Exploiting the Client Vulnerabilities in Internet E-voting Systems: Hacking Helios 2.0 as an Example

Saghar Estehghari
University College London
s.estehghari@cs.ucl.ac.uk

Yvo Desmedt
UCL and RCIS

Abstract

Helios is a web-based open-audit voting system designed using state of the art web technologies and advanced cryptographic techniques to provide integrity of ballots and voter secrecy in an insecure Internet environment. In this paper, we demonstrate a simple attack against Helios 2.0 that takes advantage of the fact that every candidate in Helios can provide a URL referring to his/her candidacy statement. A malicious candidate, who wishes to win a Helios-managed election, uploads a specially crafted PDF file containing a candidacy statement to his/her website. The attack is then triggered against each voter who is using a vulnerable machine. The security of the machine is undermined, e.g., when the voter visits the attacker's webpage. In essence, we exploit Adobe Acrobat/Reader's vulnerabilities to install a malicious browser extension on the voters' machines. Such an extension provides an opportunity for an attacker which may fool the voter (using Social Engineering) into accepting a hacked ballot. Due to our attack Helios 2.0 was upgraded to Helios 3.0. We discuss generalizations and the impact of the latest upgrade of Helios on security. We also discuss defences against this attack, generalizations and the impact of the latest upgrade of Helios on security.

1 Introduction

Voting systems are highly sensitive in nature, and as such they are a prime target for opportunists interested in biasing the results of an election. Electronic voting systems have opened new avenues to these adversaries that might be tempted to exploit any software vulnerability in these systems to break the integrity and secrecy of ballots (also called anonymity of the voters). Some of the current e-voting systems suffer from exploitation of many vulnerabilities, e.g. [32], [30], [9], etc. Indeed, these voting systems are highly dependent on the security of the software run on the voting terminal, and therefore these are vulnerable to any flaw in the security design.

Researchers have proposed cryptographic voting schemes (e.g. Mix-Nets, Homomorphic schemes, etc.), to provide several desirable properties such as security, efficiency and accuracy. In this paper we analyze the impact of software vulnerabilities on such cryptographic voting schemes.

Web-based voting, among these e-voting systems, are providing organizations with more flexibility to conduct their internal elections. In designing and deploying Internet voting systems, advanced cryptographic techniques and state of the art web technologies can be employed to protect such systems on the server side. However, such technologies are limited on the client side which may make these systems vulnerable to different hacking techniques. As a result the impact of both server and client security on the privacy and integrity of the Internet voting systems must be carefully considered.

In this paper, we focus on Helios 2.0, a promising web-based open-audit voting system [6], which uses lots of cryptography to demonstrate how an adversary might exploit clients machines vulnerabilities to mislead the voter.

Although we did not break Helios 2.0 from a cryptographic viewpoint, our attack was able to demonstrate that compromising Helios 2.0 on the client side is more feasible than previously believed. More precisely, in Helios 2.0, except if one is certain that large majority of voters take extraordinary precautions, our attack undermines the integrity of the voting system (for more details, see the rest of this paper).

We identify potential approaches to allow for a successful attack against an Helios-based election and select one of them which is more feasible and less detectable (Section 4). We remark how Helios 2.0 was modified to weaken the impact of our attack (Section 5.1). Then we discuss the legal issues, costs, impact of the attack (Section 6). Along with the flaws we exploit to launch our successful attack, we identify its limitations, and hence propose defenses and techniques that could prevent such type of attacks (Section 7).

2 Related Work

2.1 E-voting Systems

Before we survey prior attacks on e-voting systems, we mention that we discuss here exploits that can be performed on e-voting systems using non-verifiable voting schemes. From a *cryptographic* viewpoint there is a fundamental difference between verifiable cryptographic voting schemes and non-verifiable ones. Helios, the web-based e-voting system that is analyzed in this paper, utilizes verifiable voting scheme where each voter is able to audit the ballots and to verify its correctness (see Section 3.1). In Sections 4 we study the impact of our attack on internet based verifiable voting systems, such as Helios.

Prior research has focused on analyzing and evaluating security measures considered in the context of booth-based e-voting systems. In such systems, voters go to the polling stations and submit their ballot electronically using the voting terminal, i.e. [32], [30], [9], etc. These systems are highly dependent on the security and correctness of the software running on the voting terminal. Any flaws or bugs contained in the software can promote a malicious voter or insider to exploit the system.

The analysis of Diebold AccuVote-TS carried out by Kohno et al. [32] was one the first papers in this field. The researchers had access to the source code of the system, since it had appeared on the Internet. According to [32], Diebold violates confidentiality and integrity of the ballots and anonymity of voters.

The research conducted by Hursti [30] on Diebold AccuVote-TS6 and AccuVote-TSx exploited publicly accessible documentations, "source code excerpts" [30] and the results were obtained by testing a real system. According to [30], exploitation of design flaws in these systems may help an attacker to compromise the security of an election.

Recently, research was carried out by Appel et al. [9] on Sequoia AVC Advantage voting machine. Due to the Court order, the authors could study the actual voting machine's hardware and source code for a month. In [9], they showed that fraudulent firmware can be installed on the machine, allowing one to tamper with the integrity of an election.

Other research related to AVC Advantage machines was carried out by Checkoway et al. [15]. In their work, the scientists did not have access to the source code of the voting software or to the documentation. They found security flaws in the system by reverse engineering the voting software and hardware. They could successfully show the possibility of a vote-stealing attack by employing Return-Oriented Programming [39] technology without code injection (as the system rejects any code injection).

2.2 Browser Rootkits

The basic idea for our attack is inspired by an attack proposed in [33]. In this paper, researchers have developed a malicious extension for Firefox called BROWSERSPY. They installed the extension (without introducing a specific approach for installation) by injecting it to Google Toolbar extension. The Browser Rootkit steals saved passwords in the browser and send them together with the URL to the remote attacker (also see [42]).

