Appendix 2B

Review of Hardware, Software Security and Testing

DUBLIN CITY UNIVERSITY

Charlie Daly
David Gray
Michael Scott
Renaat Verbruggen

Table of Contents

1 Introduction

In undertaking to examine the electronic voting system we decided to approach it from a number of different fronts. Each of these is presented in their own sections below.

- Security Policies & Software Assurance
- Software Testing
- Hardware/Software Interface
- Analysis of the hardened PC

We feel that such a method provides a broader and more thorough understanding of the system than any singular method would provide.

Each section is complete with its own references and self-contained in terms of its description.

1.1 Executive Summary of Findings

Security Policies & Software Assurance

1. There is no well-defined, comprehensive security policy covering the development, deployment or use of the chosen system in Ireland.
2. A number of important security goals have been specified and a significant amount of effort has been expended to ensure that these goals have been or will be met. However, the equally important goals of voter trust in the system and prevention/detection of insider attacks have not been fully addressed.
3. The system should be assured to a recognised international security standard such as ITSEC or the Common Criteria.

Software Testing

1. Critical errors should not be found in software 3 months before live rollout of the full system. This causes new versions of software to be written and released too close to full use.
2. Errors that have been found in the IES software do not inspire confidence in the software development approach taken.
3. A review of the software contains a typographical error.

Hardware/Software Interface

1. The voting system is 1980's technology. In the 1980's the threats to this kind of technology were not as well understood as they are today; furthermore many effective defensive counter-measures have been perfected in the meantime (such as the use of cryptography) which are not deployed here.
2. Security is inadequate. A determined individual insider with short-term access to a voting machine, ballot modules or the count-centre computer could significantly affect the recorded votes. With the possible exception of the voting machine such tampering could be done in an undetectable fashion.
3. Security, such as it is, relies largely on the long discredited concept of "Security Through Obscurity". It is a well-established principle in the world of electronic and computer security, that this is inadequate.

Analysis of the hardened PC

1. The 'hardened' PC is not secure. An attacker with physical access to the computer can gain

complete access to the machine and modify it in any desired way. The attacker could then change the election software or databases to modify the result of the election.

Conclusions

As a result of our study, carried out over a short period of time, we have discovered significant shortcomings in the electronic voting system and there are likely to be more problems as yet undiscovered. These problems will require a significant investment of time and effort to fix. We would have serious concerns about the integrity of any election carried out using the current system.

2 Security Policies & Software Assurance

Dr. David Gray
Senior Lecturer
School of Computing
DCU

2.1 Security Policies

Given the nature of the Irish electronic voting system and its importance to Irish democracy, there should be a well-defined *security policy* and the system should be *assured* to a recognised security standard such as ITSEC [1] or the Common Criteria [2]. From the documentation made available, it would appear that no such policy has been developed and consequently, no security assurance exercises have or can be undertaken.

Two documents (*Request for Tender* [3] and *Security and Audit Features of the Election Management System* [4]) relating to security requirements have been studied. Document [3] has a very short statement about the security requirements of the system to be procured:

> *"4.5 Security – the system will offer full application security, details of which must be supplied."*

These requirements are surprisingly brief and appear to have left the extremely important requirements for security in the hands of the vendor. In addition, it is not clear how responses to the Request for Tender were properly assessed given that there were no detailed security requirements against which to judge the security of proposed systems.

Document [4] appears to be a post-procurement attempt to define the security of the chosen system. While this document could form the basis of a security policy, in its present form it has a number of serious shortcomings.

1. The document establishes five security objectives; namely integrity, confidentiality, enfranchisement, availability and verifiability. While these objectives are extremely important, it is not clear why other objectives were not included.

   In particular, it is not clear why *voters' trust in the system* was not included as a security objective. In many respects, the existing security requirements are too focused on secrecy

issues and have ignored the equally important requirements for transparency so that voters can develop trust in electronic voting and the chosen system.

2. A security policy needs a suitable *security model* defining the components of the system, user roles, threats etc. It appears that no such model has been developed. Again, without such a model it is not clear how the security of the system can be assessed.

   In addition, in many cases it is not clear what threats particular security mechanisms are intended to counter and it is also not clear that the possibilities of *insider attacks* have been properly addressed.

3. Most secure systems have audit trails and there is typically a separation of duties between personnel operating the system and personnel responsibility for ensuring the security of the system. In many cases, the different roles are fulfilled by personnel from different organisations or at least, different departments/sections within the organisation owning/operating the system. Typically, security personnel are given read-only access to audit information and normal users (via the software they use) are given append-only access, and audit trails can then be used to monitor the use of the system and detect security breaches.

   Document [4] discusses auditing for the chosen system in terms of the features available rather than requirements that the system should exhibit. As described in document [4], these features are far from adequate.

   a. In general, only the external operation of the system appears to be audited. For example, users are required to complete paper documents and attach printouts from the voting machine and IES Software. There does not appear to be any audit information generated and stored automatically as the system is being used.

   b. The information that is audited appears to be rather limited. In particular, the actual voting data is not audited and the integrity of the votes cast and the correctness of the count depends entirely on the correct operation of the software. This approach makes it impossible to monitor the behaviour of the software for both correctness and possible attacks.

      Note that there is a distinction between the use of “internal audit trails” to allow electoral staff to check the correct functioning of the system and to detect breaches of security, and “Voter-Verified Audit Trails”; see below.

   c. The auditing features identified in document [4] appear to offer only weak security. For example, during polling, the voting machine prints a number of documents that are used to audit the voting process. While these documents are useful for checking for hardware, software and human errors, they could not resist a knowledgeable insider attack.

   d. Document [4] gives no details on the separation of duties of the different staff configuring, programming and operating the system. Some information is available in other documents; but this information is not related back to security requirements and the mechanisms used to prevent attacks. As a result, it is extremely difficult to determine what trust needs to be placed in the personnel operating the system.

In addition, there appears to be no well-defined role equivalent to a security officer.

e. As indicated above, the main focus of security in document [4] is on secrecy and integrity aspects of the chosen system. Hence, much of the design and testing of the system is directed towards secrecy and it would appear that voter trust in the system has not been considered important. So while it may be possible to arrive at a system that is secure relative to these secrecy and integrity requirements, it is very unlikely that this will lead to voters trusting the system. In particular, there remains the possibility of a coordinated insider attack that could not be detected.

   Voter trust could be established by the participation of a Trusted Third Party (TTP) in the operation of the system or better still, the use of a Voter-Verified Audit Trail (VVAT).

4. Document [4] does not draw a distinction between the security requirements and the security features of the chosen system. As a result, limitations of the system are not being explored and improvements to address security weaknesses are limited and rather ad hoc.

