RIES - Rijnland Internet Election System: a cursory study of published source code

Rop Gonggrijp, Willem-Jan Hengeveld, Eelco Hotting,
Sebastian Schmidt, and Frederik Weidemann

Wij vertrouwen stemcomputers niet
Linnaeusparkweg 98, 1098 EJ Amsterdam, The Netherlands
`rop@gonggri.jp`
`http://www.wijvertrouwenstemcomputersniet.nl`

Abstract. The Rijnland Internet Election System (RIES) is a system designed for voting in public elections over the internet. A rather cursory scan of the source code to RIES showed a significant lack of security-awareness among the programmers which – among other things – appears to have left RIES vulnerable to near-trivial attacks. If it had not been for independent studies finding problems, RIES would have been used in the 2008 Water Board elections, possibly handling a million votes or more. While RIES was more extensively studied to find cryptographic shortcomings, our work shows that more down–to–earth secure design practices can be at least as important, and the aspects need to be examined much sooner than right before an election.

Key words: electronic voting, internet voting, RIES, The Netherlands

1 Introduction

The Rijnland Internet Election System (RIES) processed around 90.000 votes in public elections in The Netherlands in 2004 and 2006. Based on total votes processed in public elections, RIES is one of the largest internet voting systems worldwide. As an interesting feature, RIES offers cryptographic end-to-end verifiability. This enables the voter to use cryptography to verify that her or his vote was counted as cast. After some delay, the source code to RIES was published[1] on June 24th 2008. This paper describes the result of a few days of looking at the source code and documentation of a rather complex internet voting system. This study began when the source code for RIES was published, on June 24th 2008. The first preliminary results of this study were available to the Dutch media and members of parliament four days later on June 28th. This paper can by no means be understood as an exhaustive study. Such a study would require much more time, study and effort.

[1] `http://www.openries.nl/downloads/broncode`

2 Permission

Verifying some of the problems we found in the source code on the actual system without permission from the people operating RIES would probably be prosecutable as computer crime. So in the early evening of Friday, June 27th we asked nicely and kindly got permission to attempt penetrating the RIES portal server at `https://portal.ries.surfnet.nl` from Simon Bouwman at "Het Waterschapshuis", a national cooperation of Water Boards that planned to operate the servers for the Water Board elections. He also kindly added one of our IP-numbers to the list of sites allowed to approach this server, a protection measure they were apparently just installing that very evening.

As a condition for getting permission, we accepted to print a brief reaction of "Het Waterschapshuis" along with our findings.

3 Sequence of Events

3.1 Water Boards

The Water Boards ("Waterschappen" in Dutch, sometimes also translated as "District Water Control Boards") are 27 different regional authorities dealing with water management in The Netherlands, a country that has long been highly dependent on a complex infrastructure of pumps and dykes to stay dry. Dating back to the 13th century, they are the oldest democratic structures in The Netherlands and among the oldest in the world. Since they are separate bodies of government the boards of these authorities are each directly elected by the people that live and/or own property in their area. In recent times these elections have been held by postal ballot, and in recent years turnout has been very low, often far below 30%.

RIES was developed by one of these Water Boards (Hoogheemraadschap van Rijnland) in conjunction with a number of private companies. It was used experimentally in the 2004 election by two of the Water Boards, and roughly 70 thousand voters cast their vote via the internet in that election.

3.2 2006 Parliamentary Election

RIES, in a version called RIES-KOA[2] was also used for the 2006 national parliamentary election as a complement to the postal voting available to Dutch citizens residing outside of the Netherlands. (The Netherlands do not offer postal voting to voters not residing abroad.) Roughly 20.000 votes were cast using RIES in that election. Because the Netherlands have proportional representation, these votes were added to the national totals for each candidate.

[2] KOA stands for "Kiezen Op Afstand" (Remote Voting), which is (was?) the Dutch government's remote e-voting project.

3.3 RIES-2008

A lot has happened with regard to electronic voting in The Netherlands in the past few years. The country was 100% e-Voting in 2006, and has since abondoned all electronic voting in polling stations. Our previous research [1] into the security of the Nedap system in use in 90% of the precincts played an important part in the decision making process. The use of RIES for these low-turnout Water Board elections would have made RIES the last remaining electronic voting system in use in The Netherlands.

