. Without the possibility of extensive unrestricted testing, following a sound scientific methodology, it cannot be said that the current format of the event significantly improves the security of the voting system. It only allows the detection of easily exploitable vulnerabilities which allow simple attacks with limited effects.

VULNERABILITIES

In this section, we describe the sequence of vulnerabilities which allowed the team of authors to recover the list of ordered votes in several consecutive simulated elections, one of them using a realistic number of electors.

Digital Record of the Vote (DRV)

Following the introduction by electoral law of the current DRE voting machines in 1997 (Presidency of Brazil, 1997), voter-verified paper audit trails (VVPATs) were adopted in Brazilian elections for the first time in 2002 (Presidency of Brazil, 2002). They aimed to distribute among all electors the possibility of independent verification of their individual votes. Paper audit trails consist of a voter-verified physical record of the votes that can be stored for later recount without allowing electors to prove their choices to any interested parties. Without independent verification of results, trust has to be put on the limited software auditing measures exercised by the political parties before the election and on the good faith of the technicians responsible for the voting system (van de Graaf & Custódio, 2002, page 23). After allegations by the election authority that the additional printers increased costs significantly and created many operational problems, VVPATs were discontinued in 2003 (Presidency of Brazil, 2003). In their place, a purely digital substitute was adopted. Today, the only record of the votes is stored as a data structure called the DRV in the voting machine's electronic memory.

The DRV is a table separated into sections, where each section is devoted to a different race. This table shuffles the votes cast by the electors during storage to disassociate the order of the votes and the order of electors. It was introduced as a replacement to VVPATs to supposedly permit independent verification of election results. For this reason, it is a public document made available to the political parties after the elections. However, while paper audit trails in fact allow independent verification of the votes computed electronically, the DRV is produced by the same software component which tallies the votes and produces per-machine partial results. This way, any successful attack against the tallying process can also compromise the integrity of the DRV.

Hence, the DRV does not serve any practical purpose besides compromising ballot secrecy if it is not designed or implemented securely. Figure 2 presents a fictitious DRV for an election with 3 races and 7 electors of which only 3 participated. The first elector choses candidate number 13 for Governor, 31 for Senator and casts a BLANK vote for President. The second elector chooses 71 for Governor, casts a NULL vote for Senator by inputting an invalid number and chooses 37 for President. The third and last elector also chooses 71 for Governor, casts a BLANK vote for Senator and chooses 37 for President. Observe that the final version of the file apparently does not allow recovery of any correspondence between electors and their votes, and that unused positions are conserved by the shuffling process.

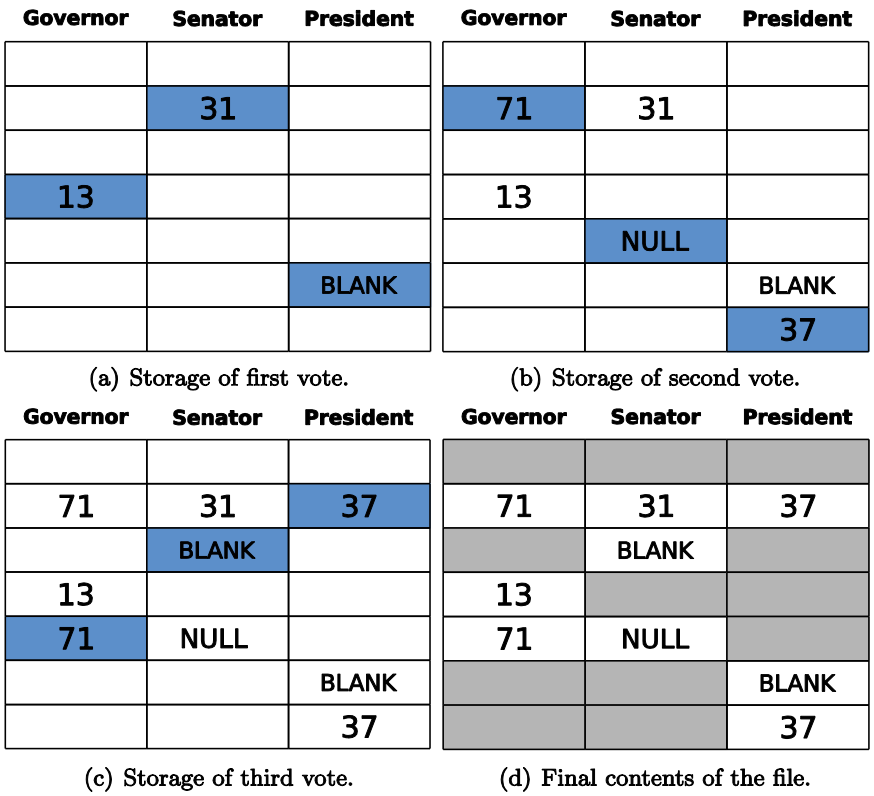

Figure 2. Example of shuffled storage of votes in the DRV.

Hypothesis

The vote shuffling mechanism was presented as a security feature in the opening talk (Azevedo, 2012) and immediately raised suspicion among our team. The reason for this was the clear observation that the vote shuffling should reach cryptographic strength, and only someone with proper training in computer security would recognize that this is as important for ballot secrecy as software integrity is for reliable tallying. Still during the opening talk, the team raised the hypothesis that the DRV was not

designed and implemented securely. With only a few recursive searches for well-known insecure functions for random number generation in the first hour of source code studying, the hypothesis was considerably strengthened. It only remained to determine which data was needed to revert the shuffling and recover the votes in the order they were cast.

