On the use of Internet Voting on Compromised Computers

Philippe Beaucamps, Daniel Reynaud-Plantey, Jean-Yves Marion
Equipe CARTE - LORIA
Campus Scientifique - BP 239
54506 Vandoeuvre-lès-Nancy Cedex (France)
{beaucphi,reynaudd,marionjy}_at_loria.fr

Eric Filiol
Army Signals Academy Virology and Cryptology Laboratory, Rennes (France)
eric.filiol@esat.terre.defense.gouv.fr

October 21, 2008

Abstract

Internet voting is the process of letting voters cast their vote over the Internet, at home or on public computers. It is a way to reduce the cost associated with elections and to obtain higher participation, but it also raises important security problems. In this paper, we study shortcomings related to this technology, and more particularly shortcomings due to the presence of dedicated malware on the voters' computers. Common literature usually focuses only on designing a secure voting protocol, either discarding the malware issue or proposing prohibitive solutions, such as the use of dedicated hardware. However the purpose of Internet voting is precisely to allow *anyone* to vote from home, making the use of dedicated hardware a non conceivable solution. Therefore, we analyse the reliability of possibly malware-infected mainstream computers. Specifically, we do not consider the security of the voting protocol but define the data available to the malware and the attacks that can be carried out thereby. We show that current Internet voting implementations are vulnerable to these attacks, due to weak or irrelevant security measures. Thus we describe *reasonable* solutions that aim at coping with the lack of security of current implementations on general-purpose computers, even though some attacks cannot be prevented but can only be mitigated. For example, it is impossible to prevent the malware from stealing the user credentials with no hardware support, but it is easy to design a system in which user credentials are useless to an attacker: therefore we can prevent more serious attacks such as automatic vote changing and voter impersonation. Among other solutions, we describe and study reliability of hybrid voting mechanisms, using a medium which can not be accessed by the malware, as well as of Human Interaction Proof implementations to prevent automatic vote changing and the election invalidation that could result from this class of attacks.

1 Introduction

Internet and electronic voting (e-voting) are used increasingly by businesses and organisations for their elections (Mohen and Glidden, 2001), and some countries such as the United States, Switzerland, United Kingdom and Estonia even start using it for local or national elections (Mohen and Glidden, 2001; Hensler, 2004; Wearden, 2002; Estonian Electoral Committee, n.d.). Electronic voting eases the counting of votes and possibly reliance of this counting. Internet voting is a form of electronic voting that uses Internet to let voters vote from their home or from close dedicated places. It aims to make voting simpler, possibly more accessible and thereby to increase voter turnout. However security and certification of such systems is questionned or proved to be inappropriate for democratic elections scenarios (Enguehard, 2008). In the specific case of Internet voting, the security of this process is difficult to ensure at different levels: the correctness of the client and server voting software, resistance to man-in-the-middle attacks, anonymity and so on (Phillips and von Spakovsky, 2001). Here we focus on the security of the client computer used to pass a vote. We claim that any current Internet voting process can be altered by compromised computers in some way.

Although the "virus and trojan threats" are usually considered an acceptable risk in studies on Internet voting (California Internet Voting Task Force, 2000), we try to show how easy it is to exploit the current Internet voting technologies. By merely using a browser addon, one can alter a vote — randomly or in

favour of some candidate — or deny voting. We describe such an addon which takes about 20 to 30 lines of Javascript code to transparently alter a vote. Different attacks can be carried out, that we describe in the article. Cheating is of course the most typical purpose of such attacks. However, in an information warfare perspective, one can imagine other motivations, such as a foreign country trying to tamper with an election, for political or economical purpose. Totalitarian regimes may also use such attacks to alter or spy on the voters' choices, in a pseudo-democratic election. Or hacker communities might even use such attacks to invalidate elections they believe to be rigged. These attacks will typically rely on email infection, using botnets[1] to optimize the infection process. Our proof of concept malicious addon shows how such malware could easily tamper with a voting process. Moreover, by relying on a botnet or simply on browser addons, one can easily upgrade the malware for future election processes.

