Software vulnerabilities in the Brazilian voting machine

Diego F. Aranha, dfaranha@unb.br
Marcelo Monte Karam, karam@unb.br
André de Miranda, andremrnd@unb.br
Felipe Brant Scarel, fbscarel@unb.br

Department of Computer Science, Center for Informatics
University of Brasília

Short version

```
srand(time(NULL));
fprintf(public_document,
"%d\n", time(NULL));
```

Context

Brazilian Elections:

- Is majoritarian or proportional, depending on public office position
- Administered, executed and judged by the Supreme Electoral Court (SEC) presided by judge from the Supreme Court
- After rampant ballot paper fraud, became electronic in 1996
- Held in October with mandatory participation (140 million voters)

Brazilian Voting Machines:

- Claimed to be 100% secure (...but never really tested until March)
- Have hardware manufactured by Diebold (half a million)
- Have software developed by the SEC since 2006
- Adopted GNU/Linux in 2008 (after Windows CE and VirtuOS)
- Run software accessible to political parties under NDA
- Experimented with VVPATs in 2002

Context

Figure : Classic DRE voting machine: from left to right, election officer terminal and voter terminal.

Cronology of the Public Security Tests

1. Preparation phase (March 6-8, 2012):
   - Opening presentation
   - Access to the source code in a sealed room
   - Formal submission of technical questions
   - Formulation of hypotheses and attack methodologies

Cronology of the Public Security Tests

1. Preparation phase (March 6-8, 2012):
   - Opening presentation
   - Access to the source code in a sealed room
   - Formal submission of technical questions
   - Formulation of hypotheses and attack methodologies

2. Testing phase (March 20-22, 2012):
   - Validation of hypotheses
   - Execution of attack methodologies
   - **Joint** writing with the SEC of team reports

Cronology of the Public Security Tests

1 Preparation phase (March 6-8, 2012):
   - Opening presentation
   - Access to the source code in a sealed room
   - Formal submission of technical questions
   - Formulation of hypotheses and attack methodologies

2 Testing phase (March 20-22, 2012):
   - Validation of hypotheses
   - Execution of attack methodologies
   - **Joint** writing with the SEC of team reports

3 Public audience (March 29, 2012):
   - Publication of results
   - Symbolic prizes

Cronology of the Public Security Tests

1. Preparation phase (March 6-8, 2012):
   - Opening presentation
   - Access to the source code in a sealed room
   - Formal submission of technical questions
   - Formulation of hypotheses and attack methodologies

2. Testing phase (March 20-22, 2012):
   - Validation of hypotheses
   - Execution of attack methodologies
   - **Joint** writing with the SEC of team reports

3. Public audience (March 29, 2012):
   - Publication of results
   - Symbolic prizes

4. Publishing of official documents (April 10, 2012):
   - Testing plans and team reports
   - Appreciation by the Evaluation Committee with team scores

Objectives

According to the call for participation:

1. Failure
   - Violation of the system specification
   - Inconsistent state of execution
   - No effect on integrity or secrecy of ballots
2. Fraud
   - Intentional act
   - Effect on integrity or secrecy of ballots
   - No traces left

Team participation – Opening presentation

Election Algorithm

1. Voting machines loaded with software from Compact Flash
2. Zero-proof printed (between 7AM and 8AM in election day)
3. Voting session opened (at 8 AM)
4. Votes cast by electors
5. Voting session closed (at 5PM if there is no queue)
6. Media of Results (MR) written
7. Transmission of authenticated public products (Partial Sum (PS), Digital Record of Votes (DRV), LOG) to the centralized totalizator

Important: Only voting machines were tested!

Team participation – Opening presentation

Digital Record of the Votes (DRV):

- Public access file which replaced VVPATs in 2003
- Stores the votes in shuffled order

Governor	Senator	President
	31	
13		
		BLANK

Team participation – Opening presentation

Digital Record of the Votes (DRV):

- Public access file which replaced VVPATs in 2003
- Stores the votes in shuffled order

Governor	**Senator**	**President**
71	31	
13		
	NULL	
		BLANK
		37

Team participation – Preparation phase

Digital Record of the Votes (DRV):

- Public access file which replaced VVPATs in 2003
- Stores the votes in shuffled order

Governor	Senator	President
71	31	37
	BLANK	
13		
71	NULL	
		BLANK
		37

Team participation – Preparation phase

Digital Record of the Votes (DRV):

- Public access file which replaced VVPATs in 2003
- Stores the votes in shuffled order

Governor	Senator	President
71	31	37
	BLANK	
13		
71	NULL	
		BLANK
		37

Team participation – Preparation phase

Hypothesis: DRV was not designed and implemented in a secure way.

- Is it possible to simulate the shuffling order out of the machine?
- Is it possible to recover the votes using this order?
- Which information is needed for this recovery?