Design and Implementation

The shuffling mechanism was designed and implemented with a progression of errors which culminated in allowing its reversal. The implementation uses a pseudo-random number generator, a computational procedure which produces a sequence of numbers apparently random, but that can be uniquely determined from a small parameter called the *seed* which must be chosen in a truly random fashion. When the sequence of numbers should be protected from independent derivation by an attacker, the seed must not only be truly random but also be kept secret. In the following, we present the progression of software vulnerabilities that forced the pseudo-random number generator to work outside of its operation limits, not fully reaching its security properties:

- **Inadequate choice of pseudo-random number generator:** the standard generator included in the C programming language and implemented through functions `rand()/srand()` was chosen. This generator has an extremely short period and accepts seeds with only 32 bits. Thus, it does not reach cryptographic strength (Wheeler, 2003). Just this choice of generator already allows a probabilistic attack method.
- **Inadequate choice of seed:** the seed was chosen at the initialization of the voting software as a time measurement with precision of seconds in the UTC time zone and implemented through the function `time()`. This choice of seed is obviously not truly random. The system must be initialized on election day between 7 and 8 AM and this information alone reduces the exhaustive search space to just 3600 values.
- **Public seed:** the seed was not only deterministic but also made public in the LOG of events and in the zero tape, both official documents. The former becomes public to the political parties after the election, while the latter becomes public right after its printing, when it receives handwritten signatures by election officials and inspectors from the political parties. Given the right time that the zero tape was printed, it is trivial to recover the ordered votes efficiently and exactly, without any error probability or need for an exhaustive search. The digital signature mechanism on the LOG file and the handwritten signatures on the zero tape further guarantee that the documents are authentic and the timestamp contained in them is indeed the correct seed.

Algorithms 1 and 2 present simplified versions of how the pseudo-random number generator was initialized and how votes were stored in the DRV, respectively. Figure 3 presents a copy of a real zero tape found on the Internet, with the seed (which should be random and secret) highlighted. Let n be the number of electors who voted in an election with m total electors. The way the DRV conserves the empty positions allows one to try different values for the seed and obtain the correct one when $n < m$. This test is possible by comparing the empty positions in the DRV with the empty positions generated by storing votes of n electors with the potential seed being tested.

Algorithm 1 DRV Initialization.

Input: Table T representing the DRV, total of m electors.
Output: Table T initialized and pseudo-random number generator seeded with a timestamp.

```
1: srand(time(NULL));
2: for i ← 0 to m do
3:     T[i] ← EMPTY
4: end for
```

Algorithm 2 Storage of a vote in the DRV.

Input: Table T representing the DRV, i-th vote V, with $0 \leq i < n$.
Output: Table T updated with vote V stored.

```
1: j ← rand() mod m
2: if T[j] ≠ EMPTY then
3:     {Collision found!}
4:     Increment or decrement j until a new free position is found
5: end if
6: T[j] ← V
```


Inst. Federal de Educação Ciência
e Tecnologia do Rio Grande do Sul
Campus Bento Gonçalves

Zerésima

Eleição do IFRS
(28/06/2011)

Município 88888
Bento Gonçalves

Zona Eleitoral 0008
Seção Eleitoral 0021

Eleitores aptos 0083

Código identificação UE 01105161
Data 28/06/2011
Hora 08:32:08

RESUMO DA CORRESPONDÊNCIA
588.653

Figure 3. Document showing the seed for shuffling votes during storage.

Attacks

The progression of vulnerabilities presented in the last section allows the formulation of two attack methodologies:

- **Direct attack:** given the seed, recovered from the LOG file or zero tape corresponding to a polling place, it is possible to simulate the shuffled storage of *n* votes and detect in which position of the public DRV each vote was stored. This makes possible the recovery of all votes in order, only from documents specified by the current system as essential for making the electoral process auditable.

- **Indirect attack:** given the votes stored out of order, it is possible to perform an exhaustive search in the seed space and discover the correct seed by comparing empty positions. With the correct seed detected, the direct attack can be executed.

Both attacks above are essentially untraceable, since they do not involve modification of any software or hardware component of the voting machine and do not require invasion of its physical perimeter. Reading public products of an election never leaves traces, since it is not possible to differentiate between inspection for auditing purposes and attacks on ballot secrecy. The attacks are also deterministic, exact and reproducible with no error probability. It becomes clear that the sole mechanism used by the voting machine software to protect ballot secrecy was defeated. This is aggravated by the fact that secret ballots are a constitutional requirement in Brazil (Presidency of Brazil, 1965). Algorithm 3 presents the direct attack described above. After the trials, the team obtained the information that the public LOG of events produced by the voting machine also stores the timestamp of when each vote is cast (SEC, 2008). When the time information is associated with the list of ordered votes, it is also possible to recover a specific vote cast in a specific time instant.

Algorithm 3 Recovery of ordered votes from the DRV.

Input: Table T representing the DRV, public seed s, number n of electors who voted among m total electors.
Output: List of ordered votes.

```
1: srand(s);
2: for i ← 0 to n do
3:    j ← rand() mod m
4:    if T[j] = MARK then
5:       {Collision found!}
6:       Increment or decrement j until T[j] ≠ MARK
7:    end if
8:    Print vote stored in T[j]
9:    T[j] ← MARK
10: end for
```

Consequences

Now suppose an attacker capable of coercing *k* electors and monitoring their behavior on election day. Voter coercion is historically so common in Brazil that it even has its own name in Portuguese: *voto de cabresto*. Recovering the list of ordered votes allows this attacker to obtain *mathematical certainty* in different types of electoral fraud violating ballot secrecy:

- Inserting the coerced electors into the *k* first positions of the voting queue. This does not seem hard to achieve if the attacker funds transportation for electors and arrives early at the polling places.
- Using a marker vote to indicate the beginning of the block of *k* coerced electors in the voting queue. If arriving early to the polling place is an issue, the attacker can instruct one elector to vote in a previously determined way (nulling his/her vote with a prescribed invalid number, for example), after which the sequence of coerced votes begins.
- Recording the identities and position of all electors in the voting queue or the time they cast their votes. This allows an attacker to break secrecy for all *n* electors, even those not coerced

by the attacker. Observe that this information can be obtained by collaboration with election officers or inspectors from the political parties.

The time a specific vote was cast determines the position in the voting order that a certain elector cast his/her vote. Examining the corresponding position in the ordered list of votes recovered from the DRV directly reveals the choices made by that elector. This directed attack, besides violating a constitutional requirement, can cause significant issues for public personalities (politicians, entrepreneurs, ministers). Note that the place and time they vote is frequently reported by the press on election day. For example, the time and place the then president of the SEC voted in the last elections was reported by the Court's internal press office (News Agency of the SEC, 2010).

Mitigations

Correcting the progression of vulnerabilities starts with strengthening the pseudo-random number generator which determines the positions votes are stored in the DRV. This improvement can be implemented from the components already available in the voting machine. A secure way to perform this correction is replacing the pseudo-random number generator currently used with a cryptographic pseudo-random number generator. Examples of such generators are documented in standards (National Institute of Standards and Technology, 1998) and implementations can be found in general-purpose cryptographic libraries (OpenSSL, 2012).

