A Framework to Strengthen Password Authentication Using Mobile Devices and Browser Extensions

Peipei Shi, Bo Zhu and Amr Youssef
Concordia Institute for Information Systems Engineering
Concordia University, Montreal, Quebec, Canada
{pe_sh, zhubo, youssef@ciise.concordia.ca}

Abstract

Shoulder-surfing, phishing and keylogging are widely used by attackers to obtain users' sensitive credentials. In this paper, we propose a framework to strengthen password authentication using mobile devices and browser extensions. This approach provides a relatively high resilience against shoulder-surfing, phishing and keylogging attacks while requires no change on the server side. A prototype implementation of the proposed approach and its security analysis are also provided.

1. Introduction

Text-based passwords are still the most common form of authentication adopted by online service providers such as banks, online stores, and social networking sites. Generally, a combination of a valid username/userid and a password is enough to obtain full access to a given account. Moreover, in many scenarios, the username/userid is not considered as sensitive information and can be easily leaked. Consequently, the protection of users' accounts relies heavily on the protection of users' passwords which makes passwords the hot target of identity theft criminals.

There are many techniques that are used by attackers to obtain and exploit passwords. Among these techniques, phishing has been one of the most popular and effective approaches. It is widely adopted by attackers all over the world, partially because it is easy to launch: a forged website and/or a fake e-mail is usually enough to trick novice users and acquire their passwords and/or other sensitive information.

In addition to phishing, keylogging programs are also widely used by adversaries in order to collect sensitive information. Sometimes, people have to enter their sensitive information using untrusted machines (e.g., using public PCs in Internet cafes), which makes them particularly vulnerable to keyloggers. While computers in public places have a higher probability of being infected by keylogging programs, home computers can also be infected by such programs due to the flooding of spywares and botnets. As the name implies, shoulder surfing is used to obtain passwords, PINs or other sensitive information through observation, such as by looking over someones shoulder. It is a simple, yet effective, method to get information especially in crowded places such as coffee-shops where it is relatively easy to stand next to or behind someone without being noticed.

In this paper, we propose a framework aiming at improving the security of online password authentication against shoulder surfing, phishing, and keylogging. We also present a prototype implementation of our design. Our solution is inspired by the idea presented in [15]: instead of sending the user's password, *pwd*, in plaintext to a remote server, we first extract the domain name of the remote server *dom* and then calculate the hash value *hash(pwd, dom)*, which is sent to the remote server as the password. Here, domain name *dom* works as a salt in this hash function. This design does not require any change to the server side. Moreover, since domain names are unique, the hash values of passwords for any two domains *hash(pwd,* dom_1*)* and *hash(pwd,* dom_2*)* are very unlikely to collide. As a result, passwords acquired by phishing sites are not useful for login at any other domain. Furthermore, in order to protect against keyloggers and shoulder surfing, users do not type in their passwords. Instead, these passwords are securely stored on the users' mobile devices (such as PDAs and cell phones) and are sent directly to the PCs through the Bluetooth link. The rest of this paper is organized as follows. The related work is reviewed in Section 2. Details of our proposed solution and the prototype implementation are presented in Section 3, followed by the security analysis of the proposed framework in Section 4. Finally, we conclude our work in Section 5.

978-1-4244-9991-5/11/$26.00 ©2011 IEEE

2 Related Work

Several recent countermeasures to defend against phishing attacks have been proposed from both academia and industry. A popular method is to develop/find distinguishing features of phishing sites and warn the user if the features extracted from the accessed site matches these previously determined features. This approach has been implemented in several browser toolbars such as TrustBar [7], SpoofGuard [1], Netcraft Toolbar [12], eBayToolbar [2] and SpoofStick [16]. These heuristics toolbars detect malicious URLs, and some of them provide more information about the domain they are visiting, such as the true domain name, location and country name, the creation date of the domain. Users are warned when these toolbars consider or judge the visited site as fraud. Because of the inherent imprecision of these heuristic methods, some users are bothered since they are required to make the final decision whether they should trust the toolbar advice or not. Moreover, many factors considered by these toolbars are transparent which opens a window for attackers to adapt to them in order to bypass the underlying heuristics. Additionally, 13-54% of users, who are warned by anti-phishing toolbars, ignore the warning and continue their browsing [18].