It is clear that considerable effort has been expended on trying to establish the security of the chosen system. This has included work by a number of independent test centres and the results appear to be encouraging. However, without a well-defined security policy, at best, these efforts can only be ad hoc. In particular, it would appear that the security objectives [4] have been developed in relation to the procured system and not independently. As a result, other objectives (most notably trust and the prevention/detection of insider attacks) have not been thoroughly addressed.

2.2 Software Assurance

Assurance is the process of establishing that a system can be *trusted*. This entails establishing that the system satisfies its functional specification *and does not have any undesirable behaviour*. In particular, that it does not have any undesirable security weaknesses. Assurance is not solely a technical issue as its goal is to give humans sufficient confidence to trust a system.

For software, high levels of assurance can only be achieved by a combination of testing and source code inspection/analysis. As is well known, black box testing can be used to detect bugs and raise one's level of trust, but only by inspecting/analysing source code is it possible to determine the total behaviour of a software component, and possibly prove the absence of bugs. In addition, software assurance normally requires a secure development methodology using trusted personnel.

A number of documents [5..11] relating specifically to the software components of the chosen system have been studied. However, no access was granted to the source code for any of the software components.

There is no documentation to indicate that there has been any attempt to assure the system using a recognised international standard. However, it would appear that an ad hoc effort is being made to establish the functional correctness of the software components of the system and that the security objectives have been met. However, there are a number of serious shortcomings with the current approach.

1. There is no sufficiently detailed security policy against which the software can be assured.

2. The system testing currently being undertaken is not intended to achieve ITSEC or Common Criteria assurance.

3. It is generally accepted that obtaining assurance after software has been developed is problematic and the recommended approach is to use an assured development methodology throughout the lifecycle of a software product. There is no indication that such a methodology has been used during the development of the chosen system.

4. The testing related documents [7..11] describe the use of a mixture of black box testing and code inspection/analysis. There is also evidence that tests are related back to specific requirements given in [5].

   However, there is insufficient detail about how many of these tests were conducted and to ascertain what *assurance level* could be conferred on the system as a result of these tests.

   For example, § 6.2-12 of [8] claims that the requirement 6.2-12 of [6] "The vote shall be recorded only once" is fulfilled since "The variable is copied once only, and the storage function is called only once". It is not clear how the tester arrived at this conclusion.

   a. Was it by manual inspection of the code?
   b. Was it via some rigorous/formal analysis of the code?
   c. Was it achieved by using an analysis tool?
   d. Would this conclusion stand up to peer-review?

   In addition, this requirement is not limited to the voting machine; the entire system must satisfy this requirement. There is no evidence that other components were tested to ensure that they also satisfied this requirement.

5. Assurance cannot be achieved via the ad hoc commissioning of testing exercises. In addition to requiring an assured development methodology, it must be possible to repeat such testing in a controlled and transparent manner as the system is updated. It is not clear that the testing reported in [7..11] could be easily repeated.

2.3 Transparency & Trust

The current approach to assuring that the security objectives are met is to demonstrate that the chosen system is 100% correct and can be operated without the use of comprehensive audit trails and/or the involvement of trusted third parties in the operation of the system. In addition, there seems to be a belief that restricting access to information about and generated by the system will improve security.

Using this kind of approach it is extremely difficult to build a secure system that can command high levels of trust. However, if sufficient effort is expended and a proper approach to assurance is adopted, it might be possible to assure the system to an acceptable level. Given the current security objectives, such an approach would give election officials a system that *they* could trust to hold fair elections.

From reading the various documents and operating the system, it is obvious that there are a number of outstanding bugs and security weaknesses. Therefore, it is very unlikely that an acceptable level of assurance could be achieved by June 2004.

However, given the current security objectives, it is very unlikely that an assured system would lead to voter acceptance and trust in the system. In particular, the operation of the system is not transparent and the electorate would have absolutely no way of determining if the election process had been subject to an insider attack. Page 12 of document [4] states:

"The voter can have confidence that the software properly records his/her vote as it has been tested and certified by an independent international institute."

Given the current system and security objectives, this is an unrealistic expectation.

2.4 Bibliography

[1] *Information Technology Security Evaluation Criteria*, Harmonised Criteria of France, Germany, the Netherlands, and the United Kingdom, Version 1.2, June 1991.
[2] *Common Criteria for IT Security Evaluation*, NIST, http://csrc.nist.gov/cc/index.html.
[3] *Request for Tender*, Department of the Environment and Local Government, June 2000.
[4] *Security and Audit Features of the Election Management System*, Department of the Environment and Local Government, January 2004.
[5] *Software Requirements for Voting Machines for Use at Elections in Ireland*, March 2003.
[6] *Functional Specification, NEDAP Voting System ES12, Powervote*, April 2003.
[7] *Type testing of a voting machine for elections/referenda in Ireland*, Physikalisch-Technische Bundesanatalt, September 2003.
[8] *Test report 2 – Voting machine ES12 – Software for elections in Ireland*, Physikalisch-Technische Bundesanatalt, September 2003.
[9] *Architectural Assessment & Code review of IES for use at June 2004 Elections (Build 0111)*, Nathean, December 2003.
[10] *Code Review of IES Build 0111*, Nathean, December 2003.
[11] Report on Irish STV Software Testing, Joe Wadsworth & Brian Wichmann, March 2004

3 Software Testing

Renaat Verbruggen
Lecturer
School of Computing
DCU

3.1 Introduction

The aim of this section was to analyse the software testing that has been achieved on the voting software. In the absence of any source code no extra reviews could be completed and all tests were done in a "black-box" fashion based on the existing code supplied with no internal access. Because of the limitations of this approach and the extremely short time-scale involved it was not felt worthwhile to try to run a battery of new tests against the system. Instead a comprehensive review was carried out of the test reports that already exist. This meant reading the reports evaluating the approaches and searching for potential flaws discovered.
Results from this approach are described below.

3.2 Examination of: Nathean Technologies Code Review of IES Build 0111, Date: December 2003

This document [1] relates to reviews of modifications carried out to the source code. The document is constrained in its introduction with the following condition:
"*By agreement, Powervote did not supply any units which were not modified or specifically written for the Irish edition of IES*"

This is an obvious limitation as errors can still arise in so-called "old" software simply because the new software calls it in unexpected or incorrect ways.

In any case the document then reviews each of the software units giving short summaries of their contents and no problems are indicated with any modules.

However when the reviews were examined the following issue was found:
Page 12 gives a review of the unit **Calc_Count_Section.pas.** The unit contains code used to determine which surplus to distribute and Pseudo-code is given.

If iNumSurpluses = 1 then
DistributeSurpluses
Else iNumSurpluses = 1 then
Begin
    If LargestSurplus

    PseudoCode continues

As it is written the PseudoCode does not make any sense as the condition on both the if and the else is the same **(iNumSurpluses = 1)**. If this was implemented the code would never distribute a surplus if there was more than one surplus to distribute. We cannot see the source code that this review reflects however the possibilities are:

- The PseudoCode is a correct reflection – therefore both code and PseudoCode are wrong.
- The PseudoCode is wrong but the code is correct.