The Water Boards would have liked to deploy RIES in its present incarnation, called RIES-2008, for the Water Board elections, which all took place simultaneously in November 2008. The ministry of Transport and Water Works had drafted legislation which, among many other things, also codified various procedures specific to the RIES system. Our foundation has lobbied with parliament to force rapid publication of the source code as well as for clear technical requirements and a procedure to formally test whether a proposed voting system meets these requirements. As a consequence the ministry created one and a half page of requirements [2] which for the greater part simply point to e-Voting recommendations issued by the Council of Europe [3].

The day after our preliminary findings were made available to the media and members of parliament, the ministry announced[3] that it would likely not approve RIES. As it turned out, the company hired to do the formal approval (Fox-IT) had also found very serious problems with RIES. They had concentrated on the underlying cryptography [4].

3.4 Post-2008

After the ministry's announcement, the Water Boards have expressed the intention to continue work on RIES for the 2012 elections. The Water Board elections of 2008 then ended in a fiasco. There were major problems with the postal ballot, mostly centered on the year of birth that people had to fill out coupled with the fact that the ballot forms were mixed inside households. Together with various other problems with scanning the ballots, this led to elections that saw more than 10% unintentionally spoiled ballots. Also, despite a massive advertising campaign and changes in the way the election was held, turnout remained very low, around 20%. After the election, the Water Boards have stated they would prefer not to hold direct elections again[4], making the future of this type of elections very unclear.

It is interesting to note that during the 2008 elections, RIES was also used for processing the paper ballots. Each ballot had a unique machine-readable code printed on it, and on one side of the process, files existed which coupled specific voters to this unique ID. On the other side of the process, there were files which

[3] Announcement to parliament: `http://wijvertrouwenstemcomputersniet.nl/images/9/98/Brief-VenW-geen-internetstemmen.pdf`

[4] Source: Volkskrant interview available at `http://www.volkskrant.nl/binnenland/article1100640.ece/Foutenmarge_bij_waterschapsverkiezingen_hoog`

coupled these IDs to votes. So though some of the vulnerabilities related to internet-voting were no longer in play, the vote secrecy related problems largely remained.

4 Known Security of RIES

4.1 Literature

The `www.openries.nl` website listed a large number of documents on RIES. On the page "What do others think?" we read (translated from Dutch):

Various prominent institutions have tested and positively evaluated RIES: TNO Human Factors from Soesterberg tested usability of the voting interface; A team of specialists from Peter Landrocks Cryptomathic (in Aarhus, Denmark) tested the cryptographic principles; Madison Gurka from Eindhoven tested the server and network setup and security; A team under supervision of Bart Jacobs (Radboud University Nijmegen) did external penetration tests.

Scientists [5–10] as well as other parties [11, 12] have looked into various aspects of the design and/or security of (parts of) RIES.

Apart from purely scientific work, RIES has been subject to an accessibility test, a browser compatibility test, a modules test, a disaster recovery test, a functional acceptance test, a chain test, a regression test, a risk analysis, a security and usage evaluation, a server audit, evaluations of the various elections held with it and a report [13] to see how RIES matches up to the 112 recommendations of the Council of Europe with regard to e-Voting [3] and many, many more studies and reports.

4.2 Theoretical Upper Bounds

RIES, as well as many other internet voting systems, can only be as secure as the weakest parts of the system. With RIES, this means that if an adversary can steal data from specific parts of the process, vote secrecy is compromised. Also, election integrity and vote secrecy with RIES depend on cryptographic functions not being manipulated or observed while running on individual voter PCs. Since these circumstances place clear upper bounds on the amount of real-world security that can be obtained, it is good to mention the largest fundamental threats in a little more detail.

4.3 Limited Security Against Insiders

Elections like the ones performed with RIES legally require secrecy of the vote. In RIES this requires the operators to destroy information they held at some stage during the process. If anyone manages to hold on to this information the publication of a verification file at the end of the election allows whoever has this information to tie every vote to an individual voter. Hubbers et al [7] conclude that the cryptography used in RIES offers no protection against insiders:

"RIES is built on certain cryptographic primitives, like one-time signatures. Keys for individual voters are generated centrally. There are no anonymous channels. The structural protection and safeguards offered by cryptography are therefore rather limited. Many of the guarantees in RIES thus rely on organizational controls, notably with respect to (voter) key generation, production of postal packages, insider attacks (especially at the server), integrity and authenticity of the software, and helpdesk procedures."