Kirda and Kruegel [9] presented a Firefox extension, AntiPhish, that aims to protect users against spoofed website-based phishing attacks. This solution is inspired by automated form-filler applications which usually have a master password to protect sensitive information. Whenever the user enters information into text field elements of type *text*, *password* and *textarea*, AntiPhish checks the list of previously captured values. If each value is identical to the one just provided by the user, the domain is checked with the previously stored one. In other words, this AntiPhish tracks the sensitive information of a user and generates warnings whenever the user attempts to submit sensitive information to a website which is considered to be untrusted by the application.

Parno *et al.* [13] proposed a mechanism, Phoolproof, that leverages a trusted device to perform mutual authentication between client and server. This solution eliminates reliance on perfect user behavior, thwarts Man-in-the-middle (MITM) attacks and protects users' account information against keyloggers and most forms of spywares. Phoolproof is a two-factor authentication system in which the trusted device works as an additional authenticator. Therefore, the attacker must compromise the trusted device and the user's credentials in order to impersonate the user. A long-term secret is required to be stored on the mobile device, and most cryptographic operations are conducted on the trusted mobile device. As mentioned by the authors, this design is vulnerable to session hijacking attacks and needs modification on the server side.

Mannan *et al.* [11] presented a simple approach, using a mobile device, to strengthen the authentication process from an untrusted computer. In their design, the user's long-term secret is cryptographically separated from the untrusted computer which means that a client PC can only access temporary secrets even though most computations are performed in this untrusted PC. The user's long-term secret (typically short and of a low-entropy) is input through a trusted personal device such as a mobile phone or a PDA. The long-term secret is never stored in this personal device. The trusted personal device only provides a user's long-term secret to the client PC after encrypting the secret with a pre-installed and valid public key of a remote server. This proposed solution is intended to protect sensitive information from various attacks such as keylogging, phishing and pharming attacks, as well as to provide transaction security to foil session hijacking. However, in order to achieve these goals, a modification of the server side is mandatory.

3 Proposed Solution and Prototype Implementation

3.1 Main Idea

Nowadays, computers and mobile devices with Bluetooth functionality are very common. Moreover, in many practical scenarios, these mobile devices can be consider as trusted devices or at least far more trustworthy than PCs. Based on this observation, we argue that it is better to shift the sensitive login burden from untrusted computers to these relatively more trusted devices (e.g., cell phone).

As mentioned above, the password sent to the server is constructed by hashing the user supplied password, *pwd*, with a salt value obtained from the login site. The objective of using a salt in the password hashing process is to generate different passwords for different domains. Thus, the generated password will be resistant to spoofing websites because the domain of a spoofed site is very likely to be different from the genuine one. Different possible values can be considered as candidates for the salt such as the domain name of the site hosting the current page and the SSL certificate of the target domain. As mentioned in [15], different choices have their own advantages and disadvantages. However, based on the availability and security concerns, the current domain seems to be the most suitable choice to be used as a salt. For instance, using SSL certificate has several drawbacks such as being hard to replicate manually. It also suffers from compatibility problems when the target site has no SSL certificate.

Some popular websites have multiple domains for different countries or territories. For instance, eBay has *ebay.ca* for Canada and *ebay.cn* for China. So when a user creates an account using *ebay.ca* as a salt, it is not possible for the

user to login to the site for China. On the other hand, if a user creates an account at *ebay.ca*, usually the user should seldom login to *ebay.cn*. Even if the user has to do so, she can just create a new account for the specific domain name in this situation. Yahoo and Wikipedia have a different way to organize their domain names. For example, the domain name for Yahoo Canada is *ca.yahoo.com* while the domain for Yahoo China is *cn.yahoo.com*. In this scenario, we can just extract the two top-level domains and use them as the salt for all the domains that match "*.yahoo.com". Fortunately, most sites only have one consistent domain.