Hackers have already exploited Internet Explorer extensions to develop and install Browser Rootkits on client's machines. Download.ject [37] is a malware that installs a malicious extension on Internet Explorer. The task of such extension was to listen to the browsing history until detecting an SSL/TLS connection related to a bank transaction. Then it captured the username and password of the victim and sent the confidential information to the attacker.

3 Background

In this section, we survey Helios 2.0 and a typical voting process, then we briefly present both software vulnerabilities and current browsers capabilities that might be exploited to allow for remote code execution so as to compromise an election in Helios 2.0.

3.1 Helios Voting System Version 2.0

Helios is an open-source web-based voting application which can be deployed by any organization, group or community to set up an election. A web browser is the only tool needed by both voters and system administrators to interact with the system. According to [6, 7], it is a secure by-design Voting system, as it "implements advanced cryptographic techniques to maintain ballot secrecy while providing a mathematical proof that the election tally was correctly computed".

Specifically, Helios is implemented utilizing additive homomorphic techniques for e-voting and Exponential ElGamal, a variant of ElGamal where g^m is encrypted instead of m, as the encryption algorithm. Helios supports threshold decryption [20] with joint key generation [8].

The web browser plays an important role in Helios' functionality. In particular JavaScript is extensively used in the Helios application. The election data, such as the encrypted ballot, the ballot's plaintext and the randomness are stored in browser's memory before the ballot submission. In addition, Exponential ElGamal is implemented using the JavaScript language. Because the JavaScript technology itself is quite slow when performing computationally complicated tasks such as generating randomness and modular exponentiation for ElGamal, Java is utilized for such computation. This requires that the Java Virtual Machine (JVM) needs to be installed on the web browser. Consequently, the encryption processes are primarily performed in the web browser.

The integrity and secrecy of votes are among the main objectives of the Helios security model. From that perspective, Helios designers claim that " ... even if Helios is fully corrupt, the integrity of the election can be verified" and "... assuming enough auditors, even a fully corrupted Helios cannot cheat the election result without the high chance of getting caught". As for any other e-voting system, different types of attacks may threaten the security of Helios. By leveraging on the benefits of a so called open audit election, the Helios system claims to defeat most, if not all, of these attacks, such as changing a ballot, voter impersonation, ballot corruption, incorrect decryption, etc. [6].

3.1.1 Role of Auditing

According to [6], Helios 2.0 uses two verification programs, the one which can be utilized by the voter during the voting process and prior to vote submission to audit the encrypted ballot given by the Ballot Preparation System (BPS), and the other, used by the administrator, to verify the shuffling, decryption and tallying of the election.

Benaloh's Simple Verifiable Voting [11] is employed for the purpose of the former verification program. The key feature of this protocol is the separation of ballot preparation and the casting of a vote in a voting system, where authentication of the voter is not required prior to the ballot casting time. According to [6], this makes testing of BPS by anyone (both eligible/non-eligible voters) possible and increases auditability.

Using the second verification program, an administrator (or anyone else) can audit the whole election, after tallying an election. In this process, a list of voters along with the hash of their ballots is published by the system. The administrator should republish this list, where the voters can compare their own hashes with the published ones and verify the correctness of their submitted ballot. The main intuition behind the auditing process is that if a vast majority [6] of voters verify the correctness of their votes then the outcome is not biased by any type of attack, and the vote result is deemed correct. This technique gives an opportunity to the voters to complain, in the case the hashes do not match. Auditing is then of primordial importance during Helios elections processes. Its security of this part is analyzed from Section 4 on.

3.1.2 Voting in Helios

A typical voting process starts when a voter receives an invitation email from the Helios administrator. Once visiting the election webpage, a voter interacts with the Ballot Preparation System (BPS) by going through the questions and selecting his/her answers. These answers are then recorded by the BPS. It is interesting to note that next to each answer, the voter is provided with a link to the candidates' personal webpage that typically contain the candidacy statements.

On the confirmation page, the selected votes/answers are displayed on the screen. If the voter is not satisfied with the choices, he/she can go back and update the votes/answers. Otherwise he/she confirms them. The BPS encrypts the ballot and shows the hash of the encrypted ballot to the voter in the next step.

The voter may decide to audit the ballot. The BPS returns a data structure, containing the election ID, the ballot's plaintext, ciphertext, and the randomness used for encryption, as an audited ballot to the voter. Moreover, the voter may decide to verify the encrypted ballot. The BPS displays then a text area inside which he/she should copy and paste the audited ballot. The BPS then returns the hash of the election, the hash of the encrypted ballot, and the ballot's plaintext as an output and states whether the encryption was verified and whether the proof is correct.

Finally, he/she can choose to submit the encrypted ballot. However if the ballot was audited, it cannot be cast and the selected choice must be encrypted again. At this step the BPS clears the randomness and the ballot's plaintext from the browser's memory and brings up the authentication page. The voter should enter the username and password specified in the invitation email. If the combination of the username and password is correct, then the ballot is signed by the BPS and stored in the database. The voter will receive an email, containing the hash of the encrypted ballot and the hash of the election, confirming that his/her ballot has been recorded. Yet, for insuring the forward secrecy of the vote, the email does not contain the name of the chosen candidate in the plaintext.

Note: The Ballot Preparation System (BPS) is implemented by utilizing JavaScript technology and it manages the client side part of the Helios application.