The main point is that getting PseudoCode wrong in reviews reduces confidence in the other results of the report and the indication that no new significant issues were raised during this examination.

3.3 Errors existing in code in February 2004 Email correspondence. [2]

From a review of testing at ERS in February 2004 it appears that a number of errors still existed in the source code of version 124 of IES.

- These included error messages appearing on screens during elections requiring drawing lots.
- Errors in arithmetic in calculating surplus remainders for distribution
- Database size becoming extremely large.

Each of these errors was tackled and some solutions have been put forward in the documentation. However, two important matters arise. The first is that new versions of the IES software are still being worked on (March-April 2004) and critical errors (i.e. errors that could cause a change in an

election results) are still being fixed in February and March. In fact we were sent a new version of the software in late March and there was no indication that this would be the last. For a system due for a May rollout this is a worrying trend and does not inspire confidence.

Secondly the treatment of one of the errors requires further study.
This was a rounding error that existed in version V124, V125, V126 and to a lesser extent in version V128.

The cause of the rounding error was the use of inappropriate types for storing values used in calculating surplus transfers. Appendix B of the ERS report [3] outlining this error history is the main reference. We would commend ERS and their approach in testing for this error. However we also read the responses [2] from the developers about this error when it manifested itself in Version 124. The developers appeared to belittle the importance of the error and having chosen an incorrect type Real, proceeded to choose an extended type that still did not remove the problem.

Quote from one email: Comments 113 EM – Developers' response (verbatim): [4]

"….these problems where caused by me using a variable type 'Real' for handling the calculated average for each candidate in the surplus distribution. This is the 'old' floating point type use in delphi. The later versions of Delphi offer also a floating point type 'Extended' which offers greater precision in computer arithmatic (no rounding problems)"

The email continues to explain they had found no errors themselves. The problem is that they do not realise how inappropriate it is to make simple equality comparisons of real numbers. Real numbers inherently can contain rounding errors.

Again due to good testing procedures at ERS the error no longer manifests itself in version V129. The developers' attitude is naïve for a software development company developing such a critical application and the process of fixing errors has required 5 new versions before the problem was eliminated. This leads us to believe that the problem, deemed trivial, was in fact not properly understood.

3.4 Maintenance Measures for IES software

The ability to use a maintenance mode with a single password to gain access to the IES system is another big weakness of the delivered system. Although we were in special circumstances, the password was sent in plain e-mail to a number of separate recipients and there is no indication that this maintenance mode will be disabled for the election process itself.

3.5 Randomness

We studied the reports and data associated with testing of the IES – File numbers F429 1 through 12 [5]. The approach of ERS to testing the IES software is exemplary and does give confidence in their results. They do however add an important constraint.

Due to the randomisation process associated with mixing votes for distributing surpluses it is impossible to recreate test circumstances from scratch. This is a weakness that has been deliberately added to the system to mimic the randomness of the current counting approach. However simple procedures associated with setting the random number could be used to allow exact results to be

recreated and tests to be re-run. This should make it a lot easier to ensure that faults found have been properly eliminated.

3.6 Ability to produce printed Ballot papers

From studying the IES software it is clear that "paper ballots" that is paper copies reflecting actual votes cast in the election can be created post-hoc after the count is completed. While not an audit trail by any means this facility could have been used to engender trust in the system by creating a parallel votes scenario as a public awareness event. Each voter could create a hand-written ballot, and then use the electronic vote system. At the end all votes could be matched up to show that the votes were correctly captured. Unfortunately the randomisation feature as outlined above would prevent an exact creation of the same poll result. However with some thought this feature could also be set for demonstration purposes.

[1] Code Review of IES Build 0111, Nathean, December 2003.
[2] Email correspondence copied at Department of Environment from File 429 – 12, ERS – testing of IES 2004.
[3] Report on Irish STV Software Testing, Joe Wadsworth & Brian Wichmann, March 2004.
[4] Comments 113 EM ERS related surplus distributions.
[5] Files F429-1 through F429-12 Documentation, Correspondence and Data used for the testing of IES by ERS.

4 Hardware/Software Interface

Dr. Mike Scott
Senior Lecturer
School of Computing
DCU

4.0 Introduction

Any voting system must meet the following critical requirements

- It must be accurate.
- It must be safe from interference by any third party.
- It must be secret. In other words no third party should be able to determine how an individual cast their vote.
- It must not be possible for a voter to prove to any third party how they voted.

It would also be nice if

- The process were fast
- The cost of running an election could be reduced
- A voter could be sure that their vote had been counted

Voting systems are vulnerable to abuses, such as

- Impersonation "vote early - vote often". This is an example of identity-theft.

- "Voting the register" - that is after the poll is closed someone votes on behalf of those that have not actually turned up to vote. This is an "insider" attack on the system.

The comparison here is between a manual system and the chosen electronic system. An abuse which applies equally to both systems will not be considered further. This applies for example to "Impersonation" and "Voting the register".

The electronic system is clearly faster and potentially a lot cheaper in terms of the cost of running an election.

We would make the following general comment about the chosen voting machine - it is very old-fashioned technology. Many advances in technology and security in the past 20 years have not been included in its design.

However, here we concentrate first on the four critical requirements.

4.1 Accuracy

An electronic system certainly has the potential to be more accurate than a manual system. Indeed it can be more accurate. The manual system distributes surpluses by selecting second (and lower) preferences from a "random" sample of the successful candidate's votes. With the electronic system this can be done precisely without any randomness. However the chosen system maintains the random element for "compatibility" reasons. We think this is a big mistake. It draws unnecessary attention to the shortcomings of the electronic method for randomisation (See Appendix A).

Whether or not the chosen system is in fact accurate is impossible to determine without access to the source code, and without more testing that was not feasible within the time-scale available to us.

4.2 Third-Party Interference

In the manual system, one of the major disadvantages is the number of people that are needed to implement it properly. However this is an advantage when it comes to security. Any attempt to organise a conspiracy would flounder on the sheer numbers who would need to be involved. The electronic system involves less people, therefore conspiracies become easier, therefore the system must have enhanced security features to make this impossible.

4.2.1 The Malevolent Voter

What can this individual do?

First there are the "denial of service" attacks which aim to prevent use of the machine. This can involve the infliction of physical damage to the voting machine. Or a conspiracy of voters may get together and nominate more candidates than the system can support, forcing a reversion to a manual count. Or a conspiracy of voters might vote every preference for every election hoping to exceed the capacity of the ballot module to store that much information.

Deception attacks. An attacker equipped with black masking tape could place this over a candidate's name. This might fool subsequent voters into thinking that the candidate had withdrawn from the election.