Proper unpredictable seeds also need to be provided for the improved pseudo-random number generator. This real randomness criterion can be satisfied by using a hardware generator based on a well-studied physical effect. According to the specification of the 2009 voting machines (SEC, 2009), a generator with these features is already available in the hardware security module inside the equipment. The AMD Geode processor mentioned in the specification also has a truly random number generator (AMD, 2007) accessible through the file `/dev/hw_random`. For previous models, engineering trade-offs must be made. A possible solution is obtaining the seed through a blocking read from the file `/dev/random` which provides entropy of cryptographic quality from nondeterministic operating system events. This approach has problems involving the predictability of the voting system initialization, which may not provide sufficient entropy for a truly random seed, and the lack of entropy impairing the equipment functionality. The last recommended solution is to relax the cryptographic strength and obtain the seed through a non-blocking read from the file `/dev/urandom`. In this case, cryptographic strength may be lost, but the quality of the shuffling should still be better than the current construction.

It is important to test all of the above suggestions and determine if they satisfy minimal security requirements established for the shuffling mechanism. The authors cannot be held responsible in case the suggested solutions do not completely remedy the shuffled storage of votes.

FLAWS

Studying the source code of the voting software revealed not only the vulnerabilities in the design and implementation of the security mechanism to protect ballot secrecy, as discussed in the previous chapter, but also several flaws in critical software components. Each flaw presented here is a potential vulnerability which allows an internal or external agent to formulate an attack methodology. The presence of flaws in critical software components attests to the presence of inherent flaws in the software development process.

In the software

In the following, several flaws found in the software are described, some of them already pointed in the 2002 report prepared by the Brazilian Computer Society (BSC), or previously discussed in the academic analysis of the voting software used in U.S. Elections (Calandrino, Feldman, Halderman,

Wagner, Yu, & Zeller, 2007). Diebold, Inc. manufactured the hardware for the Brazilian and most of U.S. voting machines, the software for the U.S. equipment and the voting software for initial versions of the Brazilian model. Currently, the SEC is responsible for producing all software running in the Brazilian voting machines.

Inadequate protection of ballot secrecy

The Digital Record of the Vote (DRV), introduced by a legal mandate in 2003 and described in the previous section does not provide any real independent verification of results because it is generated by the same software component which counts votes and produces the Partial Summation (PS). For this reason, the possibility of compromising the PS directly implies the possibility of compromising the DRV. This means that the DRV is just redundant information as fragile as what it tries to protect. Since the DRV does not have any practical value, it serves only as a source of attacks against ballot secrecy if the shuffled storage of votes is not designed and implemented securely. Even if the DRV were implemented securely, the voting machine design would not completely eliminate the possibility of associating the elector identities and their votes through malicious software (van de Graaf & Custódio, 2012), since both terminals responsible for collecting this information are electronically connected. The required information exists in the internal state of the voting machine at some point and can be captured by malicious software.

The DRV already has 9 years of history and the question of whether the vulnerability discussed in the previous chapter was also present in the voting software used in 4 past elections (2004, 2006, 2008 and 2010) poses an interesting possibility. While the authors do not currently have any intention of investigating this issue, there are only three possibilities: (i) the shuffling mechanism used in past elections was more vulnerable than the one examined by the team; (ii) the shuffling mechanism used in past elections was as vulnerable as the one examined by the team; (iii) the shuffling mechanism used in past elections was less vulnerable than the one examined by the team. The first two hypotheses indicate that there was inadequate protection to ballot secrecy in 4 past elections, leaving this security property open to attack by internal or external agents with some knowledge of the mechanism. The third hypothesis indicates that the quality of the voting software decays with time, pointing to fundamental problems in how the software is developed. The three possibilities are then equally worrisome, especially when it is considered that secret ballots are required by the Brazilian constitution and that the country has been a fertile field for electoral fraud based on voter coercion for most of its history.

Recommendation. *Eliminate the DRV and replace it by a mechanism which allows truly independent verification of results such as a voter-verified paper record. If the presence of the DRV is still a requirement, we recommend at least that the empty positions be eliminated from the final version of the file. This makes an exhaustive search in the seed space much harder. However, if the shuffled storage of votes is still vulnerable, this compression will not adequately resist to insider or well-informed attackers.*

Inadequate entropy source

Entropy has a critical aspect to several cryptographic operations which require random data, such as generation of ephemeral keys or seeding of pseudo-random number generators. In many cases, it is possible to completely circumvent the cryptographic primitive by only attacking its entropy source. Obtaining sufficient entropy in devices with limited interactivity through software-only resources is practically impossible. As discussed in the previous chapter, the voting machine software used only a time measurement with resolution of seconds as entropy source, even when better sources where available in hardware.

Collecting predictable information as an inadequate entropy source is not an unknown or new vulnerability in either voting systems or commercial software. The voting machine used in the U.S. employed equally insecure techniques (Calandrino et al., 2007, Issue 5.2.12), obtaining information from the screen contents and a time measurement with resolution of milliseconds. In 1995, PhD students from University of California, Berkeley, discovered without access to source code that version 1.1 of the Netscape Navigator had the same exact vulnerability (Goldberg & Wagner, 1996). In particular, the seed was obtained using the same function call on line 1 of Algorithm 1.

Recommendation. *Adopt the suggestions presented in the section titled "Mitigations".*

Insufficient verification of software integrity

The Brazilian voting machine has a mechanism for integrity verification of its software as a mean of detecting if the software was maliciously replaced during its installation or execution. This mechanism varies greatly depending on the presence of a customized hardware security module. Because of this, our analysis will be split into two scenarios.

Voting machines not equipped with a hardware module. Software verification is reduced to itself, and thus vulnerable to deactivation if an attacker can access the portions of the software responsible for executing the verification. To reduce this risk, it is common to implement a preliminary integrity check at BIOS level (Basic Input/Output System) to guarantee that the software executed next in authentic. However, this technique only reduces the integrity of the software to the integrity of the BIOS firmware. The problem of verifying the BIOS firmware is reduced to itself, without any external source of trust.

