Design of a Unicode Password System (Revised November 2008)

David Chan, Jaques Clapauch, Mavaddat Javid, Tony Wong

Abstract— **Password strengthening was successfully realized by increasing the password search. The result was that Unicode passwords written in different languages — even easy-to-remember passwords —have significantly increased strength to withstand attack. A web-based system was implemented to facilitate the study of this design. The password system relies on its interface to encourage users partaking in the study to produce passwords in various languages, with no additional assistance provided. (Specific results here: potential and practical entropy, estimated time-to-crack, usability and psychological acceptability). The results show exceptional success, demonstrating the feasibility and security improvement in applying the design to existing password systems.**

Index Terms— Computer Security, Cryptography, Entropy, User centered design

I. Introduction

Despite their known weaknesses to secure software systems, the cost-efficacy of passwords has ensured their lasting popularity among developers and users alike. Thus, we sought to implement a password strengthening technique based on the non-English character sets to increase security while preserving (and expanding) usability. The goal of this paper is to provide a description of our solution which was to introduce the Unicode Password System — a novel approach that leverages the as-yet untapped international familiarity of non-Latin alphabets, thereby strengthening passwords while maintaining usability to maximize psychological acceptability.

The idea behind this system is based on the assumption that there is a significant and growing computer user base in the world that is not exclusively English-speaking. This is a fair assumption to make, given the growing economies in countries with huge populations, such as China. It is also fairly reasonable to assume that such population would already have the expertise to type in their native language.

A Unicode password system, in its stead, is able to hold up to 65535 characters (even more characters if including the supplementary planes). Thus the purpose of a Unicode password system: to enable the dispersion of character sets enough, so that brute force attacks become nearly impossible.

This system raises the fundamental strength of password systems. The system is only upgrading the allowed characters set – it is not mutually exclusive with existing password strengthening techniques. So, even though dictionary attacks will still be effective, the system can be used in conjunction with various existing and proven techniques to prevent such attacks.

II. Related Work

A. Graphical Passwords

The Graphical password technique allows users to draw graphical symbols instead of typing letters to better remember their passwords. The passwords become the drawings of the users, rather than letters.

Analysis of graphical passwords proved them be easier to remember than enforced secure passwords, and less guessable to hackers, making them thus a better choice than simple passwords.

Even with these properties though, only limited combinations would work for this method if we wish to keep the passwords easily reproducible. Unicode, on the other hand allows a much less limited number of combinations, dependant solely on the language you speak. Unicode also allows a greater number of letters and symbols, than graphics alone.

B. Winrar Chinese Passwords

Winrar supports Unicode for passwords entries, allowing a much greater amount of characters to be included in a password, and allowing a greater amount of protection through a greater amount of characters. This methods allows a much harder method to brute force a password, since the amount of combinations are much greater than normal passwords. It is a formidable method to better protect your password, if you can type in a foreign language.

One of the system's main weakness is the inability to directly type Unicode characters into WinRAR, (must be copy-and-pasted). This significantly cripples the usability and is one of the issues we addressed with our design.

C. Password Strength Enforcer

Password strength enforcers may be found in most websites and allow the user to see how strong their password is, and whether he should truly rely on it. Some websites rely on it to allow or deny you from using a particular password in their mainframe for being too weak.

Tools such as this can be easily integrated with Unicode and aid in getting the users to analyze their own passwords,

[1] Manuscript received December 1, 2008.

Mavaddat Javid (corresponding author to provide phone: 604-913-1922; e-mail: mavaddat@ interchange.ubc.ca).

Tony Wong. (e-mail: tonywongk@gmail.com).

Jaques Clapauch (e-mail: phantomcaller@gmail.com).

and adjusted for the level of competence designers expects for their audiences

D. Ubuntu Unicode Systems

A Ubuntu developer once proposed a Unicode password system, where users could just type ctrl-shift-#### to type in the characters of the password.

This method, as a Unicode password system, provides all the same protections as our password system does, for the entire operating system.

Though the results are the same as in our project, the means to them differ greatly, since it isn't realistic for a person to remember the code of every character, especially since there are so many of them, and would work the same as solely memorizing a great stream of numbers. We chose to design our system for environments where programs to type foreign characters already exist.

III. Solution

A major bottleneck in password systems lies in the user. On the one hand, a typical weak password is highly usable, but presents huge security threats. Conversely, strong and complex passwords are hard to remember, prompting risky behavior such as writing down the passwords or reusing the same password for multiple systems. Our solution is to look beyond the existing methods and increase the strength of passwords without compromising the usability, to design a system that encourages strong passwords feasibly. This is done by expanding the supported character set in passwords to foreign languages, and providing enough visual indicators to encourage users to use a foreign language.