There already exists a very rich literature that deals with securing voting protocols, so we are mainly concerned with the protection against targeted malware while not altering the technology accessibility and simplicity. In particular, non-critical voting scenarios rarely implement strong security measures or these very measures are simply not an option. Therefore we are also interested in securing as efficiently as possible such scenarios that are required to rely solely on Internet.

We describe how the security of common Internet voting scenarios can be jeopardized by a simple browser oriented malware. We also survey current solutions, though they usually are not concerned with computer compromission, and we study whether they can improve the reliability of voting machine. We finally describe specific solutions to keep Internet voting systems as simple and accessible as possible for the user, and to mitigate the inherent lack of security and the implications on the validity of the election.

2 Security of the client computer

2.1 Definitions

In this article, we use the expression "Internet voting" as a reference to the use of general-purpose computers with no particular hardware on the client side during the process of an election, as opposed to e-voting which encompasses both Internet voting and the use of dedicated voting machines.

The term "malware" refers to any sort of program with unknown or unwanted functionality, such as viruses, rootkits, spyware, keyloggers or even legitimate software with exploitable backdoors.

The term "compromised" refers to the state of any computer infected with malware. We consider that the malware has complete access to the user interface of the voting system, it can capture and modify the user inputs as well as capture and modify network data. We will show that these assumptions are realistic, even for high-level malware that does not have administrative rights and therefore no complete control of the machine.

2.2 Specific attacks

The design of a secure process for elections using automatic devices is a difficult one, as many requirements have to be met at different levels (Safevote, 2000). However, this paper focuses only on the specific attacks that a malware can undertake during the vote, it does not address protocol flaws such as vote buying, coercion, cryptographic or verifyability issues. When a compromised computer is used, we consider the following attacks:

- *Automatic vote changing:* the most powerful attack, allowing the malware to change the expressed vote on the fly without the user noticing it.

- *Random vote changing:* a sort of denial of service attack, by changing the expressed vote on the fly to a different value without the user noticing it.

- *Vote dropping:* again a denial of service attack, by letting the user think his/her vote has been counted when in fact no vote has been cast.

- *Voter impersonation:* by capturing the user credentials and then sending them to the attacker, it might be possible to steal votes. This is probably one of the hardest attacks to avoid, because it is the less intrusive and involves a human attacker.

- *Vote disclosure:* by logging the user identity and the expressed vote, it might be possible to disclose publicly the vote. Since it is a passive attack it is hard to detect and easy to implement (compared to the other attacks) and can have serious consequences on a democratic vote.

---

[1]expanding malware networks that are remotely controlled and upgraded

We do not mention *voting denial*, that is preventing the user from voting by making the service unreachable, as it is the easiest, most obvious and most visible attack. We're mostly interested in *stealthy attacks*, which are the most common ones, though voting denial is not to be disregarded.

Depending on the election protocol and on the software implementation, Internet voting systems can be vulnerable to all or just some of the attacks defined above. We can now restate our claim in more accurate terms: any current Internet voting technology relying only on general-purpose computers (i.e. that do not use specific hardware devices) and that do not use side-channels (such as paper or any other medium not accessible by the malware) are vulnerable to at least one of the attacks mentioned above.

2.3   Example: a naive but realistic Internet voting system

The most straightforward implementation of an Internet voting system that comes to mind is to:

1. setup a web server, accessible over HTTPS;

2. let voters log in via a server-side script;

3. let voters input their choice using an HTML form, a Java applet, an ActiveX component or any other client-side executable;

4. confirm the choice and send it to the server (the communication being encrypted with SSL).

This implementation is probably the simplest one and the only security is the encryption between the browser and the web server. However it is realistic, since it has been encountered in commercial voting systems in 2008 for professional elections.