Voting machines equipped with a hardware module. BIOS firmware is further checked by the hardware module. In this scenario, the software integrity verification problem is reduced to the authenticity of the source of trust stored inside the hardware module. This can be a self-contained certificate chain to validate digital signatures applied to the other software components. Defeating a software verification mechanism with these characteristics requires collaboration of an insider capable of deactivating the security module, or replacing the certificate chain and computing new signatures for the malicious software with the corresponding private keys. However, according to specifications of the security module in the 2009 voting machines, the hash value of the BIOS firmware needs to be programmed into the hardware module (SEC, 2009). This means that the BIOS transmits its own hash value to be verified by the hardware module, instead of requiring that the module actively verify the BIOS firmware. Hence, a malicious BIOS can impersonate the authentic BIOS by transmitting the correct hash values and deactivate the integrity verification of the software components executed afterwards.

Furthermore, the authors observed that a critical line of code in the application manager responsible for verifying the integrity of dynamic shared libraries was deactivated with a comment, confirming that even if a chain of trust is correctly established, software integrity verification is still susceptible to sabotage or programming errors.

The BCS Report already presented an explicitly skeptical position regarding the possibility of software self- verification through cryptographic techniques (van de Graaf & Custódio, 2002, page 24). Additionally, guaranteeing that the voting software indeed was produced by the SEC does not make it secure, but rather only confirms its origin, even when the integrity verification mechanism is not circumvented and works correctly.

The software integrity verification problem is endemic in voting systems and is particularly hard to solve in practice. The same limitation in the integrity controls was observed in the voting machines used in the U.S. (Calandrino et al., 2007, Issues 4.1.5 and 4.1.6). For this reason, it is recommended to install means for software-independent auditability of results, such as by reintroducing a voter-verified paper record and adequate post-election audit procedures.

Recommendation. *Perform the verification of the BIOS contents by the hardware security module in an active manner. This recommendation was also suggested by Group 6 participating in the trials (Santos, Correia, Barbosa, & Hachem, 2012). More generally, we recommend transferring the pressure on verifying software integrity to software-independent verification of the results produced by it.*

Sharing of cryptographic keys

Every voting machine in operation uses the same cryptographic key to encrypt the protected partitions of its memory cards. Leakage of this cryptographic key has the devastating impact of revealing to an attacker the entire content of the memory cards, including the voting software, the software integrity verification mechanism and the RSA private key used to digitally sign the public products of an election (SEC, 2010a). The latter is shared by all voting machines in the same state (SEC, 2010b), and its leakage allows an attacker to produce a forged file (LOG, DRV, PS) detected as authentic by the central tabulator. We can conclude that confidentiality of the private key and, consequently, integrity of the partial summations depend only on the confidentiality of a cryptographic key shared by half a million machines (Azevedo, 2012).

In an official position, the SEC argues that using multiple encryption keys to encrypt the same files can leak statistical characteristics of plain text (Rohr, 2012). Attacks of this nature are indeed studied in cryptographic literature, but do not represent any relevant threat in practice (Hong & Sarkar, 2005). It is clear that this risk is nowhere near the consequences of a compromise of the massively shared encryption key. If a proper mode of operation for encryption is used, this risk is trivially eliminated by randomizing the block cipher input when the plain text cannot be chosen by the attacker (Hong & Sarkar, 2005), as in the case discussed here.

Recommendation. *Assign a different cryptographic key to each voting machine, or at least to each memory card used to install software in a reduced set of voting machines. Key derivation functions are cryptographic tools designed to solve this exact problem. The hardware security module introduced in newer voting machines also has unused storage capacity for private keys (SEC, 2009).*

Presence of cryptographic keys in the source code

Sharing of cryptographic keys is aggravated by their clear presence in the source code of the voting software. This means that any internal agent with unrestricted access to the versioning repository where source code is kept immediately has access to the cryptographic key which protects the encrypted partitions of all memory cards. This also means that the encryption key is part of the operating system module responsible for mounting the encrypted partitions and making their contents available. Thus, it must be stored in the plain text portion of the memory cards. The encrypted objects are stored right beside the cryptographic keys which decrypt it, qualifying this mechanism as *obfuscation* instead of a *security* measure. Leaking the key becomes possible for anyone knowing or able to discover the position in which the key is stored by simply analyzing the plain text portions of the software.

Recommendation. *Store the encryption key in the hardware security module or preferably in a tamper- resistant device external to the voting machine environment.*

Inadequate use of encryption

The encryption algorithm used to protect the encrypted partitions of the memory cards is the Advanced Encryption Standard (NIST, 2001a) at the security level of 256 bits, a recommended choice for critical applications. The selected block cipher mode of operation is Cipher Block Chaining (CBC). The combination of algorithm and mode of operation is particularly good. However, the mode of operation uses not only the same encryption key for all voting machines but also the same initialization vector (the element responsible for randomizing the block cipher input and eliminating undesirable leakage of

statistical characteristics of the plain text). Choosing a new random initialization vector for each encryption operation is a requirement for this mode of operation (NIST, 2001b). Arguing that using the same encryption key for all voting machines to prevent statistical leakage (Rohr, 2012) loses any meaning when the way the mode of operation is used violates its specification.

Recommendation. *Select a new initialization vector for each encryption operation executed by the voting machine software, respecting the original specification of the chosen mode of operation.*

Inadequate choice of algorithms

Algorithms were not only badly chosen for pseudo-random number generator. The voting machine software also employed the SHA-1 hash function (NIST, 2002) for computing digital signatures and verifying software integrity. This specific hash function is not recommended for such applications since 2006, when it was discovered that it does not offer collision resistance. Rapid migration to secure hash functions was also recommended following that discovery (NIST, 2006). A sophisticated collision in this hash function like those demonstrated in (Stevens, Lenstra, & de Weger, 2012) and (Stevens, 2013) would allow an insider attacker to construct fake voting software capable of producing election results indistinguishable from the correct outcome.

Recommendation. *Employ a pseudo-random number generator of cryptographic quality and a collision- resistant cryptographic hash function, for example, from the SHA-2 family (NIST, 2002). If the length of hash values is crucial for human verification, it is possible to truncate the output of stronger hash functions.*

Repeated implementation of cryptographic primitives

The authors found several instances of repeated implementation of cryptographic algorithms in the code base. Apparently, every software component which employs cryptography in some way receives its own implementation of the involved algorithms, making the proper auditing of all the implementations much harder and significantly increasing the chance of error.

Recommendation. *Concentrate all implementations of cryptography in the same library of critical code to ease auditing of their correct functionality. Using a well-known general-purpose cryptographic library such as OpenSSL is also recommended.*

In the development process

The flaws discussed in the previous section are the product of a fragile software development process. From now on, we discuss flaws found or inferred by context in this development process. Many of the same problems were also detected in the development process used in the U.S. voting machines manufactured by Diebold (Calandrino et al., 2007, Section 4.3).