3.2 Software Vulnerabilities and Remote Code Execution

Today buffer overflow is a well known technique enabling software exploits [17]. Typically, it consists in providing a long argument as an input to the program, so that it may result in the corruption of adjacent parts of the program's state, i.e. pointers and stored memory addresses. This erratic behavior gives the possibility to an attacker to write arbitrary code in this part of the program's state. This injected code is called the payload or shellcode [44]. As a result, the flow of the program is diverted to the hacker's program. The payload is then executed with privileges of the vulnerable software. Consequently, the attacker can gain full or partial control over the victim's machine. In the following we describe the buffer overflow vulnerabilities in Adobe Acrobat and Adobe Reader. We exploit such vulnerabilities in our attack, as described in Section 4.3.

Adobe and JavaScript Vulnerabilities. Adobe Acrobat and Adobe Reader are popular tools for viewing, searching, printing and digitally signing PDF documents. In its recent versions, Adobe Acrobat allows creators to customize PDF documents using JavaScript. To this aim, Adobe developed a JavaScript API which is only supported by Adobe Acrobat and Reader in their v7.0.0 versions or later. Using this API, it is possible to bind JavaScript functions to elements of a PDF file and events, such as user actions.

However, several Adobe versions, such as 7.0.0, 8.0.0, 8.1.0, 9.0.0 etc. are vulnerable to buffer overflow attacks. The attack can be launched against machines on which clients open a specially crafted PDF file with a vulnerable version of Adobe Acrobat/Reader. As mentioned earlier, this makes the execution of a payload possible on these machines. The buffer overflow vulnerabilities are partially due to the way the JavaScript functions are implemented in Adobe JavaScript API. Some of these functions include, `Collab.collectemailinfo()` [24], `util.printf()` [25], `Collab.getIcon()` [23] etc. Later, in Section A.1, we show how the vulnerabilities in `Collab.getIcon()` function can be exploited and utilized as a tool to launch an attack against Helios 2.0.

3.3 Browser Rootkits and Extensions

Browser extensions (or add-ons) are facilities provided to customize a browser to offer additional features that fit the personal needs of each user [33, 21]. Browser extensions are supported by both Internet Explorer and Mozilla Firefox. For the purpose of our study, in the following we only focus on Firefox extensions. However, it is worth noticing that our results are generic and our attack is easy to set up using Internet Explorer extensions.

The extensions become part of the browser after installation and are able to change the browser's behavior, as seen by the client. They have the following capabilities:

- They have access to Document Object Model (DOM) of web pages and are able to add, edit or remove DOM elements of an HTML document.
- They have access to JavaScript functions of a web page and are able to edit JavaScript contents.
- They can read from and listen to the session history of the web browser. This implies that the extensions can wait (listen) for a particular user's action (event), such as clicking on the "back" or "forward" button on the browser, etc., and return the URL of the page that the user is currently visiting.

Structure of Installed Extensions : Firefox extensions contain two important files: install.rdf and chrome.manifest. Install.rdf contains metadata about an extension and its creator. The existence or absence of this file suggests whether an extension is installed or not. Chrome.manifest is used by the browser to find and load the extension's files corresponding to User Interface (UI), Locale and Skin.

Exploiting Firefox Extensions : Extensions allow Firefox users to enhance the functionalities of the browser. However, attackers can develop malicious add-ons that are able to steal user's sensitive data or tamper with the integrity of such data (e.g. [33, 37]). These malicious extensions are called Browser Rootkits. Clients can be tricked to install such an extension on their machines. Despite the security measures implemented in Firefox, it is still possible to install an arbitrary extension on clients' machines without their permission. Indeed, the Firefox extension manager never checks the integrity of extensions on the browser startup, as it is unaware of the past and present status of these extensions. This gives an opportunity to an attacker to inject a malicious add-on to an already installed extension. This is done by copying UI files into the victim extension's folder and changing the chrome.manifest file to point to the malicious files. The files can be copied by a malware or by a payload, which is run after exploiting software vulnerabilities as explained in Section 3.2.

4 Analysis and Design

In this section, we start by providing our motivation for focusing the research on the Helios Voting system. Then we identify potential approaches to allow for a successful attack against a Helios-based election. Following, the most feasible approach is selected and a brief overview of the attack is given. Finally, we describe the assumptions under which our successful attack is launched against voters.

4.1 Why Analyzing Helios?

There are lots of web-based voting systems available on the Internet, such as BigPulse [12], Civitas [16], TrueBallot [41], Adder [36], Helios Voting System [7] etc. Some of these applications are open source, such as Civitas, Helios and Adder, and some are proprietary, i.e. BigPulse. Our aim was to analyze and evaluate the security measures of an open source e-voting system, since anyone can download the source code and deploy the application according to his/her need. Moreover, we wanted to analyze the vulnerabilities of a web-based voting system employing the state of the art cryptographic tools.

The International Association for Cryptologic Research (IACR) has decided to move towards e-voting systems for its internal elections and to explore different options. In the IACR Board Meeting on E-Voting in August 2008 [5], 8 proposals were presented, including Helios. Moreover, the deployment of Helios in [8] won the 'Best Paper' award at EVT/WOTE 2009.

For all above reasons, we became confident that the Helios Voting system was perfectly suited for the objectives of this research.

4.2 Potential Approaches to Launch an Attack

For a better understanding of the nature of the attack, we assumed a particular candidate who wants to fraudulently increase the number of votes that are in favor of his/her in an election. To prevent detection, the attacker wants to hide the attack. This implies that the attack must be implemented in a way that voter's permission for the execution/installation of a malicious software is not required. Moreover, the voter should not notice the modification of the ballot.

In the rest of this section, we survey potential approaches towards an attack.

Malware: As mentioned earlier, each candidate can have a candidacy statement. Such a document can be in PDF, DOC or HTML formats. Suppose that the attacker has prepared a PDF file in which a virus or worm is embedded. Prior to the election, the candidate can e-mail this file to a list of voters. The malware is activated upon opening of the file by the voters, who trust the candidate. The malware is set out to infect the web browser and, consequently, change the behavior of the application. Therefore, the infected browser may be able to tamper with the integrity of the election and violate the voter's anonymity (except if the voter is very dedicated to take special measures, as explained in [18] and Section 7.2). However, worms and viruses replicate themselves and spread over networks and the Internet. The aim of the research was to mount an attack against the voters while avoiding any damage to the sensitive information on their machines.