To demonstrate the inherent problems with this simple architecture, we have developed a client-side malware using a Mozilla Firefox extension. Security issues arising from the use of malicious Firefox extensions was recently detailed in (Louw et al., 2008) and (Beaucamps and Reynaud, 2008), and exposed in (Mozilla Security Blog, 2008). We published an attack scenario in June 2008, related to the use of such malicious extensions to tamper with an Internet voting process. Our proof-of-concept malicious Firefox extension uses techniques that are similar in essence to those used in banker malware for instance, such as Anserin for Microsoft Internet Explorer (Charpentier and Hamon, 2008). In particular, though we used Mozilla Firefox, please note that this malware can be easily adapted to work with current versions of Internet Explorer.

To take control of the election process, the malware just has to complete the following steps :

1. hook the "page loaded" event in the browser (if this is not possible due to lack of APIs or security restrictions, regularly scan the opened URLs)

2. if the current URL corresponds to the address of a targeted election, go to step 3., otherwise go back to 1.

3. modify the webpage before it is displayed (in order to perform the random vote changing attack) or steal the user credentials (in order to perform the voter impersonation or vote disclosure attacks)

4. modify the user input before it is sent to the server (in order to perform the automatic vote changing, random vote changing or vote dropping attacks)

5. modify the confirmation webpage so that the user does not see that an attack has been performed

Here are code excerpts from the proof-of-concept Mozilla Firefox extension showing how the above steps can be completed in Javascript :

```
// 1. hook the DOMContentLoaded event
var appcontent = window.document
                    .getElementById("appcontent"); // browser
if (appcontent) {
  appcontent.addEventListener(
      "DOMContentLoaded",
      VOTE4U.contentLoaded,
      false);
}
```

```
// 2. check if the current page corresponds to a target page
//    and behave accordingly
var urlbar = document.getElementById('urlbar');
this.target = 0;

if (urlbar.value.match(//*SOME_REGULAR_EXPRESSION*//)) {
  this.target = 1; // the current page is the voting page
  // hook the 'submit' event:
  content.document.getElementById("form1")
    .addEventListener (
        "submit",
        VOTE4U.performPayload,
        false);
}

else if (urlbar.value.match(//*SOME_OTHER_REGULAR_EXPRESSION*//)) {
  this.target = 2; // the current page is the confirmation page
  // go to step 4.
}
```

```
// 3. perform payload
// the voting page contains an HTML form named form1
// the candidates are chosen in a radio field named FIELD1

var form1 = content.document.getElementById("form1");
// get candidates list:
var results = VOTE4U.evaluateXPath (
                form1.wrappedJSObject,
                "//input[@name='FIELD1']");
// find the index of the cheating candidate:
var cand_idx = VOTE4U.lookForCandidate(
                results,
                "NameOfCheater");
// lookForCandidate also sets VOTE4U.originalVote,
// corresponding to the vote cast by the user,
// and changes the form name of FIELD1 to discard the user choice.

// adding the (hidden) bogus vote to the form
var new_f1 = content.document.createElement("input");
new_f1.setAttribute("type", "hidden");
new_f1.setAttribute("name", "FIELD1");
new_f1.value = results[cand_idx].value;
form1.appendChild(new_f1);
```

```
// 4. modify the confirmation page
// after the modification, the user will submit the page himself,
// thinking that he's voting for his original choice, when in fact
// he's confirming that he wants to vote for the cheater

table.insertRow(X).innerHTML =
  "<td width=\"5%\"> </td><td>You have voted for: " +
```

```
        VOTE4U. originalVote +
        ". By clicking the submit button, you will confirm your choice." +
        "</td>";
```

This attack implementation requires almost no effort, and totally bypasses the SSL encryption since all computations are made before data encryption and communication. It is also almost not intrusive, that is to say it performs no noisy or sensitive operation (from the security point-of-view): it does not issue system calls, and it does not perform file or network operations. The only requirements are the ability to monitor webpages in real time and to modify their DOM. As a consequence, even tight security policies are likely to permit these operations.