The CIBIT rapport [8] concludes (translated from Dutch): *"Vulnerability of the `STUF-C10` file, all temporary variants hereof and `KGenVoterKey`. Using the `STUF-C10` file one can influence the election and break vote secrecy. These objects need to be destroyed as soon as the necessity for the presence of these objects expires."*

Compared to a postal election performed in accordance with proper procedures, one must conclude that RIES created ways to surreptitiously violate vote secrecy on a scale never before possible. All that is needed for a massive breach of vote secrecy is a few people, or even a single individual, leaking critical files.

4.4 Household PCs Assumed Secure

Hubbers et al [7] also describe a central assumption during the design of RIES:

"RIES assumes that the voters PCs are secure. Attackers may however employ malware or even man–in–the–browser attacks to capture voters PCs. Powerful attackers may thus change votes, and so this involves a unique potential risk for Internet elections."

Given the prevalence of attacks against client PCs , for example with regard to electronic banking, it would seem inevitable for attacks to appear once electronic voting becomes common. The fact that candidates have apparently tried to submit faked signatures to be on the Water Boards candidate list in the past[5] proves that even for these election, there is already an apparent potential for fraud.

5 Security of RIES: a Look at the Code

We realise that previous researchers studying RIES had only design documents and not the source code to go by. So even though the source code had not been independently studied, the sheer amount of serious studies done on the security of RIES made us doubt if we would spot any problems, given the limited resources we had to study it. We were not able to perform a structured source code analysis, everything is this paper is based on a rather cursory look at the code. But even with our limited resources, we were able to spot quite a number of rather serious security concerns. We noticed a lack of input validation, creating verified opportunities for XSS and SQL injection, predictable random generation

[5] Hoogheemraadschap van Rijnland, Bestuursverkiezing Rijnland moet door fraude deels over, Persbericht 12 november 2004, `http://www.rijnlandkiest.net/asp/get.asp?xdl=../views/rijnlandkiest/xdl/Page&ItmIdt=00001440`

(for election management access tokens), hard coded values (for cryptographic keys, a CVS server, a mail server and an SMS gateway) as well as problems regarding exception handling.

5.1 Lack of input validation

XSS - Cross-Site Scripting There are locations in the code where information supplied by the user is passed back on the page that is output by the system. For example we can see[6]:

```
document.location="start.jsp?elid= \
   <%= request.getParameter("elid") %>";
```

as well as[7]:

```
<c:set var="section" \
  value="<%= request.getParameter("section")%>"/>
```

By supplying this parameter in a specially crafted URL, the user's browser could be made to execute Javascript statements within the context of the user's session on the election site. In the case of RIES, the cryptographic routines that perform the actual act of voting are implemented as client-side Javascript, making it impossible for users to protect themselves against such attacks by turning off Javascript.

We found out that we were not the only ones who had spotted this problem. As mentioned above, the RIES website lists an impressive number of studies into various aspects of the system. Among them is a report by GOVCERT.NL, the Dutch government computer emergency response team. They found this problem and reported it in September of 2006 when they did a 'web application scan' [14]. They found the same `elid` variable to be vulnerable, and recommended that the input and parameters be validated to eliminate the risk of Cross-Site Scripting. They also ominously said (translated from Dutch):

REMARK: The lack of sufficient input validation can also lead to vulnerabilities such as SQL-injection which are more serious in nature. During the scan we have not found any such vulnerability.

We are surprised that the makers of RIES present a two year old report of a scan on their website, apparently without having implemented the recommendations contained therein.

SQL Injection In 2006, GOVCERT.NL had warned RIES: if the programmer doesn't check the inputs to his/her code, the program may end up vulnerable to SQL injection attacks. During the interaction with the program, a user typically enters all sorts of text strings, such as her username when prompted. (See Fig. 1.)