Cross-Site Scripting (XSS) Attacks: These kinds of attacks are application specific. The exploit largely depends on the security measures considered in the application. Helios defeated persistent XSS attacks by avoiding insertion of any user's contributed contents in the application. In other words, Helios does not provide editing facilities and candidates cannot compose their candidacy statement inside the application. Moreover, the designer was careful about non-persistent XSS attacks by validating input data and URLs entered by the end user.

Browser Rootkits: As explained in Section 3.3, browser extensions are able to monitor the user's navigation history and to manipulate the DOM tree of a webpage. The attacker can exploit these capabilities of extensions to modify or override the JavaScript functions that control the client side of the application. Consequently, he/she can distort the behavior of the voting system. The candidate can make use of software vulnerabilities to install a Browser Rootkit on voters' machine when visiting the candidate's webpage. Through this approach, the required user permission for installation of an extension, is circumvented. Browser Rootkits go beyond XSS attacks. Like malware, they can change the behavior of a web browser without replicating and spreading themselves.

4.3 The Selected Approach

The attack proposed in this paper and developed in software is a combination of two approaches. First, the attacker exploits vulnerabilities in Adobe Acrobat/Reader to install a Browser Rootkit on a voter's machine. For this to work, the attacker prepares a PDF file containing not only the candidacy statement, but also binds a malicious JavaScript function to the "open" event of a file. Second, the payload of the first type of attack is a Browser Rootkit, which is may be able to fool the voter (using Social Engineering) into accepting an incorrect ballot in Helios 2.0 (for defenses see [18] and Section 7.2).

We now briefly describe how the attack works on the client's machine. Upon opening the PDF file, a buffer overflow is created and a shellcode will be executed. The payload installs the malicious extension on the voter's machine. More precisely, the Browser Rootkit is injected into an already installed extension to suppress the notification given by the Firefox extension manager. Indeed, as explained in Section 3.1, Helios requires Java Runtime Environment (JRE) for its functionality. This implies that each voter must install JRE on his/her machine. The software installs an extension, called Java Console, on Firefox. This insures that Java Console is installed on voters' machines and makes this extension an ideal victim in the attack.

After the Browser Rootkit has been installed, the payload closes the browser, as Firefox needs a restart to reload the changes that have been made to Java Console. When the client restarts Firefox, the Browser Rootkit listens to the browsing history of the voter. The browsing rootkit commences its malicious actions whenever the voter visits the voting website. The extension then injects some JavaScript codes into the DOM tree to override the existing functions and modifies the original functions to alter the ballots that are in favor of others to become in favor of the malicious candidate.

Through this attack, the malicious extension has full control over the client side of the voting system. It is able to change the behavior of the application and to deceive the voters to believe that they have voted for the desired candidate. For more technical details, see Appendix A.1. For more general scenario, see Section 8.

4.4 Assumptions

We presented a certain number of conditions under which a successful attack can be launched against voters. Consequently, if a voter's machine does not satisfy these requirements, then the attack will not be triggered.

Assumptions about the voter's machine : The attack works on the client machine where Windows XP is the operating system, the voter uses Firefox as the web browser and Adobe Acrobat/Reader version 7.0.0 to 8.1.2 or 9.0.0, is installed. Moreover, we assume the voter has the "write" access to the Firefox installation folder.

Assumptions about the election : A mock IACR Election was chosen to demonstrate the vulnerabilities of the Helios 2.0. It is used as the voting system targeted in the rest of this paper. The election consists of one question and only one candidate must be chosen. There are only two candidates, one of the authors of this paper, we call 'Alice' and the other 'Bart Preneel' (the current president of IACR). The names of these candidates are sorted alphabetically in the question. The administrator has given extra information about each candidate by providing a link to their personal web page. The attacker is 'Alice' who wishes to win the election.

4.5 Design

Two weeks were spent on the development of the actual attack software. Around 950 lines of code were written for this attack. Of these, roughly 50% is dedicated to the development of the malicious extension. The other 50% is related to embed JavaScript for Adobe Acrobat and the executable program. Only 10% of the codes is unique to Helios. The software does not slow the client machine down. The only noticeable event during the attack runtime is the sudden closure of the browser, as the Firefox needs a restart for loading the changes that have been made to the victim's extension.

Comparing the difficulty of hacking Helios 2.0 with non-cryptographic Internet voting systems, Helios 2.0 has an auditing feature on the client-side. As a result of this, we add extra code which may fool the voter to believe that his/her vote was encrypted correctly and the desired candidate was selected. So, we in fact use Social Engineering to mislead the voter. Although Helios is a verifiable voting scheme, nothing prevents this attack against Helios 2.0, except voters taking extraordinary precautions (see [18] and Section 7.2).

5 Recent Developments and Their Impact

5.1 Helios version 3.0

In the earlier versions (1.0 and 2.0) of Helios an adversary was able to make the auditing process much more cumbersome, as explained in Section 4. After our Crypto 2009 Rump-Session presentation [19] Helios 2.0 has been modified to address the problem we pointed out, as following.

According to [29], in Helios 3.0 the voters are now able to post the audited ballot to the Helios server. This implies that not only the voter is able to check whether the hash was properly computed, but also the ballot data, i.e. the randomness, the vote and the hash, can be posted on some public webpage. This can be done *before* casting. The voter can write down the vote and the hash value on some piece of paper (or can take a picture using a camera). That allows the voter or others to use a 2nd, 3rd, etc., computer to check the public WWW. Assuming the server to be trusted (which we do), the public WWW will compute the correct hash. (In case not, auditors can check whether the correct hash is posted on that WWW.) We now discuss whether these modifications address our security concerns.