The "false template" attack. The attacker prepares a false ballot paper overlay and places it over the panel of the machine, sellotaping it into position. The order of a pair of candidates on the false template are swapped to the advantage of one of them. Then masking tape is placed over the bottom line of the display so that the name of the candidate selected cannot be seen.

Place an LCD display over the real display, and control it remotely to issue false instructions to a voter.

Cover the real cast vote button with black masking tape, and stick beside it an identical "dummy" button that looks the same but does nothing.

Recommendation

Replace the black display surround with a more complex checker-board pattern that would be harder to overlay. Polling booths must be regularly checked to detect any tampering.

We don't consider any of these attacks to be particularly serious.

4.2.2 The Malevolent Official/Technician - the insider attack

This is much more problematic. A determined insider can potentially do a lot more damage. Note that there are three places where an attack might be effective

- An attack at a polling station, by compromising an individual voting machine or individual ballot module. If successful this could effect maybe several hundred votes.
- An attack on one or more ballot modules as they are in transit to the count centre. This could effect thousands of votes.
- An attack at the count centre, on the computer that collates votes from multiple ballot modules. If successful this could directly affect the outcome of an election.

The controlling program inside the voting machine could be modified to affect the vote. The program has a "checksum" which protects it against accidental changes, but does not protect against deliberate tampering. Anyone with appropriate technical knowledge and access to the voting machines could modify the program and at the same time yield the same checksum. It also appears that the program is being continually modified by the manufacturer to fix minor bugs. At any stage a programmer could introduce rogue "trojan" software. See Appendix D.

In practice it took a technician about 40 seconds to open the machine from the back. We observed that the controlling program chips are actually socketed for ease of access. Therefore there is little to prevent removal and substitution of the program (see Appendix C). We estimate that 2 minutes of unauthorised access would be sufficient to switch programs.

In figure 1 the program chips (ROMs) can clearly be seen. They are the chips with the white stickers.

Figure 1. Inside of Voting Machine

However there are paper seals that were broken in the process of gaining access. This offers some hope that tampering would be detected. We do not know what processes are in place to check these seals for tampering. On the face of it, it would seem quite easy to cover the broken seals with pre-prepared substitutes that would fool the naked eye. For access it would be sufficient perhaps to break only one of the two, and perhaps cut the second with the pen-knife.

<u>*Recommendation*</u>

It is possible to use technology to protect against tampering. After the software has been analysed by independent and trusted experts, its cryptographic hash could be calculated and displayed on a public Web page. This cryptographic hash (a kind of superior checksum - for example the international standard SHA-1 algorithm could be used) is a unique fingerprint of the software. It reveals nothing about the software itself, but if the software were subsequently tampered with the fingerprint would change. After an election an independent body should take away a random sample of voting machines, and recalculate their hashes. If they match the public value then one can have complete confidence that the software has not been tampered with.

When the ballot module leaves the machine it instantly becomes a vulnerable target for attack.

The ballot module is itself a purely passive storage device. Anyone who has access to it, with the appropriate equipment and the appropriate "insider" knowledge, could change its contents to anything they wanted in a fraction of a second. Compare this with similar access to an old-fashioned manual ballot box. Its only protection is in the obscurity of its design. And even that might be relatively easy to determine by "reverse engineering" one of the modules - they are really quite simple devices. An attacker could design their own special interface to a ballot module

connected to a small laptop PC with which to change the module contents.

Some protection is provided by the back-up module which is fixed inside the voting machine. This can be compared with the primary module if tampering is suspected. However, if tampering is not detectable, then the back-up module is irrelevant.

The module itself is easily opened by pulling apart the blue plastic cover. This reveals a very simple circuit, a flash-memory based passive storage device. From a detailed visual examination alone, a competent electronic technician could create their own identical module. The chips used are Intel P28F010-150, and the total storage capacity is 256k bytes. Typing "P28F010" into Google immediately brought up the full specification of the chip [1], programming voltages, etc. Equipped with this information an attacker could create a device to reprogram the module. A few seconds of unauthorised access would be sufficient either to erase the contents or to replace them. Extracting the contents of a module used in an election would we suspect be sufficient to determine the internal formats, allowing data to be undetectably modified. However, we have not had time to attempt this. Figure 2 shows the internals of a ballot module.

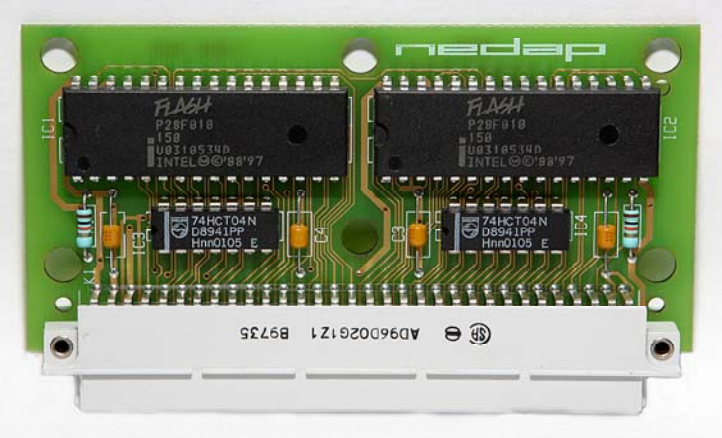


Figure 2. Inside of Ballot Module

We note here that these issues were addressed in part by the German testing organisation PTB in their report [4]. They were responding to questions asked by the Irish Government in their document [5]. We do not know how these list of questions were decided upon, but you can only get answers for the questions asked, and these in our view are not the right questions. Many of them appear to be taken from the manufacturers' functional specification document. In other words many of the questions were asking if the manufacturers' machine did what the manufacturers claimed it did. Which, not surprisingly, it did. Better questions would have been motivated by a deeper knowledge of what was required from an electronic voting machine. In Appendix C we attach the response from PTB to some questions of ours. We draw particular attention to the response to our second question. It is frankly admitted that the ballot module design is a case of "security through obscurity" (Appendix B). There is also an admission that the question of the vulnerability of the ballot module outside the machine was not addressed - "Tests of how the ballot module is protected when it is outside the machine have not been performed. They are not in the scope of the requirement". In other words we did not address that question because we weren't asked to.

The hard questions about the security of the chosen system do not appear to have been asked. There is no independent security analysis that we are aware of. A lot of the software/hardware testing that

has been carried out has little relevance to security.

Recommendation

The contents of the ballot module should be protected using cryptography. Cryptography is the IT industry's first weapon of choice when it comes to protecting data and communications (data-in-transit). It is not used here at all. Various standard solutions are available. The contents of the module could be encrypted, or protected from malicious modification by a MAC (Message Authentication Code). Anyone not knowing the secret key could not tamper with the module contents, even with full knowledge of the module's design. Alternatively the SHA-1 hash algorithm could be used to calculate a unique fingerprint of the contents which could be forwarded via an alternative secure channel to the count centre, where again tampering could be detected. Admittedly modifications such as these would involve a major redesign of the voting software.