Another challenge that we faced is that some sites have special restrictions on the chosen passwords in order to force users to avoid choosing weak or easy to guess passwords. Some of these restrictions require that the password should contain at least one non-alphanumeric character and/or at least one uppercase character. In order to overcome this problem, the hash encoding algorithms, running on the mobile device should satisfy various restrictions imposed on the servers' passwords (e.g., by including at least one uppercase, one lower case, and one numeric character in its output.)

3.2 System Architecture

Our prototype implementation consists of three components: a J2ME program (MIDlet) installed on the mobile phone, a Java desktop application on the PC and a Mozilla Firefox extension.

The first component of the system is a MIDlet application running on the user's mobile phone. This program, which can run on any J2ME-enabled mobile phone, receives the domain name of the current login page, from the Java application running on the PC, through Bluetooth communication. Then it searches this domain name in a lookup table which contains all the websites that the user wants to secure using this framework. If a match is found, the user is required to input the corresponding username and password. Users can either choose to input their credentials manually into this MIDlet, or call the saved credentials in the mobile phone. The security implications of both options are further discussed in Section 4.2. Before being sent to the Java application on the PC, the username remains the same while the password is hashed using the following pseudo-random function (PRF):

$$PRF_{pwd}(dom) = hash(pwd, dom), \qquad (1)$$

where the user's password *pwd* is used as the key while the website's domain *dom* is used as input to the PRF. In other words, as mentioned above, *dom* is used as the hash salt.

The second component of our solution is a Java application that has to be installed on the computer running the browser. It receives the URL of the current login page from the Firefox extension, and then forwards it to the MIDlet through the Bluetooth link. Once the user sends the credentials to this Java application, it forwards them to the Firefox extension.

The third component of our solution is a Firefox extension which extracts the URL of current login page, and sends it to the desktop Java application. It then waits for the response from the desktop Java application. After receiving the username and hashed password from the desktop Java application, this extension passes them to the web server. Our solution does not require any modifications to the server side.

Figure 1 illustrates the main components of the prototype and information flow among them. In what follows we provide a more detailed description for these components.

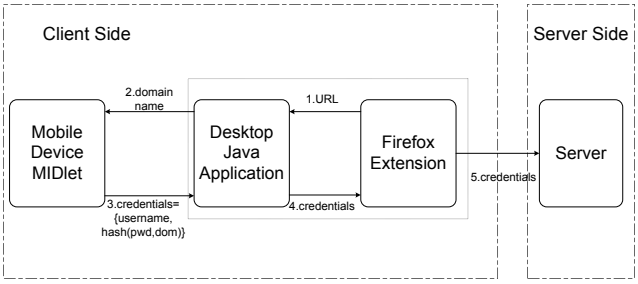

Figure 1. Architecture and information flow of the proposed solution

3.2.1 Firefox Extension

Similar to many add-ons of the Firefox Browser, our extension is developed in XML User Interface Language (XUL) and JavaScript. The developed extension works as a toolbar added to the Firefox Browser, which is activated by the user who presses a "start" button whenever the user wants to login to a remote server using our solution. The functionality of this extension is to extract the current URL of the login page and send it to the desktop Java application as an asynchronous JavaScript and XML (AJAX) request. It then waits for the reply from the desktop Java application. Once the response is received, this extension extracts the username, hash value of the password from the response string, and fills them in the login page. In the password field, the star symbols ($*$) appear instead of the symbols and characters of the typed password. When the extension tries to fill the username field automatically, it searches certain id values for the username from the source code of the login page. Most websites use common ids for the user-

name field, such as *Email* (Google Gmail), *username* (Yahoo Mail) and *email* (Facebook). If this extension can not identify the username field from the source code of the login page, then the user is prompted to manually input the username into the username field. Throughout our experiments, we did not face this problem for the password field that has the statement $type = \text{“}password\text{”}$.