5.2 Impact of Alterations on Anonymity

In Section 8 we explain how the Browser Rootkit can be further extended to mount an attack against the anonymity (privacy) of the voters in Helios. The aforementioned upgrade of Helios does *not* protect against such an attack.

6 Discussion

In this section we wonder how realistic it is for someone to actually use the above attack. To answer this question several issues need to be taken into account, which include: the motivation of the attacker (financial, political, etc.), legal ones, the cost to the attacker (e.g., setting it up, possible legal fees, etc.), the likelihood of being able to change the outcome, whether alternative attacks are more effective. We now discuss these and conclude by reflecting back on our original scenario.

6.1 Motivation of the Attacker

Although we focused on having the candidate hack the elections, in reality the attack might come from outsiders. Although hacking was originally done for "fun," or to show off, in today's Internet many attacks have financial motivations. So, obviously both motivations apply to the scenario of electronic elections. To better understand these motivations, it is evidently important to wonder for what position the election is for. Examples include: private clubs, associations (e.g., professional ones), inside a board of (large) corporation, or political ones. In the last case an election for mayor of some small village will be very different from one for the election of a president of a country where the attacker may also have political motivation.

Today several countries, such as Estonia [34], Finland [1], Switzerland [4] have already moved towards Internet voting and others as Norway [2] and the UK [3] are considering it or have done pilot tests. Moreover, due to several people being unable to vote in the recent UK election, there has been a call for e-voting as a possible solution to prevent future problems at election polling stations [10]. Since there are several problems with this approach, the Netherlands [22, 27] did not make this move. In such political elections, hackers could be lobbyists (or working for them), or politicians (e.g., these already in power), or by foreign countries.

On the other hand, as point out by an anonymous refree, web-based voting systems like Helios are likely to be used in contests with somewhat smaller stakes, and that candidates (attackers) in these scenarios sometimes still have large incentives to cheat but face smaller chances of being caught.

6.2 Legal issues

In certain countries launching our attack might be illegal. However, due to the openness of the Internet, attacks could be launched from abroad where such attacks are not illegal or the likelihood of being prosecuted are small. Furthermore, in case the attack is launched by a well financed organization, as a large lobbying group, legal fees maybe secondary and ways to bypass prosecution might be analyzed in detail by legal experts.

6.3 Costs

In the case of the proposed attack, the size of the election decides the number of alternative servers required to manage concurrent requests to download the files (the PDF and the executable). This may include a small fee for hosting subscriptions. The attacker may use free hosting services to minimize the costs. On the other hand, as explained in Section 4.5, the amount of time and effort spent by the attacker to develop such an attack is quite low.

6.4 Impact

In Section 6.1 we listed some countries interested in moving towards internet voting. In such circumstances there might be a clear incentive to hack the election, and thousands of computers would be used for voting. The impact might be that computers that were unlikely to be the target of an attack become compromised.

Adobe Acrobat and Readers were likely the most-hacked software products of the year 2009 [28]. According to [31], 83.5% of the 2.5 million users run a vulnerable version of Adobe Acrobat/Reader on their machines. As remarked by the report, this is mainly due to the obstacles in the software update mechanism, which results in ineffective distribution of security patches. On the other hand, nearly 30% [35] of Internet users utilize Mozilla Firefox as the browsing tool, and more than 80% [26] of these users have Adobe Acrobat/Reader plugin installed on their browsers. In addition, Windows XP is currently one of the most popular operating system among users [43][40].

Consequently, using Internet elections at a national scale might make millions of computers vulnerable and the proposed attack effective with profound impact (also see Sections 7 and 8).

6.5 The IACR Elections

Our attack was motivated by IACR's (International Association for Cryptologic Research) decision to move towards electronic voting. One can wonder whether a hacker would really target their elections. However, since IACR is an association of researchers working on aspects of information security, a successful attack could embarrass their association. An interesting observation made by [38] is:

> A truly "successful" attack against IACR wouldn't be discovered, and so wouldn't embarrass them. IACR members are probably better equipped to pull off such an attack than members of typical organizations, so I wouldn't dismiss the threat so readily.

The impact of having an IACR election hacked are besides a potential embarrassment, rather minimal in comparison to scenarios discussed in Section 6.1.

7 Limitations and Defenses

In this section, we discuss the limitations of the proposed attack. Then we present a number of defenses against the attack and highlight their shortcomings.

7.1 Limitations

The attack has a number of limitations which are related to:

Adobe Acrobat/Reader Updates: The attack takes advantage of buffer overflow vulnerability in Adobe Acrobat/Reader. This vulnerability has appeared in Adobe Acrobat/Reader versions 7.0.0 to 8.1.2 and 9.0.0, and it was claimed to have been fixed in later versions of the software. As a result, in its current form, the attack will fail when the voter updated the Adobe Acrobat/Reader to the latest version.

Other PDF Viewer Software: There are other PDF viewer tools that have similar functionalities as Adobe Acrobat/Reader. These tools do not support Acrobat JavaScript API. Since the attack exploits Adobe Acrobat/Reader vulnerabilities, voters that use other applications, than Adobe, to view the PDF file, cannot be targeted using our approach.

The attack only targets Firefox browser: In our feasibility study, the malicious extension was designed and developed for Firefox. On the other hand, there are other popular browsers. According to [26] , around 70% of clients used Internet Explorer. As a result, the candidate may lose a considerable proportion of votes unaffected by the proposed attack.

The attack is platform specific: The executable is implemented in a way that installs the malicious extension only on machines where Windows XP is the operating system. However, around 50% [40] of clients uses other operating systems, such as Linux, Mac etc., which were not considered when designing the attack.

