Lightweight security for Internet polls

Alessandro Basso, Francesco Bergadano, Ilaria Coradazzi, Paolo Dal Checco
Dipartimento di Informatica, Università degli Studi di Torino, Corso Svizzera 185 Torino, Italy

Abstract. Is it possible to implement practical Internet Polls that fulfill even the weakest security requirements? The technology available today would lead to a negative answer, because of the following practical constraints: standard, unmodified browsers are used, it is not economically possible to distribute certificates or even just user names and passwords, users connect from different workstations, possibly behind firewalls, proxies and address translation nodes. In this paper, we define an innovative notion of Internet Poll security, namely "Security against Massive Falsification", and we present a method that we consider to be secure with respect to this definition. We discuss the security properties of the method with respect to existing techniques, and then propose a public challenge for testing the strength of our claim.

1 Introduction

Internet polls could be an important means for collecting information on people's preferences and opinions "faster and cheaper". The quality of such information, however, is indeed very low today, due to the complete lack of security – existing Internet poll systems may be attacked by automatic programs, and the poll results may be overturned in minutes. In this paper we will define minimum security requirements for providing poll data that can be meaningful, even the weakest way. We also propose an Internet poll system that does not require user registration, accepts votes originating from the same IP address, and yet satisfies said security requirements.

2 Related work

In this section we describe the present state of the art in the area of Internet Polls. We first define what an Internet Poll is and the related functionalities. We then proceed considering the structure of a typical program used to perform automatic and multiple voting, to better understand the protection techniques used to prevent such software from being used. We conclude this section comparing a selection of existing polling services.

2.1 Poll systems

An Internet poll is a Web application with the purpose of giving users an opportunity to express their opinions regarding the topic of the poll. For this reason a lot of web sites offer to their users the possibility to insert polls in their personal web pages. Therefore, one can easily create and manage a set of polls at the same time. In order to achieve this task, an Internet user who needs such a service must sign up at the poll system site and create his own poll, compiling the special fields on a proper form. Then the user has to copy and paste the code provided by the system on his web page.

Normally, a poll system is a dynamic application, which therefore takes advantage of the client/server paradigm. We can identify two main components, the client side and the server side.

The client side consists of a graphical interface, which shows to a voter the title, message and possible choices of a poll. This interface is created by mean of a web page displayed in a generic Internet browser. An Internet user can select a voting option and express its own preference by clicking on the “vote” button or see the poll’s results.

The server side (in the form of a script, like a CGI or PHP script) has to receive the data form the client, in order to bring operations to completion. These data consist of the user’s preference and must be uniquely identified to avoid mistakes in the voting procedure.

Let us consider an example, to better understand this concept. The example poll has three different choices:

- Choice one
- Choice two
- Choice three

When a user selects one of these choices and clicks on the “vote” button, an identifier of the choice is sent to the script responsible for the voting procedure. Let us imagine of choosing the “Choice two”: the id sent to the server is “B” and the script can obviously determine our preference.

2.2 Software for multiple and automatic voting

Writing some software that is able to perform multiple and automatic voting is fairly simple (see, e.g., the discussion on cookie poisoning in [12]). The basic idea is to analyze the HTML page which contains the poll, seeking for the text of possible choices. Such a page is normally downloaded by the browser in a temporary-cached file; therefore the search can be easily performed by looking for that file.

The second step is to determine the id which has to be sent to the voting script together with the other necessary parameters, like cookies and poll’s name.

The third and last step is to write a program to simulate the voting procedure for any given number of times, by providing the voting script with the id and the other needed information. Such a kind of program must take in account that the poll application might use a protection system, like the ones described in the next paragraph. In that case, each time it sends a vote, the program has to pay particular attention in bypassing the protection mechanism of the poll.

2.3 Protection techniques against automatic voting

Because of the stateless nature of the HTTP protocol [7], there is no way of securely storing any operations performed by voters in different sessions. Therefore, creating a polling system may turn into a difficult task, since it can be a problem to understand whether a voter has already expressed his opinion. There are however different techniques to check and prevent multiple voting. Unfortunately, at the moment the security level offered by such techniques is very low because it is still possible to exploit the weaknesses of these methods to automate the voting process.