Entropy is an industrial standard measure of password strength. Entropy is calculated by

$$\mathbf{E = L \log_2 C} \qquad \mathbf{(1)}$$

where E is the entropy, L is the password length, and C is the character set. By supporting the entire Unicode character set in our password system, we are drastically increasing the character set and thus the entropy, creating stronger passwords inherently. Granted that a linguistic analysis would shrink the actual entropy by making assumptions about the Unicode usage, this design will still improve password entropy significantly. More detailed analysis of the design's impact on password entropy will be discussed in *Section V*.

A. Design

The Unicode Password System was implemented in a web-based application. This was done for several reasons: First, this allows us to avoid encoding problems – we let the web browsers handle all of that by enforcing UTF-8 using a W3 standard notation. Also, by implementing our design in a web application will allow us to use it in a cross operating system environment as the application will mainly depend on the scripting language the application is using. Finally, Unicode encoding in web-browsers comes standard more often than it does on Operating Systems.

Since the application is web-based, a proper scripting language must be chosen to develop it. Most popular options were PHP and ASP.NET. Both languages are valid environments to work on. In the end ASP.NET was chosen because of its already built-in security features in its framework, such as protection against SQL injections. In PHP, extra code would have been written or third-party code would have been used in order to achieve that functionality. In addition, one of our team members is more proficient in ASP.NET and already has the necessary tools developed for the application, thus speeding up the design process.

B. Construction

The web system consists of two forms, one to create an account and another to allow the user to login. When creating an account, it is clearly stated to the user to use a Unicode password and one that is totally different from the ones they have used before since the password is going to be known by us. The following will explain the two sections of the application; please refer to **Figure 1** for the system workflow.

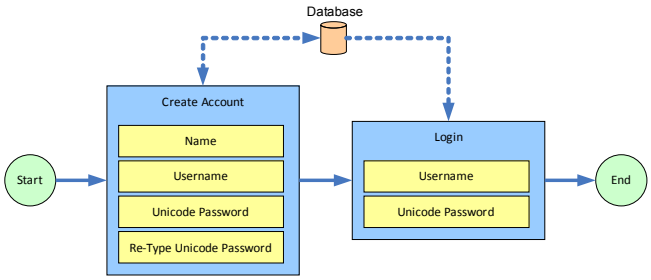

Figure 1 POC System Workflow

1) Create Account

In this form, the user needs to provide his or her Name, Username, and type twice a Unicode Password. The application will then check the database for Username duplicates and prompt the user to use a different Username if the one input already exists. If all validations are passed, the system will then create a new record in the database, with the information given by the user. In addition, the given password is encoded in MD5 hash, and saved into the database to show its feasibility in a real life application.

2) Login

After creating an account, the user can then attempt to login using the Username and Unicode Password he or she provided when creating an account. The system will check the database to match the Username and Unicode Password. When doing the matching, the system will encode the password just given to authenticate into MD5 and then compare it to the one stored in the system. In the case that everything matches, the system will show a Login Successful message to the user.

In order to protect the password from over-the-shoulder attacks while typing it in, the password was hidden by setting

the font colour of the password text the same colour as the background of the textbox; however this method does not allow the user to see the number of character he or she is typing. The user can then highlight the password entered to see the password input. To add input feedback to the user, the application shows the length of the password that the user has already input. The later feature was implemented using javascript coding.

IV. Discussion

A. Flaw

Because the application is web-based, the language encoding will heavily depend on the web-browsers encoding. Even though most web-browsers support Unicode encoding nowadays, the user is able to change its browser encoding manually. In the scenario that the user does change the browser encoding to a different one from Unicode, the system will not be able to parse the Unicode input by the user correctly. We believe that this is a minor issue since there would not be any user wanting the system to not work. In addition, the default encoding is set to Unicode using W3 standard notation and encoding instructions can be provided to the user if having technical difficulties.

B. Test Case - Study

The application was set up in a hosting account we had access to, and the link was distributed among our friends. The population of the study includes people currently located overseas such as Hong Kong and China as well as some that are currently working in a computer-related environment. This allows us to have a better understanding of the behavior of enterprise users working within a culture in which English is not their first language.

The table in the following column shows the passwords used by some of those users with their respective usernames.