Windows Vista Security Measures: Due to security measures considered in Windows Vista, the malicious extension cannot be installed on this operating system using the buffer overflow vulnerabilities in Adobe Acrobat/Reader. When the malicious PDF document is opened on the victim's machine, Adobe Acrobat/Reader crashes but the operating system prevents the execution of the payload. As a result the executable will not be downloaded and executed by the payload. However, if the attacker employs a virus or a worm to install the extension, he/she can make the attack work on this platform as well.

7.2 Defenses

There are possible ways to defeat the proposed attack, including:

Disable the JavaScript in Adobe Acrobat/Reader. Adobe Acrobat/Reader vulnerabilities can be exploited only when the JavaScript is enabled in the software. Clients can disable this option in the preferences' section of the application (by default it is enabled). When a document containing JavaScript code is opened with the software, where the JavaScript is disabled, the program pops up a dialog box saying "This document contains JavaScripts. Do you want to enable JavaScripts from now on? The document may not behave correctly if they're disabled". If the voter trusts the malicious candidate then s/he may click on 'OK' button to turn on the JavaScript and as a result the attack will be launched. Otherwise the attack will not work on his/her machine.

Third Party Verifiers. As mentioned in Section 3.1.1, auditing plays an important role in the Helios application. Contribution of third party trustees to audit and verify the voter's ballots may help to detect the alteration of its contents which is similar to what has been suggested in Helios 3.0 (see Section 5.1). However, if those systems use web-based technology to display the results of verification, the voting approach will be discussed in [18] should be used.

Malware Analysis of Candidacy Statements. The e-vote administrator can analyze the candidacy statements to check whether they contain malware or malicious

JavaScript. However, the attacker can upload an innocent document to his/her website. So, the administrator will not become suspicious. On the other hand, the malicious PDF document maybe emailed to voters by endorsers or a third party.

Using hack-free dedicated hardware As pointed out by a referee, one solution is to use a certified hack-free device that allows to check the result of the hash function. Current versions of Helios do not come with such a device.

Avoiding Helios Let us consider an alternative solution to Internet e-voting. "Code Voting" has been proposed in [14]. We briefly explain its ideas and discuss its advantages and disadvantages.

A voter needs to receive by post or other out-of-band channel a list with a unique code for for every candidate. To vote the voter just enters the code received corresponding to his/her candidate of choice.

At first glance our attack can barely work, as the client-side interface of the system is completely different and makes launching a client-side attack against such system more difficult, if not impossible. It seems a hacked machine cannot change the vote to another one (even a random one). Code Voting also protects the anonymity (privacy) better than Helios (see Section 8 for more details on privacy concerns). However, there are assumptions, which must be satisfied to guarantee these properties.

To guarantee privacy, one needs to guarantee there is no collaboration between the postal service with the returning officer. Moreover, if the party that generates these codes is corrupted, integrity can be undermined (e.g., using ballot stuffing). One could observe that there is no inherent verifiability to this scheme. (One can compare this with the verifiability Helios 2.0 provides/does not provide in practice, see from Section 4.3 on.)

8 Future Work and Generalization

We first discuss potential extensions of our attack. The attack can be adapted to target more computers of voters, e.g., to include machines running Mac and UNIX operating systems. Although Firefox is platform-specific, the extensions are platform-independent. Adobe Reader is available for several operating systems and some of these versions are also vulnerable to buffer overflow attacks. This implies the attacker can install the same Browser Rootkit on above platforms. On these platforms a different payload should be executed after the buffer overflow.

The second generalization would target voters using Internet Explorer (IE). Browser Helper Object (BHO) is a DLL module used to extend and customize IE. Like Firefox extensions, BHO can fully access an HTML Document Object Model (DOM) tree and monitor user's navigations.

Besides attacking the integrity, the Browser Rootkit can be further extended to mount an attack against the *anonymity* of the voters in Helios. *The extension can be designed to capture the name of the selected candidate before the encryption of the ballot and the email of the voter, at the authentication stage.* By developing a client email, using the SMTP service, the malicious extension can email the recorded information to the attacker from the voter's browser. Another option, as pointed out by [38], is that the malicious extension sends the information to the attacker using HTTPS connection, so nothing can be detected from the communication. *Note that the Helios 3.0 remains vulnerable to this type of attack on privacy.*

In our text we assumed that the candidate performs the attack and our attack focuses on exploiting Adobe Acrobat/Reader vulnerabilities to launch an attack. We came to the conclusion that this is the most realistic way to perform the attack, since Helios allows URLs pointing to the candidacy statements, which can be assumed to be in a PDF format. However, the attack could easily be launched by anonymous supporters of a candidate. They could use Social Engineering techniques to get the voter's machine infected, e.g., using mail or luring a voter into opening a PDF file written by a supporter. The attacker can also develop malware, such as a virus, a worm, etc., to install the malicious extension and distribute it using a USB device, or a DOC file.

Another generalization is that in our attack, we assume that the voter's vote is modified into one for a particular candidate. Other attacks include to modify a vote into a random one [38]. This may prevent the favorite candidate to win and might be harder to detect. The attacker may use the techniques mentioned earlier to install the malicious extension on the voter's machines. For a discussion on the security of Helios 3.0, see [18].

9 Conclusion

This paper demonstrates the feasibility of hacking a preliminary version of Helios. Only two weeks were spent on developing this attack. Due to this limited effort, our attack is quite platform dependent. However, with the right incentives real world hackers may work on a broader attack affecting more platforms. Indeed, botnets have shown the progress hackers have made.

Although Helios 2.0 was modified into Helios 3.0 due to our attack, several issues remain. As pointed out by an anonymous referee, it is not clear how many voters will exploit the new security mechanisms (see Section 5).

