Context Sensitive Password

Anthony Y. Fu[1,2] **Robert C. Miller**[1] **Greg Little**[1] **Min Wu**[1]

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, MA, USA

[2]Department of Computer Science, City University of Hong Kong, Hong Kong SAR

1. INTRODUCTION

There is a long history of people typing passwords into HTML textboxes to verify their identities. However, textboxes are easy to spoof by phishing websites. We propose a twist on the traditional password input field which is more secure, while remaining easy-to-use. This approach involves adding a positional component to passwords, relative to a context string.. This method achieves stronger security without requiring the user to install any special plug-ins to their web browsers. Hence, they retain the stronger security on any computer a user chooses to log in to. This method is also easy to implement, requiring no additional work for password verification modules. Context sensitive passwords are also easy to save into traditional password management databases. We discuss the general design of this idea in Section 2, and present some examples of context sensitive passwords in Section 3. Then we cover related work in Section 4, and conclude our work in Section 5.

2. CONTEXT SENSITIVE PASSWORD DESIGN

2.1 Generating Context Sensitive Passwords

The generation of Context Sensitive Passwords has several steps. First, the user creates a username and a secure code. The secure code is like a lesser password which prompts the server to provide a customized password input field. Note that it doesn't need to be as secure, since the server can generate fake customized password fields based on *any* secure code, so an attacker will have difficulty knowing whether they entered the correct secure code.

Next, the user creates the customized password field. This consists of entering a string of text (the context string). Finally, the user modifies the context string by inserting some number of substrings to create the Context Sensitive Password. The Context Sensitive Password is then saved in the server side database as a traditional password.

2.2 Using Context Sensitive Passwords

When a user wants to log into a system protected by Context Sensitive Password, they should provide their username and secure code. The system should then provide the user with a password field containing the context string, and prompt the user to make the appropriate insertions. If the context string is a meaningful string, then we believe it will be easy for the user to remember where to make the insertions.

Now let's analyze the interaction with a phishing site. First, it asks the user for their username and secure code. Next, it needs to display some context string, but it doesn't know which one. This should alert the user to something suspicious since they may not know how to enter their password into the provided context string. If they insert the substrings anyway, the phisher will still not know the true insertion points.

If the attacker knows the secure code, and the N insertion strings, and the context string has length L, then the attacker has only a $(C_L^N)^{-1}$ chance of inserting the strings correctly on each attempt with the real server. However, some users may have difficulty remembering more than 3 insertions. But even if they have only 1 insertion, the attacker still has only a *1/L* chance of inserting it in the correct place. This can be made fairly small by making the context string large. To help users create large context strings, we can customize a paragraph of a novel or a news column as a context sting. Even with long strings, users should still be able to remember where they made their insertion, since the string is semantically meaningful. Users can also reuse a small set of context strings for different services in order to lower the memory burden.

2.3 Verification

We simply check whether the string resulting from the insertions to the context string matches the Context Sensitive Password associated with the username and secure code.

3. EXAMPLES

3.1 Textual Version

Suppose Alice has an account in an e-banking system, PseudoBanking. Her User Name is "Alice", Secure Code is "MIT", Context string is "ALICELOVESBOB", and insertion string is "CSAIL" which is inserted between the letters "C" and "E" in the word "ALICE".

When Alice wants to login PseudoBanking, the login UI requests her to input her username and secure code. Alice inputs "Alice" and "MIT" into the two text boxes. The system then provides Alice with a password box that looks like Figure 1a. It is a special password box which has been filled with Alice's personalized context string. Alice remembers that she needs to modify the string by inserting "CSAIL" between the letters "C" and "E" so she moves the curser to this "E" to start inputting "CSAIL". The system then uses the password in Figure 2b to login, while the real appearance of the password box is shown in Figure 2c. To improve the security, Alice can use two insertion strings, like "CS" and "AIL". She might insert these as shown in Figure 1d.

Except for the "man-in-the-middle" attack, phishers have to guess the positions that Alice is using to insert the substrings. On the other hand, when Alice is trying to log into a phishing website, the phishing website does not know the correct context string. Hence, Alice does not know where to input her substrings. Even if she enters them anyway, at some random location, the phisher still cannot make a successful attack without knowing the insertion points. Phishers have to guess what the insertion points are. When Alice uses one insertion string, the attacker has a 1/14 chance of inserting it correctly, and when she uses two (assuming the attackers knows what they are), the attacker has a 1/91 chance of inserting them correctly.