When the ballot modules arrive at the count centre, their contents are extracted and further randomly mixed (see Appendix A). All the votes are then collated on a "hardened" PC that runs the software to perform the count and declare the results.

Note that before the vote takes place at all the ballot modules are formatted on this same machine, and details of candidates and the elections are placed inside the modules. They are then distributed to the voting machines. After the vote they are extracted and returned to the PC.

Here is another weak point. Whoever has control over this PC is in a very powerful position to tamper with the results of an election. This might well be an individual, as typically only one person at a time sits at and uses a PC. And this time there is not even the spurious protection of obscurity - once logged in this is the familiar Microsoft environment that computer students and hackers alike know only too well. It would be easy to introduce a virus, or a relatively simple program to affect election results. Note that virus protection software only protects against known viruses. It offers no protection whatsoever against new viruses of the type that might be written to attack this specific system.

The "hardening" of the system does attempt to limit access to the machine to the holder of a smart-card and with knowledge of a PIN number, and this attempt is to be commended. However as reported elsewhere this "hardening" is completely ineffective. It can easily be by-passed.

This computer is the weakest link in the whole system, and is at the same time the most effective place to attack the system.

Recommendation

The count centre computers need to be properly "hardened" to prevent illegal access. This is quite hard to do technically, and therefore rigid physical security would need to be in place as well. In particular at no time should an unmonitored individual be allowed near these computers. Even with strict controls we are nervous about the level of security which can be attained. In practice if something goes wrong, a "programmer" will be called upon to fix it. That individual, in a few seconds, could do a lot of damage without risk of detection. It would be beneficial if the computers supported key-loggers, so that an audit trail of activity on the computer could be maintained. This would at least make possible the collection of evidence if a crime were suspected. The existence of a key-logger would be a deterrent.

4.3 Secrecy of the ballot

In a manual system the secrecy of the ballot depends on the fact that the act of voting takes place in private, and that the votes get randomly mixed inside the ballot box.

These are relatively easily emulated in an electronic environment, although the randomisation in the chosen system could be improved (Appendix A). Also we have a small concern that in the high-tech environment of an electronic voting booth a small camera might go unnoticed, whereas in the no-tech environment of a manual polling station any wires or electronics would be easier to spot.

4.4 Vote buying

If you can prove how you voted, you can sell your vote. Or you can be coerced into voting in a particular way. Our attention has been drawn to the fact that modern mobile phones have a facility to record up to 3 minutes of continuous video. These are becoming increasingly popular, and quite cheap. We imagine that it will become a standard feature in the near future. Already in Italy such phones are banned from polling stations due to real concerns about Mafia intimidation and/or vote buying. See this recent BBC report [6].

Compared to the manual system, the voting machine of the chosen system seems to be particularly vulnerable. A voter holds the phone close to their chest in one hand and videos himself placing preferences and finally pressing "Cast Vote". This can be filmed in a complete uninterrupted video clip. When played back outside the polling booth to the person buying our vote, we can replay the clip, which will be satisfactory evidence to ensure payment. Note that no phone call is made - we are simply using the mobile phone as a minature camera. The vote-buyer might indeed lend us the camera while we vote, to be handed on to the next person when we leave.

It might be maintained that the manual system (as used in Italy) is vulnerable to the same abuse. However, we would consider that this would be much more difficult. In the first place its hard to mark a ballot paper with one hand while filming with the other. Secondly in the manual system the actual casting of the vote - the placing of the ballot in the box - is done in public view. Any filming of the action of the act of inserting the ballot in the box would raise the alarm. Even if we film ourselves marking the paper the vote buyer can not be sure that we did not subsequently erase it and change it before putting it in the box.

Recommendation

Consider moving the Cast Vote button to the outside of the polling booth in full view. The user places their preferences on the machine, exits, drawing a curtain over the booth, and then presses "Cast Vote" in public view. This would more closely emulate the manual system.

4.5 "The Computer Ate my Vote"

How does a voter know that their vote has actually been counted? In the manual system of course we don't know this for sure, but we commonly get to see someone's votes being counted. There are pictures on TV of ballot boxes being emptied on a table, and people are seen to be counting papers….

With the chosen voting system there is not even this limited amount of transparency. Note that electronic voting schemes have been proposed which do allow this facility [3].

Appendix A - Randomisation

In this Appendix it is important to note that we are dependent on published documentation [4,7,8]. We have not seen the actual software.

True randomisation is actually quite hard to achieve with electronic circuitry. Electronic circuitry is normally *deterministic*, that is given its current state, its next state can be accurately determined. The predictability of its behaviour is in fact important in almost all applications. It is quite hard to get it to behave unpredictably, or randomly. True randomisation can be obtained from a physically random process, like radioactive decay, or electronically by generating unpredictable electronic "noise", and sampling from it.

However there appears to be no true randomisation in the chosen system, either in the voting machine or in the PC software.

A.1 The Voting machine

Randomisation is important here to preserve the secrecy of the ballot. It is important that the order of votes in the ballot module cannot be related back to the order in which people voted, otherwise an unscrupulous official with access to the ballot module could determine how each individual voted.

On the chosen machine, according to the provided documentation, the machine as it idles between votes, cycles a pointer around all the empty locations in the ballot module. This pointer moves every 8 thousandths of a second to the next location. When a voter presses "Cast Vote", their vote is placed in the position currently pointed at. Therefore, the randomisation comes not from the device itself, but rather from the random behaviour of voters.

An unscrupulous official places a small microphone close somewhere inside or close to the polling booth. This microphone picks up the loud beep from the cast vote button and hence determines where in the module the vote will be placed. Note that it is possible to determine the position of the first vote, by having an accomplice cast a "distinguished" vote, that is a very unlikely sequence of preferences. Such a vote can be detected inside the module.

Note that the electronic polling booth is already a "high-tech" environment. Extra electronic equipment such as a small microphone (or even easier a small camera) might not be spotted. The reader will be aware of criminals targeting ATM machines by placing a "skimming" device over the card reader, combined with a small camera to read the entered PIN number [2].

Recommendation

The voting machine should either be modified to include a true random source, and/or it should make more use of the voter's behaviour to improve the randomness. For example every button pressed should contribute to the randomness, not just the cast vote button.

A.2 The PC

The PC software makes use of the built-in Linear Congruential pseudo-random number generator. This appears to be the standard generator that comes with the Delphi programming environment that was used to develop the software. This generator takes a truly random seed and from it

generates a random-looking sequence. This type of generator was invented by Lehmer in 1949. While adequate in many applications, *it is to be avoided in any application where the randomness is critical to the system function*.

In this application it is used to further mix the votes received from the ballot modules and to perform the random distribution of surpluses.