Team participation – Preparation phase

Hypothesis: DRV was not designed and implemented in a secure way.

- Is it possible to simulate the shuffling order out of the machine?
- Is it possible to recover the votes using this order?
- Which information is needed for this recovery?

Pseudo-random number generator (Wikipedia)

It is an algorithm which produces a sequence of numbers approximately random. The sequence is not truly random, because it can be reproduced from a small set of initial parameters, one of them called the **seed** (which must be truly random).

Team participation – Preparation phase

Hypothesis: DRV was not designed and implemented in a secure way.

- Is it possible to simulate the shuffling order out of the machine?
- Is it possible to recover the votes using this order?
- Which information is needed for this recovery?

Pseudo-random number generator (Wikipedia)

It is an algorithm which produces a sequence of numbers approximately random. The sequence is not truly random, because it can be reproduced from a small set of initial parameters, one of them called the **seed** (which must be truly random).

Validation: 1 hour of source code analysis gave us certainty of hypothesis.

Team participation – Testing phase

Preliminarly: Development of analysis tool.

Team participation – Testing phase

Preliminarly: Development of analysis tool.

Methodology

1. SEC produces a secret list of votes
2. Team splits into subteams A and B
3. SEC staff and subteam A insert votes in the machine
4. Subteam B examines public products of the voting process
5. Subteam B runs analysis program to recover list of votes in order
6. SEC staff and subteams A and B verify if both lists are equal

Important: No communication between subteams A and B.

Team participation – Testing phase

Preliminarly: Development of analysis tool.

Methodology

1. SEC produces a secret list of votes
2. Team splits into subteams A and B
3. SEC staff and subteam A insert votes in the machine
4. Subteam B examines public products of the voting process
5. Subteam B runs analysis program to recover list of votes in order
6. SEC staff and subteams A and B verify if both lists are equal

Important: No communication between subteams A and B.

Success: Perfect match between the two lists.

Team participation – Testing phase

Preliminarly: Development of analysis tool.

Methodology

1. SEC produces a secret list of votes
2. Team splits into subteams A and B
3. SEC staff and subteam A insert votes in the machine
4. Subteam B examines public products of the voting process
5. Subteam B runs analysis program to recover list of votes in order
6. SEC staff and subteams A and B verify if both lists are equal

Important: No communication between subteams A and B.

Success: Perfect match between the two lists.

Validation: Absolute success in elections with 10, 16, 21 and 475 voters.

Team participation – Vulnerabilities

1 Weak PRNG:
  - Small 32-bit seed space (`srand()`/`rand()`)
  - Does not attain cryptographic requirements

Team participation – Vulnerabilities

1. Weak PRNG:
   - Small 32-bit seed space (`srand()`/`rand()`)
   - Does not attain cryptographic requirements

2. Choice of seed not truly random:
   - Consists of a time measurement with precision of seconds
   - Violates the operation limits of PRNG
   - Reduces significantly the cost of exhaustive search

Team participation – Vulnerabilities

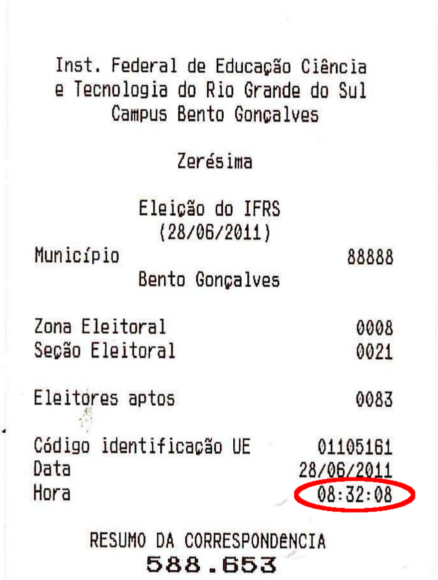
1. Weak PRNG:
   - Small 32-bit seed space (`srand()`/`rand()`)
   - Does not attain cryptographic requirements

2. Choice of seed not truly random:
   - Consists of a time measurement with precision of seconds
   - Violates the operation limits of PRNG
   - Reduces significantly the cost of exhaustive search

3. Seed made public:
   - Printed in public products
   - No exhaustive search needed

Team participation – Vulnerabilities

Secret and truly random seed:



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 CORRESPONDENCIA
588.653

Team participation – Attacks

1. Direct attack:
   - Recover votes in order from the public seed.
2. Indirect attack:
   - Exhaustive search to match "holes" in DRV and recover correct seed.

Team participation – Attacks

1. Direct attack:
   - Recover votes in order from the public seed.
2. Indirect attack:
   - Exhaustive search to match "holes" in DRV and recover correct seed.