To design a voting process secure against compromised clients, one should bear in mind the above example: every client computer is potentially compromised, and the malware can monitor every step of the election, it can capture and modify user data before it is sent. Workarounds are proposed for this scenario in section 3.

3 Solutions

We argue the only reliably secure solution in the context of remote Internet voting is a hybrid one as in (Chaum, 2001) and (Oppliger, 2002), that is using some other medium that is unavailable to a computer malware. However, these solutions can be costly so we review previously suggested solutions and propose other solutions that aim at making vote changing harder. Also we do not consider "security through obscurity" solutions to be reliable in the long term, even though they allow for a partial practical security during a short period. In particular, assuming such a solution is eventually reverse-engineered, privacy issues then arise in case when voting data has been collected for later decryption. We first describe commonly proposed solutions found in the literature and then describe solutions that are more tailored to Internet voting scenarios, taking into account possible limitations.

3.1 Secure voting places

In cases where home computers are not trusted for Internet voting (or remote Internet voting), alternative solutions might be considered, such as poll-site Internet voting (California Internet Voting Task Force, 2000) if the vote takes place on a computer and the vote is cast over the Internet at an official polling station. The security of the computer is supposed to be ensured by the physical presence of poll workers. Of course, it does not solve the problem of the client security and does not address one of the problems that Internet voting is supposed to solve: voter participation.

A similar solution is sometimes referred to as kiosk voting, it is the same as poll-site Internet voting except the vote takes place on dedicated computers in public places, such as schools and shopping malls. This might address the problem of voter participation but it is probably the worst solution from the client security perspective. Ensuring the security of a home computer is a hard problem, but ensuring the security of a public computer is virtually impossible.

3.2 Special security devices

Special devices may be considered, especially smart card readers, because it is supposedly hard or impossible to tamper with smart cards without destroying them (Joaquim and Ribeiro, 2007). Depending on the implementation and the use of the smart card, some attacks may be avoided, but the situation is the same as with cryptography: incorporating it does not mean you are secure.

A good solution might be to use more sophisticated security devices, with user input capabilities (such as a keyboard or touch pad), user output such as a screen and enough resources to perform cryptographic computations without the help of the client processor (Zùquete et al., 2007). This solution is theoretically secure with regards to the vulnerabilities defined in this paper, because the client computer is no longer used to interact with the user but rather is turned into a communication channel. However, it seems to be hard to achieve in practice due to the cost associated with special devices and to the fact that Internet voting is precisely supposed to allow more people to vote in an easier way (with as few requirements as possible).

3.3 Trusted computing

The idea of Trusted Computing is to solve the insecure platform problem by certifying that the client platform is running a trusted operated system and the unmodified voting software with the help of a a hardware chip called the Trusted Platform Module. According to (Volkamer et al., 2006) and (Rubin, 2002), a trusted computing solution ensures the existence of a trusted path between the user and the voting software, so that no malware can capture the user input or modify it.

Due to the practical difficulty of remote attestation, we believe that it is not a viable short-term solution but Trusted Computing may be efficient with a dedicated virtual machine (Joaquim and Ribeiro, 2007).

3.4 Hybrid solution

This solution was already suggested by David Chaum in 2001 with his SureVote system (Chaum, 2001). It relies on having the voter use some medium that is not connected to his voting computer. The most obvious medium would be sending a letter containing codes for the candidate. These codes must vary from one voter to another, and only the back-end system should then know, when receiving the vote, which candidate was chosen. A last step is to send the voter a code that would acknowledge his vote. This code must match a verification code in his original letter.

Such a system protects the voting process against all previous attacks. The set of valid codes should be large enough to prevent random vote changing, and only a vote should not be iterated more than a predefined number of times (in case a malware tries several codes until it's got confirmation).