We therefore leave it to the reader to decide whether the fundamental difference between non-verifiable and verifiable internet e-voting systems, pointed out in Section 2.1, has any practical relevance. Moreover, Helios privacy (anonymity) protection remains its Achilles' heel (see Section 8).

Although many engineers have argued against using internet voting for large political elections, several countries (e.g., Estonia [34], Finland [1] etc.) are already moving towards this. In the light of this, we believe only history will be able to judge whether our feasibility study was an overly cautious warning, or whether, to quote Bollyn [13], internet voting will be the "Death of Democracy."

10 Acknowledgment

We thank Stelios Erotokritou and Rebecca Wright for permission to cite unpublished work. We thank anonymous referees for suggestions on how to improve on the attack and Josh Benaloh for useful discussions. We thank Mohamed-Ali Kaafar at INRIA Grenoble - Rhône-Alpes for his feedback and support. A part of this work was done while the second author was in part funded by EPSRC EP/C538285/1 and by BT, (as BT Chair of Information Security).

References

[1] About electronic voting in Finland. `http://www.vaalit.fi/sahkoinenaanestaminen/en/yleistietoa.html`.

[2] The e-vote 2011-project. `http://www.regjeringen.no/en/dep/krd/kampanjer/election_portal/electronic-voting.html`.

[3] Implementing electronic voting in the UK. - UK Local Governmernt `http://www.communities.gov.uk/archived/general-content/localgovernment/implementingelectronicvoting/`.

[4] Official State of Geneva e-voting site. `http://www.geneve.ch/evoting/english/welcome.asp`.

[5] The International Association for Cryptologic Research (IACR), August 2009. `http://www.iacr.org/elections/eVoting/presentations.html`.

[6] Ben Adida. Helios: Web-based Open-Audit Voting. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[7] Ben Adida. Helios Voting System Blog, August 2009. `http://blog.heliosvoting.org/`.

[8] Ben Adida, Oliver de Marneffe, Oliver Pereira, and Jean J. Quisquater. Electing a University President using Open-Audit Voting: Analysis of real-world use of Helios. *EVT/WOTE'09, Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*, August 2009.

[9] Andrew W. Appel, Maia Ginsburg, Harri Hursti, Brian W. Kernighan, Christopher D. Richards, Gang Tan, and Penny Venetis. The new jersey voting-machine lawsuit and the avc advantage dre voting machine. *EVT/WOTE'09, Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*, August 2009.

[10] BBC News. Chaotic polling problems lead to calls for e-voting. `http://news.bbc.co.uk/2/hi/technology/10102126.stm`.

[11] Josh Benaloh. Simple verifiable elections. In *EVT'06: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, pages 5–5, Berkeley, CA, USA, 2006. USENIX Association.

[12] BigPulse. Online voting software and services, August 2009. `http://www.bigpulse.com/`.

[13] Christopher Bollyn. Death of Democracy or May the Best Hacker Win.

[14] David Chaum. Surevote: technical overview. in: Proceedings of the Workshop on Trustworthy Elections (WOTE'01), August 2001. presentation slides. `http://www.vote.caltech.edu/wote01/pdfs/surevote.pdf`.

[15] Stephen Checkoway, J. Alex Halderman, U Michigan, Ariel J. Feldman, Edward W. Felten, Brian Kantor, and Hovav Shacham. Can dres provide long-lasting security? the case of return-oriented programming and the avc advantage. *EVT/-WOTE'09, Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*, August 2009.

[16] Michael Clarkson, Stephen Chong, and Andrew Myers. Civitas: A secure voting system, August 2009.
`http://www.cs.cornell.edu/projects/civitas/`.

[17] Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade. *DARPA Information Survivability Conference and Exposition,*, 2:1119, 2000.

[18] Yvo Desmedt and Saghar Estehghari. Weaknesses of Helios 3.0. In preparation.

[19] Yvo Desmedt and Saghar Estehghari. Hacking Helios and Its Impact, August 2009.
`http://rump2009.cr.yp.to/1b884ce772d84af05f0f4b07bf019053.pdf`.

[20] Yvo Desmedt and Yair Frankel. Threshold Cryptosystems. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989.

[21] Mozaill Developer Center, August 2009.
`https://developer.mozilla.org/en/Extensions`.

[22] Digital Civil Rights in Europe. Electronic Voting Machines Eliminated In The Netherlands. `http://www.edri.org/edrigram/number5.20/e-voting-machines-netherlands`.

[23] Security Focus. Adobe Acrobat and Reader collab ‘getIcon()’ Javascript Method Remote Code Execution Vulnerability, August 2009.
`http://www.securityfocus.com/bid/34169/discuss`.

[24] Security Focus. Adobe Acrobat and Reader Multiple Arbitrary Code Execution and Security Vulnerabilities, August 2009.
`http://www.securityfocus.com/bid/27641/discuss`.

[25] Security Focus. Adobe Reader ’util.printf()’ Javascript Function Stack Buffer Overflow Vulnerability, August 2009.
`http://www.securityfocus.com/bid/30035/discuss`.

[26] Stefan Frei, Thomas Duebendorfer, Gunter Ollmann, and Martin May. Understanding the Web browser threat. Technical Report 288, TIK, ETH Zurich, June 2008. Presented at DefCon 16, Aug 2008, Las Vegas, USA.
`http://www.techzoom.net/publications/insecurity-iceberg/index.en`.

[27] Rop Gonggrijp and Willem-Jan Hengeveld. Studying the Nedap/Groenendaal ES3B voting computer: a computer security perspective. In *EVT’07: Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, pages 1–1, Berkeley, CA, USA, 2007. USENIX Association.

[28] Andy Greenberg. The year’s most-hacked software, December 2009.
`http://www.forbes.com/2009/12/10/adobe-hackers-microsoft-technology.-cio-network-software.html`.