At the present time, the techniques used by existing poll systems to avoid multiple voting are:

- IP locking
- Cookie-based methods

In the IP locking method, the check is performed by verifying the IP address of voters. Therefore an IP address that has already voted, cannot send another vote for a fixed time period. Unfortunately, that also means that some categories of users might not be allowed to vote for a poll. In particular:

- People connected to the Internet through a dynamic IP address assigned by the ISP. In this case, the same IP address can be subsequently assigned to two different computers. Thus, the second pc will be excluded from the vote.
- People connected to the Internet from multi-user workstations.
- People connected to Internet from a LAN (Local Area Network) which uses a NAT translator. The purpose of the NAT mechanism is to hide internal addresses of the LAN and, in the mean time, allowing all computers of the net to surf the web using only one public IP address [EF94, Bel00].
- People connected to Internet behind a Proxy. As in the NAT situation, different users have the same IP address, so they are all excluded from the voting but the first.

Sometimes, the IP locking technique is combined with an improvement called Browser header, which is based on the control of HTTP packets exchanged between client and server. In particular, it checks the request headers issued from the browser and verifies whether consecutive packets are identical. If so, the vote is not considered valid. In other words, the same IP addresses must not have the same HTTP headers.

This protection allows multiple users connected form a LAN behind a NAT or Proxy to vote, but its security is still weak. Indeed, an attacker might use the same IP address and alter the content of HTTP headers each time he wants to vote. Since the possible combinations of modified headers are quite many, it is easy to produce a very large number of different headers.

Another interesting improvement of the IP locking technique is based on time intervals. Two subsequent votes are considered invalid if the time interval between them is less than a fixed value. This schema can be easily bypassed using an automatic voting program, setting an appropriate time interval for consecutive votes.

Therefore, it avoids the main problem of the IP locking method, allowing users in an environment protected by NAT or Proxy to correctly express their preferences. It does not guarantee, however, that every user will be able to participate to the poll, since it a difficult operation setting the time interval to the appropriate value for every situation.

The second technique used to protect web polls from automatic and multiple voting is known as Cookies method. A cookie is a text file that contains the information exchanged between client and server [8]. This information interchange is necessary to the server to be able to verify whether a client has already voted. Data stored into a cookie can be various and comprehend:

- the cookie name and associated value
- the URI for which the cookie is valid.
- the validity domain.
- the validity time expressed in second.

A cookie used for management of multiple votes also contains a flag to indicate whether the client, holder of the cookie, has already voted. The main problem about using cookies concerns the fact that users can disable them or delete them making possible the creation of a program designed to alter votes' results.

3 The Method

In this section we consider the method which characterizes our solution for the problem of multiple automatic votes. First we introduce some premises related to our solution, in order to better understand it. Then we proceed describing the characteristics of the basic idea.

3.1 Human-Machine discrimination

Proving that one is a human to another human is fairly simple. This problem can be easily solved by mean of "The Turing Test", defined by Alan Turing in 1950 as the foundation of the philosophy of artificial intelligence. Basically, a human judge asks a set of questions to both a machine and a human being and discriminates among them, depending on their answers.

On the other hand, proving that one is a human to a computer is much more difficult. Indeed, it is required what M. Blum, L. A. von Ahn, and J. Langford call "Completely Automatic Public Turing Test to tell Computers and Humans Apart", or CAPTCHA, in [1]. CAPTCHA is a set of tests which can be graded from computers and designed in such a way that humans can pass but computers fail.

A generic CAPTCHA is characterized by the following properties:

- The test can be automatically created.
- It can be easily and quickly passed by a human.
- A test must be suitable for a very large majority of humans, with few exceptions.
- Virtually no machine can pass it.
- It must be able to resist automatic attacks even considering future technology improvements and even if the test's algorithm is published.

In [5], Allison L. Coates, Richard J. Fateman and Henry S. Baird present an application of CAPTCHA which involves the extensively studied gap in image pattern recognition ability between human and machine vision systems. As clearly showed in their studies, nowadays the gap in ability between human and machine vision is wide and is only slowly narrowing. This fact, which could be a serious problem in some sectors of Computer Science, on the contrary is perfectly suitable in satisfying the growing need for automatic methods to distinguish between human and machine users on the Web.