3.5 Using a known hard problem

There might be some scenarios where using a secondary medium is difficult. In that case, we can at least harden the voting process, since the attacker can potentially modify the webpage contents on the fly (ie alter the voting process and modify the vote). We rely on problems that are known to be hard to solve by automated processes but are easily solved by humans. Such problems are sometimes referred to as Turing tests that allow to distinguish between a human and a machine. For instance, CAPTCHA images ("Completely Automated Public Turing test to Tell Computers and Humans Apart") (von Ahn et al., 2004, 2003) are images with text embedded in a confused way in it: humans are supposed to be able to decipher the embedded text, whereas programs are not. One could also use linguistic problems. We will however restrict to Captcha-based solutions.

Please note however that such solutions are only partial and aim to *strengthen* the voting security in *limited* scenarios. One could of course always rely on a human to resolve the CAPTCHA challenge. The development of image recognition may also make this solution a short term one only.

3.5.1 Text Captchas

A first (easy but weak) solution would be to use such Captcha images to display the candidates names. However, current Captcha algorithms are now cracked with higher and higher success rates (and with no clue about the text to decipher). Considering also the fact that there is a limited number of candidates, and that identifying only part of the Captcha image would be enough to identify a candidate, we can infer that a malware with Captcha breaking capabilities will bypass the Captcha barrier with a success rate close to 100%. Basically, a malware would only need to determine which candidate's name is the most likely to be represented in the Captcha image, without having to decipher the precise contents of the image. Note that even the length of the candidates names is a discriminating factor. In the end, any low efficiency Captcha breaking algorithm would be appropriate.

Some litterature is covering the breaking of Captchas (Chellapilla and Simard, 2004; Mori and Malik, 2003; Chellapilla et al., 2005) and recent implementations have been brought to light in scenarios like automatic creation of accounts on Hotmail, Gmail, Yahoo Mail, Blogger, etc. Success rates are typically between 10% and 35% with an optimal time of 6s (Websense - Sumeet Prasad, n.d.). Optical character recognition (OCR) techniques can also be used in a statistical approach to determine from which candidate name an image is the closest.

3.5.2 Image Captchas (or semantic Captchas)

Image identification Image captchas can be stronger than text captchas. Common scenarios include identifying from a list of terms what best describes the displayed image. In Internet voting, one could imagine

a scenario where several images are displayed and a text telling: "To vote for candidate A, select the picture representing a cat. To vote for candidate B, select the picture representing a tower.", and so on. Some implementations exist, like Microsoft's Asirra Human Interactive Proof: given say 10 photos, the user is asked to select all photos that represent a cat. Another implementation can be found on the Captcha Project's website (The CAPTCHA Project, n.d.): four pictures are displayed to the user which is asked to identify in a list a feature common to all pictures.

This first solution has some weaknesses. First, the number of images must be big enough to resist to an exhaustive identification. Second, unless a high number of images is displayed to the voter (which is an unlikely scenario), a random vote attack is likely to have a high success rate.

Detail identification   This technique relies on the human distinction of details inside an image. So-called 3D Captcha is an implementation of such Captchas: an image is generated which assembles several features together and annotates the different parts of each feature with some text; then the user is asked to enter the text written on some part of some feature. If each text component is one letter long and there is enough different details to cover the alphabet and the user is asked to enter the text of several components, then there is no other way for a breaking algorithm but to identify each detail in the image. An example of such a 3D Captcha can be found at (Kaplan, n.d.).

In an Internet voting scenario, we have to make sure the choice pool is large enough to prevent random vote. To ensure that, we could for instance propose one 3D Captcha per candidate (generating a single image or one image per candidate). Then, to vote for candidate A, a user would be asked to enter the code sequence corresponding to part i_1 of feature j_1, followed by part i_2 of feature j_2, and so on (the length of the resulting code must be random and large enough to ensure protection against random vote and vote changing attacks). We insist that the different text components of the image must cover a large enough set of characters for this technique to be strong.