[6] `riesvotewin_source_v1.0/admin/index.jsp`, line 29
[7] `riesvotewin_source_v1.0/admin/sectionlinks.jsp`, line 3

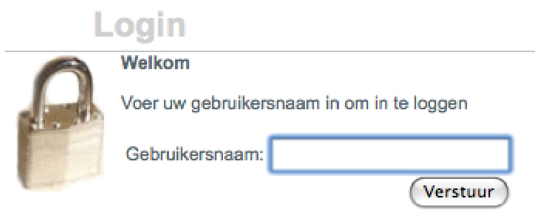


Fig. 1. Login screen

SQL queries involving user-supplied information in the RIES source code are all generated by simply inserting whatever the user entered into a query, without any checking. One of the queries that follows is the one where the code associated with the login box above tries to find the telephone number for a user to send a special SMS token to allow that user to log in[8]:

```
sbBuffer=new StringBuffer();
sbBuffer.append("select PHONE from OPERATOR where
  OPERATOR_ID='"+sUsername+"'");
oRs=oStmt.executeQuery(sbBuffer.toString());
```

As is visible from this code, the SQL statement to be processed by the database server is formed with the sUsername string. The code does not contain anything to sanitize that string first. If one enters `rop` in the username box the query to the SQL server would become:

```
select PHONE from OPERATOR where OPERATOR_ID='rop'
```

Since the program finds no corresponding entry in the `OPERATOR_ID` table it outputs ”login failed” as shown in Fig. 2.

However, we can make the SQL statement succeed by logging in with a string as shown in Fig. 3.

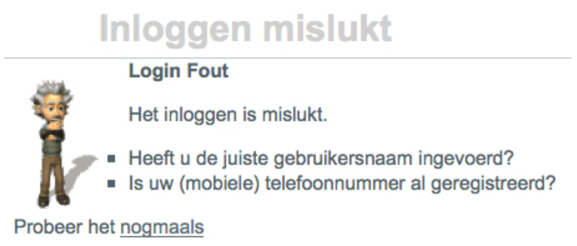
The resulting SQL statement now looks like:

```
select PHONE from OPERATOR where OPERATOR_ID='rop' OR 1=1; --'
```

And as a result we now get the output as shown in Fig. 4. The system has apparently sent the special not–so–random access code via SMS. Since the SQL statement succeeded on the first user in the user table, we suspect this user will have received such an SMS.

To actually enter the system and prove further vulnerabilities, one needs to play around a little more. We were still experimenting with this when the system

[8] `riesportal_source_v1.0.zip/WEB-INF/src/java/org/openries/portal/jaas/JAASHelperServlet.java`, line 347



Inloggen mislukt

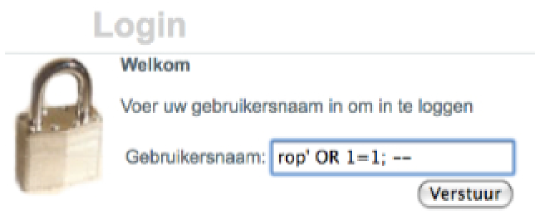
Login Fout

Het inloggen is mislukt.

- Heeft u de juiste gebruikersnaam ingevoerd?
- Is uw (mobiele) telefoonnummer al geregistreerd?

Probeer het nogmaals

Fig. 2. Login failed



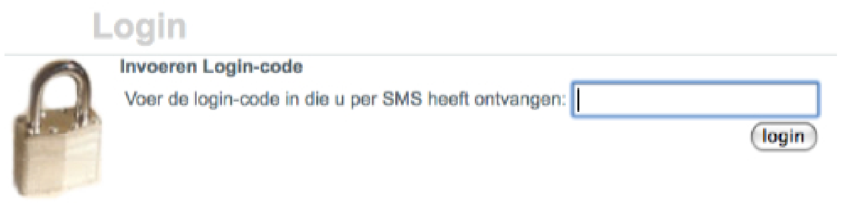
Login

Welkom

Voer uw gebruikersnaam in om in te loggen

Gebruikersnaam: rop' OR 1=1; --

Fig. 3. SQL injection



Login

Invoeren Login-code

Voer de login-code in die u per SMS heeft ontvangen:

Fig. 4. Please enter SMS code

suddenly said: *"Service Unavailable"*. A few minutes later it said *"Technical Problem"*, and then it finally said *"Closed For The Weekend"*. We guess that since we were testing on a Friday night, indeed the system could be down for the weekend. It did however briefly re-open on the following Saturday, but after a few more carefully crafted attempts it was again *"Closed For The Weekend"*.