It has been experimentally showed that, by means of a correct choice of some well determined parameters involving the quality of an image, it is possible to generate human-legible images containing some text which are illegible to several of the best present–day optical character recognition (OCR) machines. These images of printed text are characterized by low quality and quite strong degradation but still readable, with little or no conscious effort, by almost every person literate in the Latin alphabet and with some years of reading experience. However, such schemes may have vulnerabilities as shown in [10].

Coates, Fateman and Baird [5], in agreement with the discussion in [13], define a series of parameters that are usually considered problematic and cause of errors in the optical character recognition process:

- Thickened images, so that characters merge together.
- Thinned images, so that characters fragment into unconnected components.
- Noisy images, causing rough edges and salt–and–pepper noise.
- Condensed fonts, with narrower aspect ratios than usual; and Italic fonts, whose rectilinear bounding boxes overlap their neighbors'.

Our solution for the problem of automatic voting has been developed by applying these concepts to the polls context, as we show in the next paragraph.

3.2 A solution for the problem of automatic voting

As the reader can easily see, apparently there is no way to prevent poll results' falsifications, if software like the one described in paragraph 2.2 are used to attack an Internet poll system. Indeed, in such a system the following statement can be assumed as true:

> *as long as the information used as a voting parameters is kept in a machine-readable form on the client, an attacker can use it to generate arbitrary automatic votes.*

It should be clear now that the only way to avoid an attack for falsifying poll's results is to prevent the client from storing sensitive information in a machine-readable form. Therefore, we have to consider what possible techniques can represent information in a way that is easily understood by humans but difficult to comprehend by a machine. Our idea for getting a higher security level in the area of Internet polls derives directly from the concepts stated in the previous paragraphs. We though that a way for preventing automatic programs from voting relies on the ability of our system to generate a test which can be easily solved by human-voters but impossible for machine-voters.

The test that we consider is slightly different from a typical CAPTCHA. Indeed, while the latter is mainly meant for distinguish between humans and computers, ours is designed to make the voting procedure impossible to a non-human user.

The main idea is to include the poll's choices into a runtime-generated image in order to remove them from the web page sent to the client's browser. In this way, we prevent computers from analyze the HTML page containing the poll for finding correct bindings between choices' text and their ids.

The image which contains the choices is created on the server each time a user requests a poll page and the order in which each choice is displayed on the screen is randomly chosen. The choices' order must be stored somewhere in order to allow the server to understand the mapping between the id of the preference chosen by a voter and the correct choice. We name such an order "mapping scheme".

It merely consists of the number of the choice which is first in the image. Therefore, when a voter asks for a generic poll page, the server determines the mapping scheme for that request and generates the image in the following way:

1. It gets the n choices of the poll in their standard order.
2. It randomly generates a number x, between 1 and n.
3. It creates the image by starting with the choice pointed by x. The second choice is the one that follows it in the standard order and so on.
4. It stores the mapping scheme into a cookie, after encrypting it with a symmetric algorithm.

Basically, the standard choices' order is rotated using one randomly chosen choice as pivot element.

This procedure is executed each time one asks for a poll page; therefore the choices are always presented in an unpredictable order. When one votes, the id of the selected preference is sent to the server along with the cookie containing the mapping scheme. The server is then able to reconstruct the binding between the chosen id and the proper choice and therefore it can correctly increment the counter of the selected preference.

The image containing the poll's preferences is characterized by some important features to prevent OCR programs from being able to read the text inside the picture, therefore breaking the protection scheme.

In particular:

- Its font must be carefully chosen, preferring those which are known to be difficult to recognize by an OCR.
- Its quality cannot be too much poor, otherwise the image becomes unreadable even for humans.
- It should contain some pixels which have a different color shade from the others in order to prevent the identification of the image itself by means of hashing functions. These pixels are randomly chosen inside the picture as well as their shades.

Our solution includes also the classic cookie-based protection technique, so that it is even more difficult to bypass it. The voting flag is stored inside the cookie, along with the mapping scheme. This fact makes its modification more complicated because it is not possible just altering the flag value, since it is encrypted (see also [11]). Therefore, one has to delete the entire cookie to be allowed to vote again. Clearly, just the deletion of the cookie is not sufficient to permit an automatic voting. Indeed, in such a case, the attacker is only able to perform random votes, since the mapping scheme is unknown. Thus, the only achievement is to increase the number of votes of each choice in a constant way, because each preference has an equal probability of being chosen.