Another technique can finally be derived from this technique. Rather than annotating each detail of the image, one could ask the user to directly click on those details (which would be identified by polygonal areas in the image). The advantage of this technique would be to have a finer-grained control of the set of possible inputs (by segmenting an image in a set of small enough areas). Using a sequence of clicks furthermore increases the resistance of the algorithm against breaking. Also, one can use either computer-generated images or real images. Computer-generated images are easier to implement but could be in the long term easier to automatically analyse. Such images can be strengthened by adding a complex background which an human will easily distinguish, unlike a computer algorithm. Real images on the other side are complex, relying on the full distinguishing abilities of the human brain, but must have been previously segmented in areas.

Examples   We consider 3 scenarios of voting by detail identification, inspired from the previous 3D Captcha.

Input:
A computer-generated image with a standing person, a sitting person and a vase with a flower. Each part of each element is annotated with letters, which cover the whole alphabet.

Instructions:
To vote for candidate A, enter the characters annotating each of the following details:

1. The right hand of the sitting person;

2. The flower stem;

3. The left leg of the sitting person;

4. The body of the standing person.

To vote for candidate B, enter the characters annotating each of the following details:

1. ... *(Another sequence of details)*

Input:
A computer-generated image with a standing person, a sitting person and a vase with a flower.

Input:
A real image featuring among others a tower with an antenna, a tethered dog and a window store.
Instructions:
To vote for candidate A, click on each one of the following details in the image:

1. The top of the building's antenna;

2. The dog's tail;

3. The window store;

4. The dog's leash.

To vote for candidate B, click on each one of the following details in the image:

1. ... *(Another sequence of details)*

Note that none of these solutions is protected against a third party resolution of the challenge, like sending to a remote server the question along with the image. Neither are they protected against voter impersonation, that is stealing the voter's credentials and deluding him in a fake voting process. Similarly, vote dropping cannot be avoided either. Finally, sending to a server both the challenge and the user's response allows to disclose the voter's choice. Actually, in a non hybrid solution, and using mainstream computers, no solution can protect against these attacks.

4 Conclusion

Internet voting is supposed to improve participation by allowing anyone to vote in a simple way. Current implementations however have weak protections that can be easily defeated by simple browser addons, though they could be improved. Internet voting is certainly an efficient step toward improving election participation, however we argue that some attacks cannot be avoided without using a parallel medium. Denial of service, vote disclosure, credentials stealing can always be achieved in pure Internet voting scenarios — when using general-purpose computers with mainstream operating systems, where security cannot be efficiently ensured, such as public computers and to some extent home computers.

However, solutions relying on Human Interaction Proofs principles (as is done with CAPTCHA algorithms) at least allow to efficiently protect the voting process against automatic vote tampering in limited scenarios. When hybrid solutions cannot be adopted, we highly encourage toward the use of strong solutions relying on human capacities. Since this protection can still be circumvented, important elections should definitely not rely on such solutions.

References

Beaucamps, P. and Reynaud, D. (2008), Malicious Firefox Extensions. Available at: `http://lhs.loria.fr/?p=33`.

California Internet Voting Task Force (2000), A report on the feasibility of Internet voting, Technical report, California Secretary of State, Sacramento.

Charpentier, F. and Hamon, Y. (2008), Autopsie et observations in vivo d'un banker, *in* 'SSTIC'08 Proceedings'.

Chaum, D. (2001), 'SureVote, International Patent WO 01/55940 - `http://www.surevote.com`'.

Chellapilla, K., Larson, K., Simard, P. Y. and Czerwinski, M. (2005), Computers beat humans at single character recognition in reading based Human Interaction Proofs (HIPs), *in* 'CEAS'.

Chellapilla, K. and Simard, P. Y. (2004), Using machine learning to break visual human interaction proofs (hips), *in* 'Advances in Neural Information Processing Systems 17, Neural Information Processing Systems (NIPS'2004)', MIT Press.