The Lehmer generator has a number of serious failings. First its output is in fact completely predictable, even though it is "random-looking". Given the initial seed value, its entire output can be predicted. Indeed, given even a small part of its output, its past and future behaviour can be completely determined.

The software uses the time-of-day to seed the generator. The time-of-day in fact includes very little randomness. The effect of a particular seed value on the progress of the counts could be predicted. The randomness could be controlled in such a way as to favour a particular outcome.

Recommendation

There has been a lot of progress since 1949! Some versions of the Pentium processor now include a truly random source, based on unpredictable noise generation. Secure true random number generators are widely available, and there is no excuse not to use one in this application.

Note that random number generators are themselves a tempting target for attack. A small software virus could easily disable the randomisation process. Note that the randomness is entirely opaque to the user. Would you play poker with a computer if the computer gets to "randomly" deal the cards? Observable randomness, like coin-tossing or card shuffling, engenders much more trust.

Appendix B - Security through Obscurity

This is a mantra for IT security experts. It's a how-not-to-do-it mantra. Security Through Obscurity (STO) is the computational equivalent of hiding the key under the mat. For years people have tried to secure information and communications depending on the presumption that no potential attacker would know the method they were using. However *there is no long-term security in the perceived obscurity of the technology*. Nonetheless people keep making this mistake. The Germans in WWII relied on the fact that the Allies did not have access to their encryption machinery - the famous enigma machine. But the Poles had smuggled one to Britain at the start of the war. The designers of GSM mobile phones assumed that no one would know the details of their personalisation code, until one day it was posted to the Internet. There are endless anecdotes…

And it is not necessary. Properly secured systems can be designed that do not rely on STO.

The hiding of the Microsoft Access password in the executable file as detected in the IES software is a classic example of STO. It was secure as long as no-one thought of looking there. But someone did… Needless to say this does not give us confidence in the software or in those who designed it.

Appendix C - PTB testing

This is a reply received to questions asked of the German software testing company PTB, concerning the PTB tests [4]

Dear Dr. Scott,

After consulting our specialists, I would like to answer your questions.

Before stepping into details, let me state a few general things. The statement "The requirement is satisfied" which you can find repeatedly in our report is always related to the requirement as it is provided by the Irish government in the document "Requirements for Voting Machines for Use at Elections in Ireland, DVREC-2". (Compare page 3 of the test report).

As a testing body, we do not judge whether the particular requirements are sufficient to implement the general principles of democratic elections in Ireland. However, as a general position, we believe that the level of security required should not unreasonably exceed the level of security required for elections carried out in the traditional way. In particular, when the vote casting takes place in supervised places (polling stations), the technical requirement may consider this fact.

Now to the particular questions:

Question:

"1.4.1 Any alteration of the installed software by an unauthorized person should be detected".

Your conclusion in your report is that "The requirement is satisfied". However you accept that an exchange of ROM chips including presentation of the correct checksums is possible.

Are the ROM chips in fact sealed? If so exactly what kind of sealing is used? What would be required for an unauthorised attacker to remove the ROMs, put in place his own, and reseal them?

Our answer:

An exchange of ROM chips is only possible by breaking the seals, that means each storage exchange is detectable. In fact, the complete electronic unit including the ROM chips is two-fold sealed, by two independent plastic films which cannot be unfixed. This measure corresponds to the voting machine requirements of the Irish government (DVREC-2).

Question:

"1.4.2 Any alteration of the content of the ballot module by an unauthorised person should be detected."

You conclude again that "the requirement is fulfilled".

Do you agree with me that if I had a computer connected by an interface of my own design to the ballot module, that I could in fact program the ballot module in any way I pleased, including the presentation of false checksums? I could also reprogram the module number.

The ballot module is a purely passive storage device is it not? Therefore it can be cloned and modified by anyone with knowledge of its internal formats, programming voltages etc. Is this correct? Would you agree that the obscurity of the module design is in fact its only protection against attack if it should fall into the hands of a well-equipped and well-informed attacker?

Our answer:

Yes, the ballot module is a purely passive storage device. The technical protection against attacks is (only) given by the secrecy of the module design.

In addition, hardware-oriented security measures of the voting machine prevent from irregular writing the ballot module as long as the device is inside the machine (this is the case during the election). Tests of how the ballot module is protected when it is outside the machine have not been performed. They are not in the scope of the requirement. In both cases, the ballot module is inside or outside the machine, it is only possible with the help of specific knowledge to technically manipulate the module. The at least necessary specific knowledge is: - programming procedures of the ballot module (commands, sequence of commands, voltages), - contents and formats of data (which kind of data at which position, in which format and in which byte length, etc.), - redundancy and checksums (type of redundancy, number of repitians, type of checksums, areas that are protected by checksums, position of the nominal checksum, etc.).
This knowledge would (only) be available from the chip manufacturer (first item) and from Nedap or PTB (second and third item).

Question:

"8.8-1 The votes should be stored randomly in the ballot module"

You say that "the first vote is unknown". Do you accept that if certain voters deliberately cast distinguished votes that their positions in the ballot module can be detected, and that if a microphone is able to pick up the sound of the "cast vote" button and determine within 8ms the moment that the button is pressed, then the votes of other voters may be determined? By a distinguished vote I mean one that is extremely unlikely to be cast by anyone else, for example an unlikely sequence of 10 or more preferences.

Our answer:

If the first voter casts a distinguished vote and no other voter casts the same vote, then the position of the vote of the first voter can be detected in the ballot module, provided the means are available to access unauthorised to the ballot module and to perform the necessary search. Furthermore, under the same condition, one can detect the position of other distinguished votes cast, not only that one of the first voter. Precisely speaking, after some votes have been stored, it can be detected that there are distinguished votes, but it cannot be detected in which order the votes have been cast. However, to disclose the secrecy of votes, the voter itself would have to disclose his/her vote. Here we have the same situation as with traditional ballot boxes. Now to the possibility of picking up a vote: The cycle rate of the timer is about 8 msec. Storing of votes does not happen in the same moment as the CAST VOTE button is pressed, but several hundreds cycles later on. To detect the positions of votes outgoing from the point-in-time of pressing the CAST VOTE button, the following information must be available:

- which commands are processed between pressing the button and storing the votes (length of corresponding assembler commands measured in cycles),
- what is the starting point of the timer (it is not the point-in-time of activating the voting machine),
- the precise cycle rate (8 msec is an estimated average value. The real value is not known).

I hope we could answer your questions exhaustively.

Yours sincerely
Dieter Richter

Prof. Dr. Dieter Richter
Physikalisch-Technische Bundesanstalt
Department of Metrological Information Technology
Abbestr. 2-12, 10587 Berlin, Germany

Appendix D - Trojan code

We once had the dubious privilege of writing software for a “poker” machine. This gambling device used a complex internal algorithm to control the dealing of the cards in such a way as to ensure a profit for the machine. As a debugging tool we wrote a small piece of software which, in response to a very unlikely (and physically difficult) combination of simultaneous button depressions, revealed the status of the machine, basically whether it was ready to pay out or not. Clearly this knowledge would be a big advantage to a gambler.