3.2.2 Desktop Java Application

The desktop Java application works as an information relay station between the Firefox extension and the MIDlet. We can separate this application into two parts: http server module and Bluetooth module. We were not able to use socket communications in order to establish a communication link between the Firefox extension and this Java application because JavaScript does not support sockets for security reasons. Instead, AJAX technique is chosen in the solution for this specific communication requirement. In order to adopt AJAX, one part of desktop Java application is developed as a simplified http server which opens the http service at the localhost of the computer on the unused port 18000 (the selection of port 18000 is arbitrary.) This http server keeps checking for the AJAX request from the Firefox extension. After receiving the request, it parses it and sends the domain name to the Bluetooth module. Whenever this http module receives the credentials from the Bluetooth module, it forwards them to the Firefox extension as an AJAX response. The Bluetooth module is developed using bluecove 2.1.1 which is a Java API for Bluetooth. This module has two threads: a client thread and a server thread. The client thread (discovery thread) searches for new Bluetooth devices. Whenever a new device is found, a service search is performed to the found device. If the service is found successfully, a connection is established to this device. On the other hand, the server thread waits for new connections from other Bluetooth devices. After the connection is constructed with the MIDlet of the mobile phone, the communication channel is open. When this module receives the domain name of the login page from the http server module, it forwards it to the MIDlet. Then, it keeps checking for the response of the MIDlet. Whenever the Bluetooth module receives a response from the MIDlet, it passes it to the http server module.

3.2.3 MIDlet

A MIDlet is an application conforming to the Mobile Information Device Profile (MIDP) standard. In order to develop MIDlets for mobile phones, some infrastructure is needed. Sun provides a wireless toolkit for Connected Limited Device Configuration (CLDC), namely, Sun Java Wireless Toolkit 2.5.2 (WTK 2.5.2). Our MIDlet is divided into the following three modules: Bluetooth module, Hash module and Record Management module. Similar to the desktop Java application, this Bluetooth module also has two threads: a client thread and a server thread. When this module receives the domain name from the desktop Java application through the Bluetooth channel, it searches the domain name in a lookup table which contains all the websites that the user wants to associate with this solution. If the domain name matches one item in the table, the user is prompted to input the corresponding username and password. Otherwise, an error message is displayed. Two options are then provided to the user. If the user decides to input the username and password manually, the record management module will not be activated. However, the user can also call the saved credentials using the record management module. The password is then hashed using SHA-256 which returns a 256 bit message digest. The generated result is Base64 encoded which yields a 28 character string. Then the string is shrunk to a pre-specified length (e.g., 10). The length can be adjusted depending on the server side requirements. Finally, the generated hashed password and the username are sent to the desktop Java application through the Bluetooth channel.

In order to use our system, the Firefox extension and the Java application are assumed to be installed in the computer running the user's browser. In many scenarios, users may not have the access privileges to install two modules on the used computers (e.g., computers in Internet cafe). To address this issue, a secondary option is provided by the MIDlet through a Graphical User Interface (GUI) to help users generate their hashed passwords where the user only needs to input the password and the domain name. Then, the MIDlet computes and displays the hashed password to the user who can then input it in the password field of the login page using the keyboard of the browser's computer.

4 Security Analysis of the Proposed Solution

In this section, we investigate some security threats associated with our solution.

4.1 Security of the Bluetooth Link

Since the hashed version of the password is sent from mobile phone to the PC through the Bluetooth channel, the security level of this part definitely affects the overall security of our solution. When the Bluetooth connection between mobile phone and computer is set up without activating the link encryption, it can be easily eavesdropped. Moreover, it is also easy for an attacker to replace the original payload data with other data. Thus, when our solution is deployed, link encryption between these two units must be activated.