Enguehard, C. (2008), Transparency in electronic voting : the great challenge, *in* 'IPSA - International Political Science Association RC 10 on Electronic Democracy', Stellenbosch University, South Africa.

Estonian Electoral Committee (n.d.), '`http://www.vvk.ee/engindex.html`'.
　　URL: *http://www.vvk.ee/engindex.html*

Hensler, R. (2004), '`http://www.coe.int/t/e/integrated_projects/democracy/Hensler%20Swiss%20presentation.ppt` - The Geneva Internet voting project'.

Joaquim, R. and Ribeiro, C. (2007), 'Codevoting protection against automatic vote manipulation in an uncontrolled environment', *Lecture Notes in Computer Science* **4896/2007**.

Kaplan, M. (n.d.), '`http://spamfizzle.com/CAPTCHA.aspx` - The 3D CAPTCHA'.
　　URL: *http://spamfizzle.com/CAPTCHA.aspx*

Louw, M. T., Lim, J. S. and Venkatakrishnan, V. N. (2008), 'Enhancing web browser security against malware extensions', *Journal in Computer Virology* .
　　URL: *10.1007/s11416-007-0078-5*

Mohen, J. and Glidden, J. (2001), 'The case for Internet voting', *Commun. ACM* **44**(1), 72–85.

Mori, G. and Malik, J. (2003), Recognizing objects in adversarial clutter: breaking a visual CAPTCHA, *in* 'Proceeding of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition', Vol. 1, pp. I–134–I–141 vol.1.

Mozilla Security Blog (2008), '`http://blog.mozilla.com/security/2008/05/07/compromised` - Compromised file in Vietnamese Language Pack for Firefox 2'.
　　URL: *http://blog.mozilla.com/security/2008/05/07/compromised*

Oppliger, R. (2002), How to address the secure platform problem for remote Internet voting, *in* 'Proceedings of the 5th Conference on "Sicherheit in Informationssystemen" (SIS 2002)'.

Phillips, D. M. and von Spakovsky, H. A. (2001), 'Gauging the risks of Internet elections', *Communications of the ACM* **44**(1).

Rubin, A. D. (2002), 'Security considerations for remote electronic voting', *Communications of the ACM* .

Safevote (2000), 'Voting system requirements', *The Bell* **1**(7), 3.
　　URL: *http://www.thebell.net/papers/vote-req.pdf*

The CAPTCHA Project (n.d.), '`http://www.captcha.net/cgi-bin/esp-pix`'.
　　URL: *http://www.captcha.net/cgi-bin/esp-pix*

Volkamer, M., Alkassar, A., Sadeghi, A.-R. and Schulz, S. (2006), 'Enabling the application of open systems like pcs for online voting', *Frontiers in Electronic Elections* .

von Ahn, L., Blum, M., Hopper, N. and Langford, J. (2003), 'CAPTCHA: Using hard AI problems for security', *Advances in Cryptology – EUROCRYPT 2003* pp. 646–646.

von Ahn, L., Blum, M. and Langford, J. (2004), 'Telling humans and computers apart automatically', *Communications of the ACM* **47**(2), 56–60.

Wearden, G. (2002), '`http://news.zdnet.co.uk/internet/0,1000000097,2109249,00.htm` - UK's first online voters go to the polls'.
URL: *http://news.zdnet.co.uk/internet/0,1000000097,2109249,00.htm*

Websense - Sumeet Prasad (n.d.), '`http://securitylabs.websense.com/content/Blogs/3063.aspx` - Microsoft Live Hotmail Under Attack by Streamlined Anti-CAPTCHA and Mass-mailing Operations'.

Zùquete, A., Costa, C. and Romao, M. (2007), An intrusion-tolerant e-voting client system, *in* 'Workshop on Recent Advances on Intrusion-Tolerant Systems (WRAITS 2007)'.