Complexity

Security is a result of simplicity, transparency and correct evaluation of trust assumptions and conditions. The millions of source code lines required to carry out simple elections in Brazil eliminates any reasonable possibility of a full and effective software audit review. It can be argued that a significant volume of this software is dedicated to the operating system and thus does not need a review. However, we verified that the SEC insert code sections into the operating system components. For example, the encryption key is directly inserted into the source code of the operating system module responsible for mounting encrypted partitions. It is also worrisome that insufficient compartmentalization and vulnerabilities in non-critical portions of software can create severe vulnerabilities in critical portions which affect security measures.

A volume of source code of this magnitude will, inevitably, have vulnerabilities which can be exploited. For this reason, the code base needs to be completely oriented around a small set of critical functionalities. The correct and secure functioning of the equipment should rely on this critical set. As a reference value, researchers who evaluated the Diebold voting software in a 60-day interval concluded that the thousands of lines of code dedicated only to the application layer had such complexity that it is not possible to make them secure (Calandrino et al., 2007, Issue 4.1.2).

Recommendation. *Reduce code volume by reuse, compartmentalization and refactoring techniques. Avoiding interventions in the external source code and isolating code portions of the operating system from the application layer can facilitate internal software audit reviews.*

Insufficient external software audit

Inspectors from political parties have the guaranteed right to examine the source code of the voting software, but for this they have to sign a Non-Disclosure Agreement (NDA) which prevents them from publicly disclosing any problem observed in the code. Consequently, inspectors cannot reveal the quality of the voting software or its security measures in detail, while malicious agents are free to attempt electoral fraud. Since inspection from independent investigators is extremely limited, during a period where the immense code base is constantly modified and under inadequate conditions, or, more recently, consisting only of a few days of work and under complete monitoring, in practice no effective auditing is done in the software components of the electronic voting system. This problem was also previously raised by the BCS report (van de Graaf & Custódio, 2002, page 23).

In DRE voting machines without voter-verified paper trails, integrity of results depends only on software integrity. The scenario discussed here looks perfect for untraceable electoral fraud.

Recommendation. *Provide auditing capabilities to any Brazilian citizen, specialist or not, without any legal impediment.*

No static analysis of source code

The vulnerable function family employed for the shuffled storage of votes is detected as potentially insecure by any tool for static analysis of source code. For example, the free tool Flawfinder (Wheeler, 2007), produces the following warning when it examines code containing the function call, such as our implementation of Algorithm 3:

This function is not sufficiently random for security-related functions such as key and nonce creation. Use a more secure technique for acquiring random values.

Recommendation. *Adopt industry-standard tools for static code analysis in order to minimize the impact of programming errors capable of creating severe vulnerabilities, respecting good practices for developing mission-critical software.*

Inappropriate attacker model

The security mechanisms in the Brazilian voting machine are designed to only resist attacks from external attackers and ignore the risk of insider threats. In particular, as it is made clear by the SEC's official position (Rohr, 2012), detection of potentially malicious behavior promoted by internal agents is performed by an auditing process also executed by internal agents. The sharing of encryption keys mentioned previously is a perfect example of this phenomenon, since there is enormous emphasis on esoteric statistical attacks mounted by external attackers while the risk of leakage by insiders is completely ignored. Storing this encryption key as plain text in the voting machine memory cards shows that security is not designed to resist well-informed attackers.

Recommendation. *Adopt security mechanisms resistant to external agents and, particularly, internal agents armed with detailed knowledge of such measures.*

No internal security exercises

In a meeting between the authors and the SEC members responsible for designing and producing the voting machines, right after the public audience of the Public Security Tests, we offered a technical talk to illustrate all the problems found in the software and the reasoning which let us detect and explore the vulnerability previously discussed. The offer was well received, because it would allow the interested parties to exactly understand "how the attacker mind works", in the words of the SEC members. There was no further concrete invitation for this, but our reading of this meeting indicates that there is no internal team responsible for periodically simulating an attacker and exercising potential attack methodologies.

Recommendation. *Establish, train and direct an internal team of simulated attackers, a recommended practice for mission-critical software (Calandrino et al., 2007). Design of security measures needs to be accompanied by simultaneous attempts at defeating them.*

No formal training

The flaws discussed in this section, found even in critical security mechanisms, demonstrate clearly that the SEC employees responsible for developing voting software do not receive sufficient training to implement secure software. The hypothesis raised by the authors, as early as the opening talk, that the vote shuffling mechanism was not designed and implemented securely due to lack of training confirms this observation. The absence of internal simulations to model plausible attackers due to the lack of understanding of how an attacker works also supports our claim, since any well-trained professional in computer security naturally alternates between the roles of security designer and attacker to test the quality of his or her own work.

Recommendation. *Provide proper training for the development team to consequently improve the quality of delivered software. It is not realistic to expect secure software as the result of a software development team with no formal training in computer security.*

Critical data made available to investigators

The machines dedicated to studying the source code in a sealed room during the Public Security Tests apparently came directly from the development team. The evidence for this is the availability to all investigators of critical information regarding usernames, passwords and internal network paths to the software versioning servers. An attacker equipped with this information and able to enter the SEC internal network can maliciously modify the source code and make the changes effective under the credentials of an innocent programmer.

Recommendation. *Sanitize equipment made available to external visitors in a way that critical information is not disclosed.*

Ignorance of relevant literature

As discussed previously, the vulnerabilities found in the vote shuffling mechanism have been well-known for at least 17 years (Goldberg & Wagner, 1996). Several flaws discussed in this report were already described by technical reports evaluating other voting systems (Calandrino et al., 2007), or even the one under discussion (van de Graaf & Custódio, 2002), and represent the opposite of recommended practices and formal specification of cryptographic techniques. Persistence of these issues in a code base with 16 years of history is unjustifiable and clearly shows that the SEC team responsible for the electronic

voting system does not adequately follow the relevant movements in the field of electronic voting or computer security in general.