Such code as this is very easy for a rogue programmer to insert for malicious purposes. In this context the trojan code might cause a single vote to be counted a thousand times. However, it would take a lengthy process of code review for another programmer to spot it, as modern programs have typically many tens of thousands of lines of code.

References

[1] www.sunmark.com/datasheets/28f010.pdf
[2] http://www.utexas.edu/admin/utpd/atm.html
[3] http://wired.com/news/politics/0,1283,62983,00.html
[4] PTB Test Report 2 - Voting machine ES12 - Software for elections in Ireland
[5] Requirements for Voting Machines for Use at Elections in Ireland, DVREC-2.
[6] http://news.bbc.co.uk/1/hi/technology/3033551.stm
[7] Nathean Code Review, December ’03
[8] Electronic Voting and Counting System, Technical Info Paper, Government Information Paper Jan 04

5 Analysis of the hardened PC

Charlie Daly
Lecturer
School of Computing
DCU

This report examines the hardened PC Hewlett Packard D530 and associated election software. A number of weaknesses are identified with the PC and an attack is described that allows a user without a valid account or smartcard to gain complete control of the system. The attacker could install a program which waits for the results to come in and then modifies the database files so that their chosen candidate wins the election.

5.1 Security issues with standard PCs

PCs were designed from the start to be open architecture; generally, the motherboard and other components may be removed or replaced. In particular, the hard disks may be removed and examined in another machine. The BIOS and other code is generally easy to examine. In fact, Microsoft say that if you don't have physical security, you have no security [1].

There are many possible attacks on a PC, but one of the most common is to examine/change the information on the hard disk. The only real way to defend against this is to use strong encryption on your hard disk. The entire hard disk needs to be encrypted, not just the contents of personal data which Microsoft's EFS provides. If you are using smartcards, then the disk should be encrypted using a key stored on the smartcard.

5.2 The Hewlett Packard D530

The Hewlett Packard D530 looks formidable. In addition to normal passwords, it uses a special keyboard with a smartcard reader. So you need the right card and password to be able to access the machine. It looks secure, but it is all a facade. In fact it is possible without a smartcard and without any password to gain full control of the machine.

Although there is a lock on the cover, it is easy to remove the cover by forcing it. One could then access the hard drive and the motherboard to change the BIOS password. Normally an attacker would want to access the BIOS password to change the setup so that he could boot from an external device (e.g. floppy, USB drive, CDROM). Remarkably, that is not necessary in the case of the 'hardened PC' – although the floppy is disabled - the PC can boot from either a CDROM or USB disk. This makes an attack very straightforward.

In any event, the security on the hardened PC is irrelevant. Anybody can run the election software on a normal PC. It is analogous to having a vault where the front door is locked, chained and monitored by armed guards but the back door is unmonitored and unlocked. You can make the locks on the front door even stronger, it does not change the fact that anybody can get in with impunity.

To make things even easier, the hardened PC has a writable CD, so that the election software can be copied and installed on another machine.

Although the BIOS has a setup password, you can view the current setup without the password. In addition, the hard disk password is not used.

5.3 Security Philosophy

The whole approach to security seems very haphazard. The keyboard and smartcards, use strong security techniques whereas the election software actually tells the user the password. The security of a system is as strong as the weakest link, which in this case means that there is effectively no security on the system. However, the ad hoc measures might make it look secure to a non expert. This is dangerous as the system will then be trusted by normal users. In addition, the fact that the election software will actually give a user the password, sends all the wrong signals about password use. If the system doesn't care about password security, why should a user care?

5.4 Sample Weaknesses of the system

1. It is not necessary to use the hardened PC to run the election. Simply connect another computer to the election hardware and you bypass the security in the hardened PC.

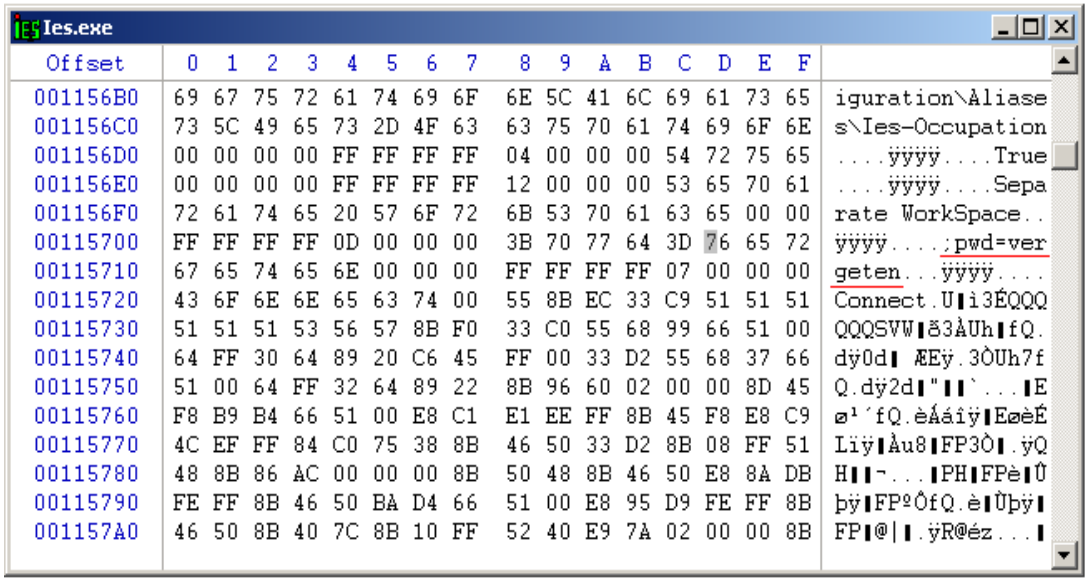
2. It is possible to boot from a CDROM or USB, thus bypassing the operating system. This would allow attackers to install/modify whatever data files or software that they wanted onto the system. Attackers could also copy whatever they wanted from the computer (see report below). In fact, using this technique, we were able to extract the administrator password which means we have full control of the machine without requiring the boot disk or smartcard.

3. The hard disk is not encrypted.

4. The PC box can be forced open allowing access to the BIOS and hard disk.

5. The Access 97 passwords used on the databases are not secure.

6. The database password is stored in plaintext in the election software (See fig 1). In fact there are 11 occurrences of the plaintext password in the executable.