[29] Stuart Haber, Josh Benaloh, and Shai Halevi. The Helios e-Voting Demo for the IACR, May 2010.
`http://www.iacr.org/elections/eVoting/heliosDemo.pdf`.

[30] Harri Hursti. Diebold TSx Evaluation: Critical Security Issues with Diebold TSx. *Black Box Voting*, May 2006.

[31] Trusteer Inc. Flash security hole advisory trusteer, August 2009.
`www.trusteer.com/files/Flash_Security_Hole_Advisory.pdf`.

[32] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an Electronic Voting System. In *IEEE Symposium on Security and Privacy*, pages 27–. IEEE Computer Society, 2004.

[33] Mike Ter Louw, Jin Soon Lim, and V. N. Venkatakrishnan. Extensible Web Browser Security. In Bernhard M. Hämmerli and Robin Sommer, editors, *DIMVA*, volume 4579 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2007.

[34] Epp Maaten. Towards remote e-voting: Estonian case. In *Electronic Voting in Europe - Technology, Law, Politics and Society*, volume 47 of *LNI*, pages 83–100. GI, 2004. July 7th–9th 2004, Bregenz, Austria.

[35] Mozilla. Mozilla’s Q1 2010 Analyst Report - State of the Internet, August 2009.
`http://blog.mozilla.com/metrics/2010/03/31/mozillas-q1-2010-analyst-report.-state-of-the-internet/`.

[36] University of Connecticut. Adder: an Internet-based Voting System, August 2009. http://cryptodrm.engr.uconn.edu/adder/.

[37] Microsoft PressPass. Microsoft Statement Regarding Download.Ject Malicious Code Security Issue, August 2009. http://www.microsoft.com/presspass/press/2004/jun04/0625download-jectstatement.mspx.

[38] Anonymous Referee.

[39] Ryan Roemer, Erik Buchanan, Hovav Shacham, and Stefan Savage. Return-Oriented Programming: Systems, Languages, and Applications, 2009. In review.

[40] StatOwl.com. Operating Systems Market Share, 2010. http://statowl.com/operating_system_market_share.php.

[41] TrueBallot, August 2009. http://www.trueballot.com/trueballot.aspx.

[42] Julian Verdurmen. Firefox extension security. Master's thesis, Radboud University Nijmegen, the Netherlands, January 2008.

[43] W3Schools. Web Statistics and Trends, OS, August 2009. http://www.w3schools.com/browsers/browsers_os.asp.

[44] Jun Xu, Zbigniew Kalbarczyk, Sanjay Patel, and Ravishankar K. Iyer. Architecture support for defending against buffer overflow attacks, 2002.

A Appendix

A.1 The Attack: Details

On Election Day, the voter uses the provided link in the invitation email to access the election webpage. After the voter starts the voting process, a question is displayed on the screen where he/she should choose one of the candidates. In the meantime, he/she may decide to visit Alice's candidacy statement. By clicking on the provided link (next to her name), the voter is redirected to her webpage. After the page is loaded, a JavaScript function is called that checks whether the current browser is Firefox and the Adobe Acrobat plug-in is installed on the web browser. If these conditions are satisfied, it loads the PDF document.

The loaded PDF document contains a malicious JavaScript function which is called upon opening of the file. This function is designed to wait for 10,000 milliseconds (until the voter has read some parts of the file). This delay partially covers the traces of the attack and reduces the probability that the voter becomes suspicious about the maliciousness of the PDF. Then the function calls `Collab.getIcon()` function (from Adobe JavaScript API) by passing a long argument to it. If the Adobe Acrobat/Reader running on the machine has the specification assumed earlier, then buffer overflow vulnerabilities in this function are exploited. This gives an opportunity to Alice to remotely execute an arbitrary program on the victim's machine by running a 'Download and Exec' payload. This type of payload is able to connect to a specified host server over the Internet, download an exe file and execute the file on the client's machine. For the purpose of this attack, the executable injects a malicious extension into the Java Console extension. This is done by copying the extension's essential files into a folder and altering chrome.manifest to load those files. Since Firefox needs a restart to load changes made to Java Console, the program closes the browser. Also, it terminates Acrobat's process, as the software does not function properly after the buffer overflow attack.

The voter should restart the browser and start the election from the beginning. This time the UI files of the malicious extension are loaded into the browser instead of the Java Console's. The extension starts listening to the browser's history, until the voter visits the election webpage. Then it injects malicious JavaScript code snippets into the DOM tree of the Helios application. This is done only once because Helios has employed a 'single-page application' technology, where clicks cause background actions rather than full-page loads [6]. The injected JavaScript codes override some JavaScript functions in the application. In other words, the injected JavaScript functions will be executed instead of the original ones.

The voter may decide to read Alice's candidacy statement again. In order to prevent a repetition of the attack, the overridden function changes the link to her webpage, where an innocent PDF document is opened instead.

On the question page, if the voter votes for the non-malicious candidate (Bart Preneel), the ballot is processed by the overridden function and altered to Alice. However, in order to fool the voter, on the confirmation page Bart Preneel is displayed as the chosen candidate. In the encryption stage, the voter may decide to audit the ballot. Each candidate is given a number, so Alice (the first candidate) corresponds to 0 and Bart Preneel to 1. The plaintext part of the audited ballot is also modified, to show that the desired answer is chosen (in this election the answer equals to 1). However, above alterations are insufficient. Indeed, if the voter decides to verify the ballot, then the system shows the "Encryption doesn't match" message as the result. To prevent the voter be able to detect this, the malicious extension was modified. This was achieved by reprogramming the JavaScript function used to check the actual encryption. The corresponding hacked JavaScript now always outputs "Encryption is verified", regardless whether the encryption is correct or not. After the vote is submitted, the voter receives a confirmation email that does not contain the name of the candidate and as a result the voter may not realize that he/she has voted for another candidate.