Recommendation. *Explicitly dedicate part of the development team to study and distribute relevant advances of practical or academic interest in the area of computer security.*

False sense of security

The incessant repetition that the Brazilian voting machine is unconditionally secure and tamper-resistant, even if this constitutes a theoretical impossibility, disturbs the critical sense of the software development team and culminates in the suspension of their self-evaluation mechanisms. The software development process used in the voting machines apparently works under the effect of suspension of disbelief, installing a generalized false sense of security. This is not the ideal environment to develop security measures, especially when these need to satisfy mission critical requirements.

Recommendation. *Install a software development process able to stimulate mutual and critical verification of the work being done, with realistic evaluation parameters.*

CONCLUSIONS AND PERSPECTIVES

We presented a collection of software vulnerabilities in the Brazilian voting machines which allowed the efficient, exact and untraceable recovery of the ordered votes cast electronically. Associating this information with the ordered list of electors, obtained externally, allows a complete violation of ballot anonymity. The public chronological record of events kept by the voting machines also allows recovering a specific vote cast in a given instant of time. The consequences of these vulnerabilities were discussed under a realistic attacker model and mitigations were suggested. Several additional flaws in the software and its development process were detected and discussed with concrete recommendations for mitigation. In particular, it was demonstrated how to defeat the sole mechanism employed by the voting machine to protect ballot secrecy. The necessity of installing a continuous and scientifically sound evaluation of the system, performed by independent specialists from industry or academia, becomes evident and should contribute to the improvement of the security measures adopted by the voting equipment.

This collection of flaws and vulnerabilities provides material evidence for hypotheses already raised by the 2002 BCS Report on the voting system (van de Graaf & Custódio, 2002). In particular, we can conclude that there was no significant improvement in security in the last 10 years. Inadequate protection of ballot secrecy, the impossibility in practice of performing a full or minimally effective software review and the insufficient verification of software integrity are still worrisome. Since these three properties are critical to guarantee the anonymity and integrity of votes, the authors repeat the conclusions of the aforementioned report and defend the reintroduction of voter-verified paper audit trails to allow simple software-independent verification of results. Paper audit trails distribute the auditing procedure among all electors, who become responsible for verifying that their votes were correctly registered by the voting machine, as long as an audit is done afterwards to check that the electronic and manual vote counts are equivalent. This auditing process can be performed in a prescribed portion of the votes to reduce the impact on the availability of results. It is important to emphasize that printed votes are only a means for independent verification and should not leave the voting place to serve as proof for external parties, as mandated by the corresponding law (Presidency of Brazil, 2009). Voter-verified paper audit trails were scheduled to return in the 2014 elections, but unfortunately they were suspended by the Superior Court of Justice under questionable allegations of unconstitutionality.

A movement in this direction would follow the current trend in electronic voting systems. With field tests for a voter-verified paper record being executed by the Indian Election Commission, Brazil is now the only major democracy that relies exclusively on electronic voting systems without independent verification of results. We believe that, for this reason, and in light of the severe security problems

discussed in this report, the software used in the Brazilian voting system does not satisfy minimal and plausible security and transparency requirements.

ACKNOWLEDGEMENTS

We would like to thank colleague Prof. Pedro Rezende from University of Brasília, Prof. Jeroen van de Graaf from Federal University of Minas Gerais, Prof. Paulo S. L. M. Barreto from University of São Paulo, and Prof. Francisco Rodríguez-Henríquez from Centro de investigación y de Estudios Avanzados del Instituto Politécnico Nacional de México, for relevant discussions during the preparation of this work. The authors are especially grateful to Prof. J. Alex Halderman from University of Michigan for providing many useful comments on a preliminary version of this report.

REFERENCES

AMD (2007). Design without compromise. Retrieved March 23, 2012, from http://www.amd.com/us/Documents/33358e_lx_900_productb.pdf

Aranha, D. F., Karam, M. M., Miranda, A., & Scarel, F. B. (2012a). Testing Plan GP1T1 - Untraceable attempt at compromising ballot secrecy (in Portuguese). *Public Security Tests of the Electronic Voting System.* Retrieved March 23, 2012, from http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/G1PT1.pdf

Aranha, D. F., Karam, M. M., Miranda, A., & Scarel, F. B. (2012b). Testing Plan GP1T2 - Untraceable attempt at corrupting election results (in Portuguese). *Public Security Tests of the Electronic Voting System.* Retrieved March 23, 2012, from http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/G1PT2.pdf

Azevedo, R. (2012). Technical Security Aspects of the Electronic Voting System (in Portuguese). Retrieved March 23, 2012, from http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/ arquivos/material/Apresentacao_aspectos-tecnicos.pdf

Calandrino, J. A., Feldman, A. J., Halderman, J. A., Wagner, D., Yu, H., & Zeller, W. P. (2007). Source Code Review of the Diebold Voting System. *California Secretary of State "Top-to-Bottom" Voting Systems Review*. Retrieved March 23, 2012, from https://www.sos.ca.gov/voting-systems/oversight/top-to-bottom-review.htm

Goldberg, I. & Wagner, D. (1996) Randomness and the Netscape Browser. *Dr. Dobb's Journal*. Retrieved March 23, 2012, from http://www.cs.berkeley.edu/~daw/papers/ddj-netscape.html

Hong, J. & Sarkar, P. (2005) New Applications of Time Memory Data Tradeoffs. In B. K. Roy (Ed.) *11th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2005)* (pp. 353 – 372), Springer.

Janino, G. D., Balcão Filho, A., Montes Filho, A., Lima-Marques, M., & Dahab, R. (2009). Report from the Multidisplinary Committee appointed by the Superior Electoral Court (in Portuguese).