Ies.exe

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
001156B0	69	67	75	72	61	74	69	6F	6E	5C	41	6C	69	61	73	65	iguration\Aliase
001156C0	73	5C	49	65	73	2D	4F	63	63	75	70	61	74	69	6F	6E	s\Ies-Occupation
001156D0	00	00	00	00	FF	FF	FF	FF	04	00	00	00	54	72	75	65	....ÿÿÿÿ....True
001156E0	00	00	00	00	FF	FF	FF	FF	12	00	00	00	53	65	70	61	....ÿÿÿÿ....Sepa
001156F0	72	61	74	65	20	57	6F	72	6B	53	70	61	63	65	00	00	rate WorkSpace..
00115700	FF	FF	FF	FF	0D	00	00	00	3B	70	77	64	3D	76	65	72	ÿÿÿÿ....;pwd=ver
00115710	67	65	74	65	6E	00	00	00	FF	FF	FF	FF	07	00	00	00	geten...ÿÿÿÿ....
00115720	43	6F	6E	6E	65	63	74	00	55	8B	EC	33	C9	51	51	51	Connect.U‹ì3ÉQQQ
00115730	51	51	51	53	56	57	8B	F0	33	C0	55	68	99	66	51	00	QQQSVW‹ð3ÀUh‹fQ.
00115740	64	FF	30	64	89	20	C6	45	FF	00	33	D2	55	68	37	66	dÿ0d‰ ÆEÿ.3ÒUh7f
00115750	51	00	64	FF	32	64	89	22	8B	96	60	02	00	00	8D	45	Q.dÿ2d‰"‹–`...ŤE
00115760	F8	B9	B4	66	51	00	E8	C1	E1	EE	FF	8B	45	F8	E8	C9	ø¹´fQ.èÁáîÿ‹Eøè
00115770	4C	EF	FF	84	C0	75	38	8B	46	50	33	D2	8B	08	FF	51	Liÿ„Àu8‹FP3Ò‹.ÿQ
00115780	48	8B	86	AC	00	00	00	8B	50	48	8B	46	50	E8	8A	DB	H‹†¬...‹PH‹FPè‹Û
00115790	FE	FF	8B	46	50	BA	D4	66	51	00	E8	95	D9	FE	FF	8B	þÿ‹FPºÔfQ.è•Ùþÿ‹
001157A0	46	50	8B	40	7C	8B	10	FF	52	40	E9	7A	02	00	00	8B	FP‹@|‹.ÿR@éz...‹

Fig. 1 Password stored in plaintext

7. The election software asks for a password. Pressing the help button reveals the password (fig 2). In any event, if the password was changed, it would be subject to a cracking attack as it is stored in the database.

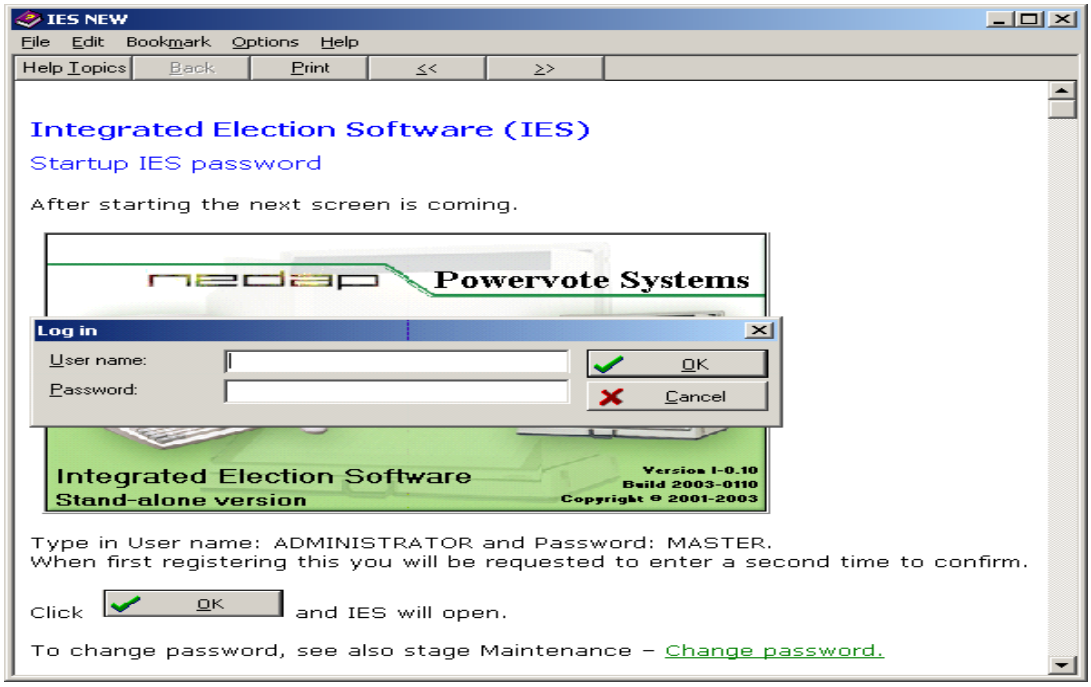


Fig. 2 Help page giving you the password

5.5 Details of a specific attack

Steps required to gain the administrator password on the hardened PC:

1. Boot the computer from CDROM into Windows 2000 (using either a specially prepared disk or a commercial product).

2. Replace spoolsv.exe with a program that will create an administrator account with a specific password.

3. Remove the CD and boot the system using the installed operating system. Log in using a normal account and spoolsv.exe will execute and create a new administrator account.

4. Log on using the newly created administrator account.

5. Tell windows to use the weaker Lan Manager hashing system. These are very easy to crack using standard cracking utilities.

6. Dump the password hashes and save them to CDROM or USB disk. (There are a number of free tools to enable you to access the password hashes.)

7. If you wish to cover some tracks, delete the newly created account and replace the spoolsv.exe file.

8. Back at base, run a password cracker on the collected password hashes. It required less than 24 hours to find the administrator password. A stronger password might have lasted longer but would probably have succumbed within a few days.

Most of the information required for this attack is freely available on the internet. An experienced attacker would require about 5 minutes to retrieve the password hashes. The hashes could then be copied to a USB drive or CDROM for offline cracking.

Note that administrators can log onto the computer without a smartcard. An administrator may also copy the contents of any smartcard to a file and may change the data on a smartcard.

Fig 3. Thumb disk

Other related attacks would take even less time. A bootable USB Thumb Disk could be inserted into one of the USB slots and, with appropriate software, could replace any of the system/database files on startup. This would merely require physical access to the machine for a few seconds and wouldn't even require touching the keyboard. The next time the computer booted up, it would run the attacker's program altering any desired files and could then boot up as normal. If desired, the attacker could force an immediate reboot by removing and reinserting the power cord.

This type of attack would be made considerably more difficult if the computer was set to only boot from the hard drive. However, it would still be vulnerable to having the CMOS changed, or having the hard disk removed.

References

[1] Microsoft's 10 Immutable Laws of Security
[http://www.microsoft.com/technet/archive/community/columns/security/essays/10imlaws.mspx]