In Bluetooth, encryption is perfomrmed using the stream cipher E_0 [17]. The core of E_0 is built on four independent linear feedback shift register (LESR) and a finite state machine as a combining circuitry which introduces sufficient nonlinearity. A detailed analysis of this algorithm is beyond the scope of this work. Instead, in here, we summarize some known attacks on E_0. Jakobsson and Wetzel [8] pointed out two attacks on the E_0, the first of which is of $O(2^{100})$ time complexity while the other one requires $O(2^{63})$ time effort using 2^{34} observed symbols. Fluhrer and Lucks [3] presented an attack on E_0 using observed keystream and the public knowledge of the encryption mechanism. This attack can recover the initial state of E_0 in $O(2^{68})$ using 2^{43} observed or known cleartext. However, since the required number of known symbols is very large compared to the symbols in a frame, the proposed attack can not directly reveal the link key. Krause [10] reported a more powerful attack on E_0 in terms of the required number of symbols. This attack has a time complexity of $O(2^{77})$ while only requires 128 known symbols. An improved correlation attack is presented by Golic [4] which achieves $O(2^{70})$ time complexity using less than one frame of known symbols. Courtois [5] also presented an attack which requires $O(2^{49})$ time complexity if $2^{23.4}$ bits are available to the attacker.

While the above attacks on the E_0 cipher are of complexity less than $O(2^{128})$, which is the complexity of exhaustive search through the key space, the effort required by all these attacks is impractical and does not present any threat to our implementation.

In order to avoid the high time complexity of a direct attack on E_0, one may try to attack the Bluetooth link during the pairing procedure. Prior to Bluetooth 2.1 specification, Bluetooth technology was somewhat sensitive to passive and active attacks on the pairing procedure. However, in Bluetooth 2.1 + EDR, a new feature called Secure Simple Pairing (SSP) is introduced in order to increase the level of security. Now, the latest specification of Bluetooth is Bluetooth 4.0 which also supports the SSP.

In summary, if link encryption is activated between the mobile phone and the PC, the Bluetooth communication channel can be considered as a secure channel for our solution.

4.2 Storing Credentials on Mobile Phones

As mentioned before, users can choose to store their credentials in the mobile phone and call the saved credentials whenever needed. Otherwise, users may choose to input the credentials to the MIDlet manually. If users choose the former, it is easy for them to conduct the authentication process since they do not need to be bothered by typing in this information. However, saving credentials on the mobile phone may have some security implications, e.g., if the mobile phone is lost or stolen. A master password which protects unauthorized access to the mobile device may mitigate this risk. On the other hand, the manual credential entry option may not be convenient for some users, especially when they are not used to the cell phone small keypad. In conclusion, users can choose either one of these options depending on their specific needs and security requirements.

4.3 Phishing Attacks

Using the proposed solution, each site would receive a hash value of the password with domain name as a salt. Moreover, since the domain names are unique, the hash values of passwords for any two domain hash(pwd, dom_1) and hash(pwd, dom_2) are almost certain to be different. Consequently, when a phishing site receives the hash value of the user's password , the attacker can not use this value to directly hijack the user's account. However, it is possible for the attacker to launch an offline dictionary attack. A straightforward method to mitigate this problem is to use the so called *slow hash function*, which is already widely implemented in the UNIX system to increase the computing resources needed to launch a dictionary attack [6, 14].

4.4 Shoulder Surfing Attacks

According to our design, users can choose to store their credentials on the mobile phone. Then, when they are required to input these credentials, they can simply retrieve them. In this case, it is impossible for shoulder surfers to obtain the users' passwords. On the other hand, some users may prefer not to store their credentials on the mobile phone and input their passwords manually, which opens a small window for shoulder surfers. However, the task of the shoulder surfer is much harder in this case because the keypad of the mobile phone is much smaller than the PC. Also, many mobile phones use one key to control three or more characters. Thus, it is reasonable to assume that the probability to mount a successful shoulder surfing attack is very slim in this scenario. Another possible chance for shoulder surfing is when users have to use the secondary option (see Section 3.2.3) to generate the hashed passwords and enter them using the keyboard of the PC. In this case, the hashed passwords might be leaked to the shoulder surfer, especially when the shoulder surfer has some recording capabilities.