Lima-Marques, M., Montes Filho, A., Imamura, O. C., Barbar, J. S., & Cardoso, S. D. (2012). Evaluation of the Public Security Tests (in Portuguese). *Public Security Tests of the Electronic Voting System.* Retrieved March 23, 2012, from http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/RelatorioFinal.pdf

National Institute of Standards and Technology (1998). FIPS 186-1 – Digital Signature Standard (DSS). *Federal Information Processing Standards (FIPS) Publications.* Retrieved March 23, 2012, from http://csrc.nist.gov/publications/PubsFIPS.html

National Institute of Standards and Technology (2001a). FIPS 197 – Advanced Encryption Standard. *Federal Information Processing Standards (FIPS) Publications.* Retrieved March 23, 2012, from http://csrc.nist.gov/publications/PubsFIPS.html

National Institute of Standards and Technology (2001b). Recommendation for Block Cipher Modes of Operation. *Special Publications (800 Series).* Retrieved March 23, 2012, from http://csrc.nist.gov/publications/PubsSPs.html

National Institute of Standards and Technology (2002). FIPS 180-2 – Secure Hash Standard (SHS). *Federal Information Processing Standards (FIPS) Publications.* Retrieved March 23, 2012, from http://csrc.nist.gov/publications/PubsFIPS.html

National Institute of Standards and Technology (2006). NIST comments on Cryptanalytic Attacks on SHA-1. Retrieved March 23, 2012, from http://csrc.nist.gov/groups/ST/hash/statement.html

News Agency of the Superior Electoral Court (2010). President of the SEC votes in the capital city of Brazil (in Portuguese). *Superior Electoral Court News Website.* Retrieved March 23, 2012, from http://agencia.tse.jus.br/sadAdmAgencia/noticiaSearch.do?acao=get&id=1336461

Presidency of Brazil (1965). Law 4,737 of July 15, 1965 (in Portuguese). Retrieved March 23, 2012, from http://www.planalto.gov.br/ccivil_03/leis/l4737.htm

Presidency of Brazil (1997). Law 9,504 of September 30, 1997 (in Portuguese). Retrieved March 23, 2012, from http://www.planalto.gov.br/ccivil_03/leis/l9504.htm

Presidency of Brazil (2002). Law 10,408 of January 10, 2002 (in Portuguese). Retrieved March 23, 2012, from http://www.planalto.gov.br/ccivil_03/leis/2002/L10408.htm

Presidency of Brazil (2003). Law 10,740 of October 1, 2003 (in Portuguese). Retrieved March 23, 2012, from http://www.planalto.gov.br/ccivil_03/leis/2003/l10.740.htm

Presidency of Brazil (2009). Law 12,034 of September 29, 2009 (in Portuguese). Retrieved March 23, 2012, from http://www.planalto.gov.br/ccivil_03/_ato20072010/2009/lei/l12034.htm

Rivest, R. L. (2008). On the notion of "software independence" in voting systems. *Philosophical Transactions of The Royal Society A, 366* (1881), 3759 – 3767.

Rohr, Al. (2012). Flaw in voting machine faithfully reproduced error from 1995, says professor (in Portuguese). *G1 Digital Security Column*. Retrieved March 23, 2012, from http://g1.globo.com/platb/seguranca-digital/2012/05/28/falha-na-urna-brasileira-reproduzia-fielmente-erro-de-1995-diz-professor/

Santos, A. L. M., Correia, M. A. S., Barbosa, L. G. M., & Hachem, S. R. F. (2012). Testing Plan GP6T1 – Security test of the electronic voting system (in Portuguese). *Public Security Tests of the Electronic Voting System.* Retrieved March 23, 2012, from http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/G6PT1.pdf

Stevens, M., Lenstra, A., & de Weger, B. (2012). Chosen-Prefix Collisions for MD5 and Applications. *International Journal of Applied Cryptography, 2* (4), 322 – 359.

Stevens, M. (2013). New collision attacks on SHA-1 based on optimal joint local-collision analysis. In T. Johansson & P. Q. Nguyen (Eds.) *32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2013)* (pp. 245 – 261), Springer.

Superior Electoral Court (2008). File Specification of the Event Log of the 2008 Voting Machine (in Portuguese), Version 2. Retrieved March 23, 2012, from http://www.tse.gov.br/internet/eleicoes/arquivos/logs2008/EspecificacaoArquivoRegistroLogUrnasEletronicasEleicoes2008.pdf

Superior Electoral Court (2009). Acquitistion of 2009 Voting Machines – Basic Project (in Portuguese). Retrieved March 23, 2012, from http://www.tse.jus.br/transparencia/arquivos/tse-projeto-basico-audiencia-publica-2009

Superior Electoral Court (2010a). Elections – List of hash values (in Portuguese). Retrieved March 23, 2012, from http://www.tse.jus.br/arquivos/tse-urna-eletronica-modelo-2009-eleicoes-2010-turno-1-e-2-atualizado-em-22-09-2010-991ue09

Superior Electoral Court (2010b). OKEY System (in Portuguese). Retrieved March 23, 2012, from http://www.tse.jus.br/arquivos/tse-chaves-das-u.f.s-eleicoes-2010-turno-1-e-2-991okey

Superior Electoral Court (2012a). Call for participation no. 01/2012 (in Portuguese). *Public Security Tests of the Electronic Voting System.* Retrieved March 23, 2012, from http://www.justicaeleitoral.jus.br/arquivos/tse-2-edicao-dos-testes-de-seguranca-na-urna-eletronica

Superior Electoral Court (2012b). Scoring criteria established by document no. 05/2012 (in Portuguese). *Public Security Tests of the Electronic Voting System.* Retrieved March 23, 2012, from http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/TSE-edital5-2012-criterios-de-classificacao.pdf

The OpenSSL Project (2012). Retrieved March 23, 2012, from http://www.openssl.org

van de Graaf, J. & Custódio, R. F. (2002). Electoral Technology and the Voting Machine – Report of the Brazilian Computer Society (in Portuguese). Retrieved March 23, 2012, from http://www.sbc.org.br/index.php?option=com_jdownloads&Itemid=195&task=view.download&catid=77&cid=107

Wheeler, D. (2003). Secure Programming for Linux and Unix HOWTO. Retrieved March 23, 2012, from http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.html

Wheeler, D. (2007). Flawfinder. Retrieved March 23, 2012, from http://www.dwheeler.com/flawfinder/

ADDITIONAL READING

Aviv, A. A., Cerný, P., Clark, S., Cronin, E., Shah, G., Sherr, M., & Blaze, M. (2008). Security evaluations of ES&S voting machines and election management system. In D. L. Dill & T. Kohno (Eds.) *2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 2008)*, USENIX Association.

Bishop, M. & Wagner, D. (2007) Risks of e-voting. *Communications of the ACM 50* (11), p. 120.

Blaze, M., Cordero, A., Engle, S., Karlof, C., Sastry, N., Sherr, M., Stegers, T. & Yee, K.P.(2007). Source Code Review of the Sequoia Voting System. *California Secretary of State Top-to-Bottom" Voting Systems Review*. Retrieved March 23, 2012, from https://www.sos.ca.gov/voting-systems/oversight/top-to-bottom-review.htm

Chaum, D., Essex, A., Carback, R. T., Clark, J., Popoveniuc, S., Sherman, A. T., & Vora, P. (2008). Scantegrity: End-to-End Voter Verifiable Optical-Scan Voting. *IEEE Security & Privacy 6* (3), 40 – 46.