5.2 Errors/problems in code

Predictable Random Tokens The code contains a method of authenticating a user via her/his mobile phone. The code calls this challenge/response, however it is technically a response only. When a user wants to log in, the system generates a random password which is sent to the user via SMS. The user must than enter this password via the internet. Below is the piece of code[9] that does the actual generation of that 'random' token:

```
Random rGen=new Random(new Date().getTime());
String sResult="";
int i=0;
while(i<6) {
    sResult+=rGen.nextInt(10);
    i++;
}
```

`Random()` will present the same output when given the same output of `Date()` and `getTime()`. Even though the latter is in milliseconds, an attacker would only need a few thousand guesses to figure out the key sent to a phone that she does not own. Since the code does not prevent someone from trying a few thousand tokens, this would not prevent the attacker from gaining access.

Note that an attacker can probably acquire a very accurate idea of system time from the http headers provided by the system or, if the goal is privilege escalation, from the token received by SMS following a valid login.

Insufficient Exception Handling Often when exceptions are handled in the code, a message is logged, but no action is taken. For example, in `_sendResponseSMS`[10], the exception handlers are (in pseudo-code) very often structured like the one in sendResponseSMS:

```
_sendResponseSMS()
{
  try {
    executeQuery
```

[9] `riesportal_source_v1.0.zip/WEB-INF/src/java/org/openries/portal/jaas/JAASHelperServlet.java`, line 280

[10] `riesportal_source_v1.0.zip/WEB-INF/src/java/org/openries/portal/jaas/JAASHelperServlet.java`, line 332

```
    if result {
      try { sendsms }
      catch(all) { return false }
    }
    else { return false }
  }
  catch (sqlexception) { logmessage }

  return true
}
```

Since the code contains no `return false` with the `catch(sqlexception)`, an exception from the `executeQuery` will still cause the calling function `sendResponseSMS` to succeed and cause the server to display the 'enter-SMS-response' page. One of the possible reasons why `executeQuery` would throw an sqlexception would be a syntax error in the SQL statement, for instance, caused by an SQL-injection attempt. But given that SQL syntax errors are ignored, an attacker can carry his manipulations beyond the point where the program should abort. This makes constructing more complex SQL-injections significantly easier. An attacker can now target specific SQL statements, without the need to keep all the other statements free of errors.

The problem is that RIES tries to trap exceptions from the library functions, and translate them in true/false but often fails to do a `return false`. In the case of the SQL exceptions, it might be even better not to trap these exceptions at all, but let the JSP server handle this. In most cases an exception should be a reason to abort any pending operations, not to 'log message and continue'.

5.3 Code mixed with configuration information

Test Key Then there's a part in `org.openries.ripocs.config.ConfigManager` where the code is apparently retrieving a stored 'salt' value from a file to create, through XOR, a smartcard key of some sort. The final lines of the code[11] that is supposed to be generating the value are:

```
// derive AbelPiKey (16 bytes)
return PKCS5.PBKDF1(sPassPhrase, abSalt, PKCS5_ITERATIONS, 16);
```

However, the entire function is commented out and instead it now reads:

```
//temp for ketentest 1 because of existing keys in smartcards
return Utils.stringToHex
  ("0123456789ABCDEF FFFFFFFFFFFFFFFF0123456789ABCDEF");
```

By interweaving (highly dangerous) testing code and production code in this way, the designers are waiting for accidents to happen. If this should be done at all, it needs to be done with `#IFDEF`s or other more suitable mechanisms.

[11] `riesripocs_source_v1.0.zip/src/org/openries/ripocs/config/ConfigManager.java`, line 23

CVS Server The code exposes a development CVS server that appears to be running on a regular home ADSL connection[12]:

```
:pserver:arnout@cozmanova.xs4all.nl:4202/usr/local/cvs-ries-rep
```

Mail Server Also, a hardcoded public mail server is used[13]:

```
private static String EMAIL_SERVER = "smtp.xs4all.nl";
```

It is not clear under what circumstances the system sends mail and whether one could perform attacks if one could destroy, intercept, modify or interject batches of these e-mails. The concept of 'phishing' for voting credentials comes to mind.