In particular, if the number of choices is n, each of them has a probability $p = 1/n$ of being the first in the image. In this situation, the only effective automatic its id and vote for it each time. Therefore, every choice has the same possibility of being chosen, making the automatic voting attack a mere increment of each choice result, without affecting the distribution of votes which remains the same.

However, there is still a method to alter the results of the poll by means of automatic voting. Indeed, if one copies the cookie content before voting, he can then reuse it for the following votes.

This reused cookie contains always the same mapping schema; therefore it is easy to vote for the chosen preference for an unlimited number of times.

To prevent such an attack, we store the content of the cookie in order to check it out at every voting request. Since it is created to be unique, we refuse multiple votes which use the same cookie content.

4 Security properties

In this section we discuss the security properties of the presented Internet poll system. First, however, we must discuss which level of security is practically achievable for polls in an open Internet scenario, where clients and remote networks are not known and cannot be modified. Then, we will define a notion of security that could apply for polls in such a scenario, namely "security against massive falsification". Finally we will argue that our proposed poll scheme satisfies this notion of security and propose an open challenge to test our claim.

4.1 Practical security for open Internet polls

Practical Internet polls depend on the following facts:

- Clients (Browsers) cannot be customized, nor initialized or modified in any way.
- It is practically impossible to distribute passwords or secrets of any kind for the authentication of voters.
- Biometric approaches are either too expensive or too imprecise, or both. Techniques based on keystroke dynamics [4] may be applicable, but require too much typed text.
- Remote network architecture is unknown and may not be modified, it may include proxies and network address translation (NAT).
- Clients may change their IP address due to user mobility and local address assignment (e.g., DHCP).

- Users may connect from more than one client machine.

Given the above constraints, it is immediately clear that a high level of security, such as that required for voting, is not a practical goal. Even under extremely restrictive poll schemes, such as the ones based on IP locking, users can simply change their workstations, connect from a different network, and vote twice. Since they do not own cryptographic tokens and are not given passwords, it is impossible to authenticate them and avoid multiple voting. One actually wonders whether even very weak security properties apply. We try to propose one such notion below.

4.2 Security against massive falsification

We must consider the following important fact:

> *multiple voting may be programmed - one may write an ad-hoc client program explicitly designed to kill the target Internet poll scheme.*

This is easily understood with cookie-based schemes, as shown in paragraph 2.2. IP-locking schemes are more robust with respect to automated voting and even manually repeated voting in general, because the number of IP addresses available for an individual is limited. Since Internet poll schemes are generally implemented over HTTP and over the TCP connection oriented transport [Bel89], IP address spoofing is not possible in general [1].

However IP-locking schemes are highly restrictive, and prevent many kinds of legal voting, as explained in paragraph 2.3.

All techniques that do allow legal votes, meaning actually all known approaches except IP-locking, are vulnerable to automated attacks. This means that not only false and repeated votes are possible, but also that such votes may be generated by a program in very large numbers. The result of the poll may then be changed completely in minutes. We call this attack “massive falsification”.

Based on the above discussion, we are now able to define “security against massive falsification” for Internet polls: a*n Internet poll scheme is* ***secure against massive falsification*** *if (1) Internet users are always allowed to participate in the poll and (2) there is a significant cost in the programming of massive multiple voting.*
The first requirement means that users may vote when they have the right to do so. This basically eliminates IP-locking, where users may be unable to vote just because somebody else had voted with the IP address they are now using. The second requirement states that it is impossible or difficult to write a program that generates an arbitrary number of repeated votes, or that it will be possible to eliminate such repeated votes in a post-processing phase. 'Massive falsification' is then avoided, but it

[1]If an IP address is spoofed, it will be impossible, in principle, to establish a connection from that client to any server, since the second message in a TCP three way handshake will not reach the originator of the first message. However, it should be noted that in some particular situations it is still possible to realize a form of 'blind' TCP/IP spoofing that would be critical for poll schemes based on IP locking. For further information about this matter, see [VKI99, BG00, Lud03].

will be possible to send a limited number of false votes, e.g., but manually voting more than once from different client machines.