Attack features:

- Efficient (exact and deterministic)
- Elegant (*nothing was changed or invaded*)
- Essentially untraceable (only requires **reading** public products)

Team participation – Results and corrections

Results

1. It is possible to recover the votes in order from the public seed
2. It is possible to recover seed from votes out of order
3. Defeated the only protection implemented for ballot secrecy.

Posteriorly: LOG associates each vote cast with a timestamp.

Team participation – Results and corrections

Results

1. It is possible to recover the votes in order from the public seed
2. It is possible to recover seed from votes out of order
3. Defeated the only protection implemented for ballot secrecy.

Posteriorly: LOG associates each vote cast with a timestamp.

Corrections

1. Use a *hardware* RNG (voting machine has two)
2. Use `/dev/random` (viable in practice?)
3. Use `/dev/urandom` with no cryptographic strength, but still superior (secure?)

Going beyond – Consequences

Attacker model

Capable of buying votes and monitoring/coercing voters on election day.

Four kinds of general attacks on ballot secrecy:

1. Attacker records order or time votes were cast
2. Coerced electors vote in the session start
3. First coerced elector cast a marker vote
4. Coerced electors vote in prearranged positions or times

Going beyond – Consequences

Attacker model

Capable of buying votes and monitoring/coercing voters on election day.

Four kinds of general attacks on ballot secrecy:

1. Attacker records order or time votes were cast
2. Coerced electors vote in the session start
3. First coerced elector cast a marker vote
4. Coerced electors vote in prearranged positions or times

One directed attack on ballot secrecy:

- Recover a specific vote, given place and time vote was cast

Team participation – Appreciation

After initial repercussions in the press:

- "Extremely positive contribution"
- "Highly advanced technological attack"
- "We learned a lot with the testing methodologies"
- "We will ask for help from academia in the next two years"

Unappealable report by the Evaluation Committee:

- Attack consists in an attempt to cause failure instead of fraud
- Needs physical access to the voting machine, media, seals and source code
- Score of 0.0313 in 0-400 range

Other problems

Flaws in the software:

- Inadequate source of entropy (17-year old vulnerability)
- Massive sharing of encryption keys
- Presence of encryption keys in the source code
- Violation of specification for block ciphers
- Insufficient verification of software integrity
- Obsolete algorithm (SHA-1) when collision-resistance is needed
- Repeated implementations of cryptographic primitives

Other problems

Bad engineering practices:

- Emphasis on obfuscation instead of security
- Focus on external instead of internal attackers
- No internal exercises or formal training
- Usernames, passwords and hostnames available
- Complete lack of static code analysis

Flawfinder

"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."

- False sense of security

Official positions by the SEC (my translation)

Regarding the media encryption

Using a single shared key is more secure due to *cryptanalytic attacks based on statistical estimators*.

Official positions by the SEC (my translation)

Regarding the media encryption

Using a single shared key is more secure due to *cryptanalytic attacks based on statistical estimators*.

Regarding internal attacks

It is not practical to execute an insider attack without leaving traces detectable by an audit.

Official positions by the SEC

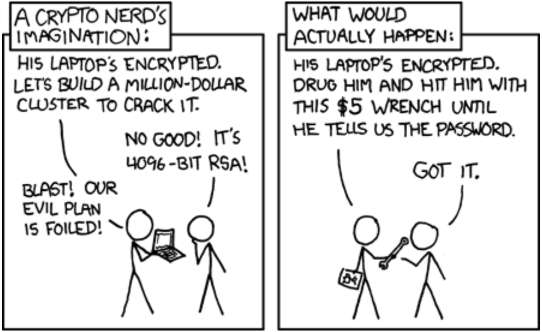
Technical criticism:

- Statistical attacks are of academic interest only
- Leaking the shared key a single time has critical impact
- Mode of operation randomizes plaintext (if used correctly)
- Argument only reinforces focus on external attacks

Official positions by the SEC

Technical criticism:

- Statistical attacks are of academic interest only
- Leaking the shared key a single time has critical impact
- Mode of operation randomizes plaintext (if used correctly)
- Argument only reinforces focus on external attacks



Official positions by the SEC (my translation)

Regarding VVPATs

Obtaining absolute ballot secrecy and integrity is impossible. It does not make sense to focus on integrity only. VVPATs only give a false (and expensive) sense of security. Arguing that a correctable software vulnerability requires reintroducing VVPATs is a *threat to democracy*.

Technical criticism:

- Fallacy of the perfect solution
- Is it really better to trust (demonstrably) vulnerable software produced by a flawed development process?

Conclusions

1. Kerckhoff's principle (1883)
2. Shannon's maxim (around 1945)
3. John von Neumann (1951): "Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin".

Acknowledgments

- My team of brave researchers

- EVT/WOTE 2012 organizers for the invitation

- Increased transparecy from the Supreme Electoral Court

Thank you for your attention!
Any questions?

`http://sites.google.com/site/dfaranha/projects/`