SMS gateway The code also contains an SMS gateway with a valid account[14]:

```
private String _sServiceURL="https://secure.mollie.nl/xml/sms/";
private String _sUsername="mdobrinic";
private String _sPassword="riesdemo";
private String _sGateway="2";
  // development default; 1=more reliable; 2=cheaper
```

Assuming the authors want their voting system to be 'more reliable' as opposed to 'cheaper', the setting of the `sGateway` value shows how easy it is for undesirable development artifacts to make it into production code, especially if the programmer doesn't separate code from configuration.

5.4 Current Status Regarding Fixing These Problems

The URL supplied to download RIES still points to an overview page, but the links to the actual source code zip files give Microsoft IIS "The page cannot be found" error messages. There appears to be no way to verify progress, if any, on fixing these problems.

6 Conclusions

These are all issues stemming from general poor code quality and a lack of secure design principles. The cross-site scripting problem, the SQL injection problem and the token generation problem are especially serious problems that could, each in and by itself, lead to compromises of the entire system. When the RIES

[12] `rieslogin_source_v1.0.zip/org/CVS/Root`

[13] `riessystem_source_v1.0.zip/source/org/openries/system/messaging/EmailMessage.java`, line 52

[14] `riessystem_source_v1.0.zip/source/org/openries/system/messaging/SMSConfig.java`, line 22

portal is compromised, all election management functions can be performed by the attackers and all data passed through the portal can be destroyed or manipulated, clearly placing election integrity at risk.

Computer security appears not to be part of the mindset of the people programming RIES. For example, in the case of the SQL-problems, it would have been better to use `prepareStatement` in addition to sanitizing user input. Normal best-practice secure design principles prevent most if not all of these problems from occurring. All problems can be repaired relatively quickly. However, given the general questions this quick study raises about overall code quality in the RIES project, such fixes would by no means yield a secure system. We plucked the low-hanging fruit in the part of the system which was facing the internet, many other parts of the code have not yet been studied.

We were amazed to find a system so apparently well-studied yet so fundamentally and undeniably insecurely programmed. This is not so much criticism of the people studying before us; it mostly shows how little one can say about security of a system without access to source code. Scientific studies of RIES seem to have concentrated on the more scientifically 'sexy' theoretical security offered by the inventive cryptographic protocols. Only source code review can efficiently examine some of the threats posed by very straightforward and down–to–earth attack methods that are much more likely to be used in the real world.

The scope of this paper is not to completely understand the RIES system but only to outline a number of immediately-visible security problems. As a reality check, we are happy to have proven that SQL injection actually works on the live system. Further examination of RIES, including actually attempting to disturb/manipulate elections would likely require further study of the inner workings of RIES and is beyond the scope of this first examination.

Given what we found in the scope of this quick study, it is very worrisome that a previous version of RIES has actually been used in the context of a real-world parliamentary election. If society decides to go ahead with internet voting (thus implicitly deciding that the advantages of remote e-voting outweigh more fundamental problems with vote secrecy and lack of transparency/observability) it is clear that more attention should be given to secure programming. Internet voting suppliers seem reluctant to allow independent study of their source code. The Water Boards had to be forced by the Dutch Parliament to open the source code, and code review has also been controversial[15] in the recent Austrian national student body elections where an internet voting system made by voting technology supplier Scytl was used. Our findings clearly show that society cannot afford to merely study the outside-world interfaces of an internet voting system, even if there are "nice" cryptographic tricks involved.

To create a system that appears secure, there are two approaches. The proper approach is to design a system with security in mind. The other approach is to retrofit an insecure system with security-measures that make a system look secure but which in fact add little security. Such measures are usually intended

[15] Source: description of source code review procedures at `http://papierwahl.at/2009/05/19/details-der-sourcecode-einsicht`

to impress onlookers. There are situations where adding an SMS-token is a useful addition to other security measures. However in combination with the proven lack of security awareness during the implementation of the system, RIES' SMS-token appears to fall in the impress-the-onlookers category.