User Name	Password
ivan.wong	黃黃黃
mu	パスワード
mikoyung	容容
fishyan	阿魚
juliana	油2342234
alazy	楊思琦
herakleitos	بشقشقشاشق
eric	דב
kammy	柔柔
georgelopez	🏰🏇🤴✊👸🤴🏇🏰

As you may see, all passwords are using Unicode characters and most of them are using the Chinese character set. Moreover, it is found that most users are using a password related to their names such as their last name in Chinese. The last record of the table by shows an interesting password input by georgelopez. The user used a password that can be related to chess pieces ordered the same way it is in back line of a chess board.

We turn now to a closer inspection of the theoretical and practical security increases that can be expected as a result of our system. We herein define a *theoretical security increase* as a gain in the CIA properties of a system resulting from an attacker having to span the total search space of a chosen password length to definitely resolve a password (e.g., a brute-force attack). A *practical security increase*, on the other hand, we define as an incremental gain in defense resulting from an attacker employing character frequency analysis or character-combination frequency analysis to span a sub set of the total search space of a password to breach its security within a high degree of probability (e.g., a dictionary attack). Notice that in both cases we consider cracking the password with certainty, not within an *expected* time. This analysis can be used to yield a parallel analysis of expected password cracking times for a given cracking engine. To determine cracking time, we considered the Deep Crack machine developed by the Electronic Frontier Foundation (EFF), which can crack 90 billion 56-bit keys per second[1], as a hypothetical test case. As we show, our system provides significant increases to both theoretical and practical security, although more careful work needs to be done in analyzing what can be reasonably expected from practical considerations.

Let us first consider theoretical security increase by looking at entropy. For a password of length six, ANSI (222 possible characters) provides a theoretical entropy of $ceil\left(\log_2\left(222^6\right)\right) = 47$. By comparison, a hypothetical Unicode character set (2^{16} = 65,536 possible characters) will provide theoretical entropy of $ceil\left(\log_2\left(2^{6\cdot16}\right)\right) = 96$. This analysis is shown with more detail in Table 1.

Table 1: Theoretical gains in password security from using Unicode character set. ANSI is compared to Unicode in general, and then between English and Chinese, respectively.

Pass Length	1	2	3	4	5	6	7	8
ANSI Entropy for English	7	13	20	26	33	39	46	53
ANSI Entropy	8	16	23	31	39	47	55	62
Unicode Entropy for Chinese	15	30	45	60	75	90	105	120

[1] "EFF DES Cracker Machine Brings Honesty To Crypto Debate". EFF. Retrieved on March 27, 2008, Available HTTP: http://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_descracker_pressrel.html

Unicode Entropy	16	32	48	64	80	96	112	128

Considering the time to crack gained by these numbers, an ANSI password of six characters would take Deep Crack $2^{47}/(90E9 \cdot 56) = 2.375$ seconds to crack. By comparison, an arbitrary Unicode password of six characters would take Deep Crack $2^{96}/(90E9 \cdot 56) = 1.572E16$ seconds (498,147,380 years) to crack. This analysis is considered with more breadth in Error: Reference source not found.**:** Theoretical time (in seconds) to crack arbitrary ANSI and Unicode passwords of various lengths using EFF's Deep Crack machine. ANSI is compared to Unicode in general, and then between English and Chinese, respectively.

Pass Length	1	2	3	4	5	6	7	8
ANSI	4.4E-11	9.8E-9	2.2E-6	4.8E-4	1.1E-1	2.4E1	5.3E3	1.2E6
ANSI for English	2.0E-11	1.9E-9	1.8E-7	1.7E-5	1.6E-3	1.5E-1	1.5E1	1.4E3
Unicode for Chinese	6.5E-9	2.1E-4	7.0E0	2.3E5	7.5E9	2.5E14	8.0E18	2.6E23
Unicode	1.3E-8	8.5E-4	5.6E1	3.7E6	2.4E11	1.6E16	1.0E21	6.8E25

Turning now from theoretical to practical, we consider the actual gains one could reasonably expect from our system given typical passwords in English and Chinese, by comparison. We found that the most frequently used Chinese characters (more than 80%) comprise only 2,965 characters in the language. This means that the practical effect that one could expect from implementing Unicode passwords in Chinese would be dramatically reduced from the theoretical case. However, ~3,000 characters is still significantly more than even the theoretical case for ANSI, resulting in an entropy of $ceil\left(\log_2\left(2965^6\right)\right) = 69$ and a crack time of $2^{69}/(90E9 \cdot 56) = 1.171E8$ seconds (3.71 years). Thus, our system still provides significantly increased security, even within a practical space.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," *Phil. Trans. Roy. Soc.* London, vol. A247, pp. 529-551, Apr. 1955.