Chaum, D., Carback, R. T., Clark J., Essex A., Popoveniuc, S., Rivest, R. L., Ryan, P. Y. A., Shen, E., & Sherman, A. T. (2008). Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes", In D. L. Dill & T. Kohno (Eds.) *2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 2008),* USENIX Association.

Checkoway, S., Feldman, A. J., Kantor, B., Halderman, J. A., Felten, E. W., & Shacham, H. (2009). Can DREs Provide Long-Lasting Security? The Case of Return-Oriented Programming and the AVC Advantage. In D. Jefferson, J. L. Hall, & T. Moran (Eds.) *2009 USENIX/ACCURATE/IAVoSS Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE 2009),* USENIX Association.

Cunha, S. S., Marcacini, A. T. R., Cortiz, M. A., Fernandes, C. T., Stolfi, J., Rezende, P. A. D., Brunazo Filho, A., Moura, F. V., Carvalho, M. A. M., & Teixeira, M. C. (2010). Report on the Brazilian Electronic Voting System (in Portuguese). Retrieved March 23, 2012, from `http://www.brunazo.eng.br/voto-e/textos/RelatorioCMind.pdf`

Cunha, S. S., Marcacini, A. T. R., Cortiz, M. A., Fernandes, C. T., Stolfi, J., Rezende, P. A. D., Brunazo Filho, A., Moura, F. V., Carvalho, M. A. M., & Teixeira, M. C. (2010). Report on Overseeing Elections in Argentina (in Portuguese). Retrieved March 23, 2012, from `http://www.votoseguro.org/textos/relatoriocmind-arg2011.pdf`

Dill, D. L. & Castro, D. (2008). The U.S. Should Ban Paperless Electronic Voting Machines. *Communications of the ACM, 51* (10), 29 – 30.

Gonggrijp, R. & Hengeveld, W.-J. (2007). Studying the Nedap/Groenendaal ES3B Voting Computer: A Computer Security Perspective. In R. Martinez & D. Wagner (Eds.) *2007 USENIX/ACCURATE Electronic Voting Technology Workshop (USENIX EVT 2007)*, USENIX Association.

Kohno, T., Stubbefield, A., Rubin, A. D., & Wallach, D. S. (2004). Analysis of an Electronic Voting System. In *IEEE Symposium on Security and Privacy 2004 (S & P 2004)* (pp. 27 – 42), IEEE.

Halderman, J. A., Rescorla, E., Shacham, H., & Wagner, D. (2008) You Go to Elections with the Voting System You Have: Stop-Gap Mitigations for Deployed Voting Systems. In D. L. Dill & T. Kohno (Eds.) *2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 2008),* USENIX Association.

Heninger, N., Durumeric, Z., Wustrow, E., & Halderman, J. A. (2012). Mining your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. In T. Kohno (Ed.) *21st USENIX Security Symposium (USENIX Sec 2012),* USENIX Association.

National Institute of Standards and Technology (2009). Draft Voluntary Voting System Guidelines Version 1.1. Retrieved March 23, 2012, from `http://www.eac.gov/assets/1/AssetManager/VVSG_Version_1-1_Volume_1_-_20090527.pdf`

Neuman, P. G. (2004). The Problems and Potentials of Voting Systems. *Communications of the ACM*, 47 (10), 28 – 30.

Ryan, P. Y. A., Bismark, D., Heather, J., Schneider, S., & Xia Z. (2009). The Prêt à Voter Verifiable Election System. *IEEE Transactions on Information Forensics and Security 4* (4): 662–673.

Sturton, C., Jha, S., Seshia, S. A., & Wagner, D. (2009). On voting machine design for verification and testability. In E. Al-Shaer, S. Jha, & A. D. Keromytis (Eds.) *16th ACM Conference on Computer and Communications Security (CCS 2009)* (pp. 463 – 476), ACM.

Wolchok, S., Wustrow, E., Halderman, J. A., Prasad, H. K., Kankipati, A., Sakhamuri, S. K., Yagati, V., & Gonggrijp, R. (2010). In E. Al-Shaer, A. D. Keromytis, & V. Shmatikov (Eds.) *17th ACM Conference on Computer and Communications Security (CCS 2010)* (pp. 1 – 14), ACM.

KEY TERMS AND DEFINITIONS

Attacker: a malicious entity whose aim is to prevent users and systems from achieving a security goal. In the context of elections, the security goals mainly comprise ballot secrecy and integrity.

Authentication: the act of confirming the truth of a data or entity attribute. This might involve confirming the identity of a person or authorship of a software program.

Ballot Secrecy: the security requirement for any voting method in which voter choices in an election or referendum are anonymous. This property protects the elector and his or her choice against influence by intimidation or bribery.

Ballot Integrity: the security requirement of a voting system in which votes cannot be modified, forged, or deleted without detection. This property guarantees that the election outcome matches voter intent.

Digital Signature: a mathematical scheme for demonstrating the authenticity of a digital message or document, providing reason to a recipient that the message was created by a known sender and that the message was not altered in transit.

Digital Recording Electronic (DRE): a voting machine that collects votes by means of a ballot display provided with electronic components that can be activated by the elector, processes data by means of a computer program, and records voting data and ballot images in memory components.

Encryption: the reversible process of encoding information in such a way that unauthorized eavesdroppers cannot read it. In symmetric encryption schemes, this is performed with the help of a cryptographic key shared by the communicating parties.

Entropy: the randomness collected by an operating system or application for use in cryptography or other computational method that requires random data.

Independent Verification: the property of any voting system in which an independent, honest observer (voter or poll watcher) can determine whether a declared election outcome correctly represents the votes cast by electors.

Software Independence: the property of a voting system in which an (undetected) change or error in its software cannot cause a change or error in an election outcome.

Software Integrity: the assurance that software components can only be modified by authorized agents. This involves some form of integrity verification mechanism designed to detect unauthorized software manipulation.

Voter-verified paper audit trail (VVPAT): the method of providing feedback to electors using electronic voting systems without physical ballots, intended to allow electors to verify that their vote was cast correctly.

Voting Machine: the collection of electronic equipment used to define ballots, cast and count votes; report or display election results; and maintain and produce any required audit trail information.