4.5 Keylogging Attacks

In the normal setup of our solution, the keyboard of the computer is not used to input any sensitive information. Instead, all the sensitive user's credentials are sent directly by the mobile device to the Firefox application. Even if the keylogging program has some features such as screen

grabbing, it will still not be able to acquire any useful credentials from the users. However, our solution might be vulnerable to sophisticated keyloggers that are able to monitor communications on certain ports or webform grabbers that can get web submissions. In order to defend against this threat, we can apply the one time password (OTP) technique to the proposed solution. In particular, we can replace $hash(pwd, dom)$ by $hash(pwd, dom, time)$.

5 Conclusion

We have proposed a framework to strengthen the authentication process of the Internet services. The proposed solution provides improved security against some prevalent attacks such shoulder surfing, phishing and keylogging attacks. An advantage of the proposed solution is that it can be directly deployed to current services without the need to change the server side.

References

[1] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell. Client-side defense against web-based identify theft. In *Proc. of the 11th Annual Network and Distributed System Security Symposium (NDSS '04)*, February 2004.

[2] eBay. ebay toolbar. http://pages.ebay.com/ebay_toolbar/.

[3] S. R. Fluhrer and S. Lucks. Analysis of the E0 cryptosystem. In *Proc. of the 8th Annual International Workshop on Selected Areas in Cryptography (SAC 2001)*, volume 2259, pages 38–48, August 2001.

[4] J. D. Golic, V. Bagini, and G. Morgari. Linear cryptanalysis of bluetooth stream cipher. In *Proc. of the International Conference on the Theory and Applications of Cryptographic Techniques: Advance in Cryptology (Eurocrypt 02)*, volume 2332, pages 238–255, April 2002.

[5] J. D. Golic, V. Bagini, and G. Morgari. Fast algebraic attacks on stream ciphers with linear feedback. In *Proc. of the International Conference on the Theory and Applications of Cryptographic Techniques: Advance in Cryptology (Eurocrypt 03)*, volume 2729, pages 162–176, May 2003.

[6] J. Halderman, B. Waters, and E. W. Felten. A convenient method for securely managing passwords. In *Proceedings of the 14th International World Wide Web Conference (WWW 2005)*, pages 471–479, May 2005.

[7] A. Herzberg and A. Gbara. Trustbar:protecting (even naive) web users from spoofing and phishing attacks. Cryptology ePrint Archive, Report 2004/155, 2004.

[8] M. Jakobsson and S. Wetzel. Security weaknesses in bluetooth. In *Proc. of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA (CT-RSA01)*, volume 2020, pages 176–191, April 2001.

[9] E. Kirda and C. Kruegel. Protecting users against phishing attacks with antiphish. In *Proc. of the 29 Annual International Computer Software and Applications Conference (COMPSAC 2005)*, pages 517–524, July 2005.

[10] M. Krause. BDD-based cryptanalysis of keystream generators. In *Proc. of the International Conference on the Theory and Applications of Cryptographic Techniques: Advance in Cryptology (Eurocrypt 02)*, volume 2332, pages 222–237, April 2002.

[11] M. Mannan and P. van Oorschot. Using a personal device to strengthen password authentication from an untrusted computer. *Financial Cryptography and Data Security*, 4886:88–103, February 2007.

[12] Netcraft. Netcraft toolbar. http://toolbar.netcraft.com/.

[13] B. Parno, C. Kuo, and A. Perrig. Phoolproof phishing prevention. *Financial Cryptography and Data Security*, 4107:1–19, February 2006.

[14] N. Provos and D. Mazieres. A future-adaptable password scheme. In *Proc. of the Annual Conference on USENIX Annual Technical Conference*, June 1999.

[15] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *Proc. of the 14th Conference on USENIX Security Symposium (USENIX 2005)*, pages 17–32, April 2005.

[16] SpoofStick. Spoofstick home. http://www.spoofstick.com/.

[17] Wikipedia. E0 (cipher). http://en.wikipedia.org/wiki/E0_(cipher).

[18] M. Wu, S. Garfinkel, and R. Miller. Users are not dependable - How to make security indicators to better protect them. Talk presented at the Workshop for Trustworthy Interfaces for Passwords and Personal Information, June 2005.