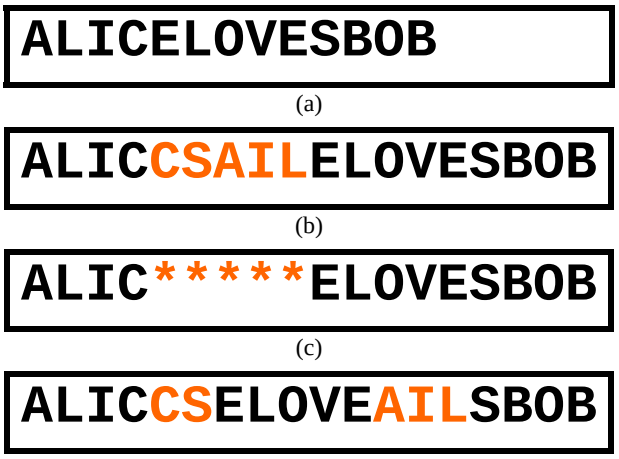


Figure 1. Context Sensitive Passwords

3.2 Icon Version

Alternatively, we can use icons to replace characters in the context string. There are 26×2 letters + 10 numbers + 32 symbols + 1 space, giving us 95 icons to use. Figure 2 shows some mapping samples of icons to characters (the icons are from [1]). The complete table has 95 mappings. We replace characters in the context strings of Figure 1a, b, and c with icons, and we show the results in Figure 3a, b, and c respectively. Obviously, Alice only needs to find the first "Mango" icon (which is in the middle of the top row) and inserts her string their. The icon version may not be superior to the textual version. However, there could be a way to make a hybrid of the graphical and textual authentication methods that may have some of the advantages of both, and we plan to explore this in future work.

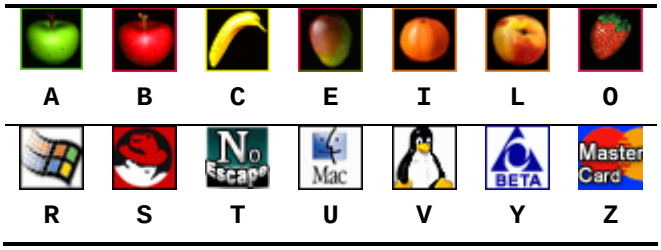
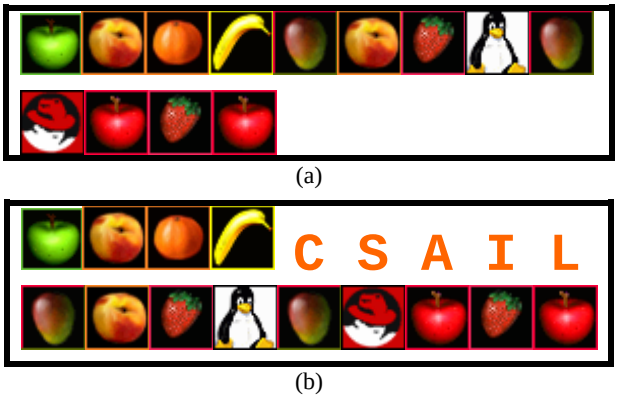


Figure 2. Icon to Character Mapping Samples



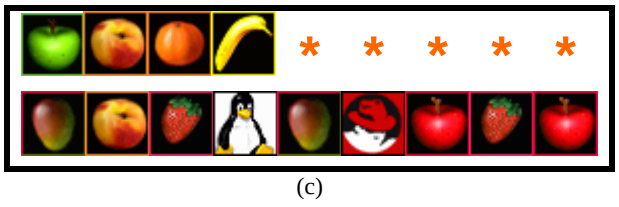


Figure 3. Icon Version of Context Sensitive Passwords

4. RELATED WORK

Min Wu et al show that visual indicators may not work well through user studies [5]. They claim that users are always focusing on the goal to finish their job, and do not remember to check the display of toolbars or indicators every time they log in. PWDHash [3] uses user passwords and domain names to hash the real password for web browsers. Password Multiplier [2] is a similar application which provides stronger cryptographic protection and can be used out of web browsers. Graphical Password [4] is a password authentication method that involves clicking particular spots in an image instead of inputting alphabetical passwords. This method forces users to look at personalized images so that they can click them appropriately. However, to achieve the same level of protection offered by textual passwords, users may need to click an uncomfortable number of times. Another problem with this approach is that it takes time to click all of the images, which exposes the password to people looking at a user's screen. These three methods rely on client plug-ins, which cannot protect users using computers without such plug-ins. In our approach, we can integrate this security into the standard HTML textbox, which does not require a plug-in.

5. CONCLUSION AND FUTURE WORKS

The biggest goal of phishings is to steal users' passwords. We propose a novel password protection method, Context Sensitive Password, to fight against such password stealers. There are three preconditions for attackers to log into Context Sensitive Password protected systems: 1. the attacker must know the username, secure code, and context string; 2. the attacker must know the insertion points; 3. the attacker must know the insertion substrings. They cannot carry out a successful attack if they lack any one of the three preconditions. We present a textual version and an icon version of Context Sensitive Password. They are essentially equivalent. However the different forms of representation could affect usability. We would like to carry out related user studies to compare these approaches. We would also like to provide an easy-to-use programming API for Context Sensitive Password to benefit future work in this area.

REFERENCES

[1]. AlienEntity Free Icons, http://www.entity.cc

[2]. Halderman, J., Waters B., and Felten, E., *A Convenient Method for Securely Managing Passwords*, in the 14th International World Wide Web Conference

[3]. Ross, B., Jackson, C., Miyake, N., Boneh, D., Mitchell, J. *Stronger Password Authentication Using Browser Extensions*, in the 14th Usenix Security, 2005.

[4]. Wiedenbeck, S., Waters, J., Birget, J., Brodskiy, A., Memon N., *Authentication using Graphical Passwords: Eeffects of Tolerance and Image Choice*, in SOUPS 2005

[5]. Wu, M., Miller, R., Garfinkel, S., *Do Security Toolbars Actually Prevent Phishing Attacks?*, in the CHI 2006