The `www.openries.nl` site says: *"Various prominent institutions have tested and positively evaluated RIES"*. This research shows that one must be very wary if scientific and other studies into some part of a not yet published and changing system are used to implicitly claim the entire system is secure. No amount of voluntary studies of some part of a voting system - often paid for by stakeholders wanting to see the system in use - can ever replace clear and stringent government-imposed requirements that include independent source code review. Such reviews need to pragmatically and 'holistically' look for security problems as well as test the code against more formal coding standards and practices.

In their 2004 article 'Stemmen via Internet geen probleem' [6] Hubbers and Jacobs "vote in favour" of use of this system when they state (translated from Dutch): *"Summarizing, [RIES] is a relatively simple, original and understandable system that has been implemented with the appropriate care and transparency. [...] When the use of RIES during these Water Board elections is involved, we clearly vote in favour!"*. We pose that RIES has quite clearly not been implemented with "appropriate care". Given the dependence of society on their judgment, scientists should probably refrain from endorsing electronic voting systems until the entire system is open for public examination and at least until they have seen independent studies of all parts of the systems involved.

Despite obvious code quality and apparent quality management problems, the Water Boards need to be commended for the publication of the source code and documentation as well as for allowing outside researchers to study the security of the system. It seems that they are at least trying to do the right thing.

The amount of problems we found, as well as the class of problems, imply that If RIES were ever to be used again in elections, it must undergo much more testing and quite possibly partial code rewrites. Use of RIES in real-world elections without allowing independent source code review was, in retrospect, irresponsible. The attempt by the Water Boards to use RIES in the 2008 election even after our findings were known to them shows how deep government can become entrenched once the e-Voting train is in motion. Recent events in The Netherlands and Germany seem to indicate that government bodies are over-reliant on information provided to them voting technology providers. The Dutch government commission that studied past decision-making with regard to electronic voting stated in its report [14] (translated from conclusions on p. 51): *"Decennia of trusting the information provided by suppliers Nedap, Sdu and certivication agency TNO has placed the ministry at a disadvantage."*

The Dutch government had to be forced by a majority in Parliament to develop any standards at all for internet voting. The resulting half-hearted and minimalist legal requirements [2] (covering a whole page and a half) or the recommendations of the Council of Europe [3] that these requirements point to contain no provisions that would have prevented any of the problems we found.

7 Acknowledgements

The authors would like to thank Simon Bouwman of "Het Waterschapshuis" for giving permission to test some of our attacks against the system. Furthermore they would like to thank the entire crew of "Wij vertrouwen stemcomputers niet" and Zenon Panoussis for their insight and for helping in proofreading this paper.

8 Reaction Het Waterschapshuis

As stated in the introduction, we agreed with "Het Waterschapshuis" to include a brief reaction with our findings. They responded as follows:

"An advantage of open source is that the code can be reviewed and improvements of the code can be made. The published code of RIES is not yet the production code. Internal reviews and tests have to be made. Recommendations from external - as in this paper - are welcome. New versions of the code will be published in the coming weeks.

As mentioned in the paper, RIES is a rather complex (internet) voting system. RIES contains several sub-systems. Each sub-system is a combination of software, configuration and administrative procedures. Each sub-system has very different tasks and settings and also different security requirements. For instance, the VoteWindow is the only sub-system which is public to the world-wide Internet. All other sub-systems are limited access only, not accessible through the internet. The RIES-Portal access will be controlled by VPN, RIES-RIPOCS is only accessible via RIES-Portal, and RIES-ROCMIS is an offline machine used within a proper set of administrative procedures. Therefore, RIES cannot be evaluated from source code alone to measure the security strength.

Keep in mind; this part is NOT production code yet. Many of the issues are related to proper input validation. And we agree that proper input validation is required and we will fulfill that requirement. Mainly Struts input validation mechanism will be used. In the published source packages and the development system investigated, the feature was not switched on for development reasons. Therefore again: were in a state of functional sequence test (ketentest) and not yet in production."

The original response was slightly longer and added a list where each issue we found was discussed separately. Since this was a little too long to be included in this paper, we agreed to include a link to the full response, which will be available on the RIES website at `http://www.openries.nl/wvsn-paper` .