4.3 Security properties of order-based polls

We claim that our proposed order-based poll scheme satisfies the requirements of security against massive falsification:

1. Users may vote if they have not voted already because there is no IP-locking, one may vote any number of times from the same IP address. Hence, users behind a proxy or NAT may vote normally. Users with dynamic IP assignments are guaranteed to be able to vote.
2. The order of the options as displayed by the poll window is random and unpredictable by the client. One must therefore look at the window to know the order and therefore the option to be selected. If one knew the option order every time, one could generate the vote automatically and achieve massive falsification. However, the only way to know option order is to interpret the image sent to the poll window. The image is explicitly designed to be difficult to interpret automatically, and yet be readable by human eyes.

4.4 Challenge

We now propose an open challenge, where readers and the Internet security community are invited to try to break our scheme. At the address www.certimeter.com/pollchallenge/thepoll one finds a poll, where it is possible to vote one of two possible choices: (A) Italian cuisine and (B) British cuisine.

The poll is initialized with 500,000 votes in favor of A and 500,000 votes in favor of B. We therefore have 50% of people in favor of Italian cuisine and 50% in favor of British cuisine. At the beginning of every month the poll is reinitialized to those values. Thus, challengers have one month of time to break to scheme, but they can try again. The goal of the challenge is to cause the poll result to be at least 90% in favor of British cuisine, before the end of the month.

5 Conclusions

In this paper we presented an Internet Poll system that has been implemented and is based on an unpredictable ordering scheme for poll choices, so that the user is forced to look at the Web page before sending her vote. This is, to our knowledge, the first proposed Internet Poll system that (1) is secure against programmed attacks, (2) is not based on IP locking and (3) does not require user registration.

Acknowledgments

The authors thank Regione Piemonte, through grant "Sinapsi" for supporting this work. The prototype and the challenge setting have been developed with the cooperation of the Department of Computer Science of the University of Turin and the participating company, Certimeter S.r.l.

References

1. M. Blum, L. A. von Ahn, and J. Langford, *The CAPTCHA Project, Completely Automatic Public Turing Test to tell Computers and Humans Apart*, www.captcha.net, Dept. of Computer Science, Carnegie–Mellon University, November 2000.
2. S.M.Bellovin, *Security Problems in the TCP/IP protocol suite*, Computer Communication Review, AT&T Bell Laboratories, 1989.
3. Steven M. Bellovin, *A Technique for Counting NATted Hosts*, AT&T Labs Research, 2000.
4. F. Bergadano, D. Gunetti and C. Picardi, *User Authentication through Keystroke Dynamics*, ACM Transactions on Information and System Security (ACM TISSEC), 5(4), 2002.
5. Allison L. Coates, Richard J. Fateman and Henry S. Baird, *Pessimal Print: A Reverse Turing Test*, Sixth International Conference on Document Analysis and Recognition (ICDAR 2001), Seattle, Washington, September 10-13 2001.
6. K.Egevang, P.Francis, *The IP Network Address Translator (NAT)*, RFC-1631, May 1994.
7. R.Fielding, J.Mogul, H.Frystyk, L.Masinter, P.Leach, T.Berners-Lee, *Hypertext Transfer Protocol HTTP 1.1*, RFC-2616, June 1999.
8. D.Kristol, L.Montulli, *HTTP State Management Mechanism*, Request for Comments RFC-2965, October 2000.
9. Albert Ludwing, *Ip Address Spoofing*, Univ. Freiburg, www.ks.uni.freiburg.de/inetwork/papers/ipspoofingPaper.pdf
10. G. Mori and J. Malik, *Breaking a Visual CAPTCHA*, UC Berkeley, http://www.cs.berkeley.edu/~mori/gimpy/gimpy.html
11. Joon S.Park, Ravi Sandhu, AreeLatha Ghanta, *RBAC on the web by secure cookies*, security XIII:Status and prospects, Kluwer, 2000.
12. Eran Reshef, Izhar Bar-Gad, *Web Application Security*, Sanctum Inc., settembre 2000.
13. S. V. Rice, G. Nagy, and T. A. Nartker, *OCR: An Illustrated Guide to the Frontier*, Kluwer Academic Publishers, 1999.
14. Marco de Vivo, Gabriela O. de Vivo, Roberto Koeneke, Germinal Isern, *Internet Vulnerabilities Related to TCP/IP and T/TCP*, ACM SIGCOMM Computer Communication Review, January 1999.