There is a lot to be said regarding this reaction, but that would turn this paper into a discussion forum. Suffice it to say that we stand by our original conclusions and recommendations. Despite RIES not being used in 2008, the debate on whether or not RIES is suitable for public elections may well continue at some point in the future.

References

[1] Gonggrijp, R., Hengeveld, W.-J.: Studying the Nedap/Groenendaal ES3B voting computer, a computer security perspective. Proceedings of the USENIX/Accurate Electronic Voting Technology workshop (2007).

[2] Ministerie van Verkeer en Waterstaat: Regeling waterschapsverkiezingen 2008. 15 mei 2008/Nr. CEND/HDJZ-2008/587, Staatscourant 23 mei 2008, nr. 97 / pag. 11. `http://www.wijvertrouwenstemcomputersniet.nl/images/e/e7/SC85731.pdf`

[3] Council of Europe: Recommendation Rec(2004)11 of the Committee of Ministers to member states on legal, operational and technical standards for e-voting (2004). `https://wcd.coe.int/ViewDoc.jsp?id=778189`

[4] Gedrojc, B., Hueck, M., Hoogstraten, H., Koek, M., Resink, S.: Rapportage Fox-IT - Advisering toelaatbaarheid internetstemvoorziening waterschappen (2008) `http://www.verkeerenwaterstaat.nl/Images/20081302%20Bijlage%201%20Rapport_tcm195-228336.pdf`

[5] Hubbers, E.-M., Jacobs, B. Pieters, W.: RIES - Internet Voting in Action. In R. Bilof, editor, COMPSAC'05, Proceedings of the 29th Annual International Computer Software and Applications Conference, COMPSAC'05, pages 417-424. IEEE Computer Society, 2005. 26-28 July 2005. `http://www.cs.ru.nl/~hubbers/pubs/compsac2005.pdf`

[6] Hubbers, E.-M., Jacobs, B.: Stemmen via internet geen probleem.Automatisering Gids #42, 15 October 2004, p.15. `http://www.openries.nl/aspx/download.aspx?File=/contents/pages/77743/stemmenviainternetgeenprobleem.pdf`

[7] Hubbers, E., Jacobs, B., Schoenmakers, B., Van Tilborg, H, De Weger, B.: Description and Analysis of the RIES Internet Voting System 24 June 2008. `http://www.win.tue.nl/eipsi/images/RIES_descr_anal_v1.0_June_24.pdf`

[8] Van Ekris, J.: CIBIT, Beoordeling KOA, Een beoordeling van de integriteit van "Kiezen op Afstand", 11 September 2008. `http://www.openries.nl/aspx/download.aspx?File=/contents/pages/77743/eindrapportcibit.pdf`

[9] Nijmegen University - Security of Systems:?Server Audit van RIES, 23 July 2004. `http://www.openries.nl/aspx/download.aspx?File=/contents/pages/77743/reportkun.pdf`

[10] Hugo Jonker and Melanie Volkamer, Compliance of RIES to the proposed e-Voting protection profile, VOTE-ID 2007

[11] Jens Groth, CryptoMathic: Review of RIES (v 0.3), Cryptomathic A/S, 21 January 2004. `http://www.openries.nl/aspx/download.aspx?File=/contents/pages/77743/reviewofries.pdf`

[12] Lucas Kruijswijk: Internetstemmen met RIES onder de loep (2006). `http://www.wijvertrouwenstemcomputersniet.nl/Internetstemmen_met_RIES_onder_de_loep`

[13] Unie van Waterschappen: Aanbevelingen van de Raad van Europa, Evaluatie voorziening internetstemmen RIES, conform artikel 5 onderdeel b Regeling waterschaps-verkiezingen 2008, version 6, June 2008. `http://www.openries.nl/aspx/download.aspx?File=/contents/pages/77726/evaluatieaanbevelingenraadvaneuropa.pdf`

[14] GOVCERT.NL: Webapplicatie-scan, Kiezen op Afstand, 1 September 2006 `http://www.openries.nl/aspx/download.aspx?File=/contents/pages/77743/webapplicatie-scan.pdf`

[15] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties: Stemmachines, een verweesd dossier, 17 april 2007 `http://www.minbzk.nl/contents/pages/86914/rapportstemmachineseenverweesddossier.pdf`