Safeguard against *Unicode Attacks*: Generation and Applications of UC-SimList

Anthony Y. Fu Wan Zhang Xiaotie Deng Liu Wenyin
Department of Computer Science, City University of Hong Kong, Hong Kong
anthony@cs.cityu.edu.hk, {wanzhang, csdeng, csliuwy}@cityu.edu.hk

ABSTRACT
A severe potential security problem in utilization of Unicode on the Web is identified, which is resulted from the fact that there are many similar characters in the Universal Character Set (UCS). The foundation of our solution relies on evaluating the similarity of characters in UCS. We develop a solution based on the renowned Kernel Density Estimation (KDE) method to establish such a Unicode Similarity List (UC-SimList).

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**]: General – *Security and protection.*

General Terms
Security, Legal Aspects, and Verification.

Keywords
Unicode, Phishing, and Secure Web Identity.

1. UNICODE ATTACKS
With the popularity of the Internet, people from various countries/regions/cultures flush to the information pool on the Web, and we can find most of the natural languages in the world appearing on the Web. The most successful way of making these characters from different languages correctly displayed relies on the utilization of Unicode. The Universal Character Set (UCS) is a repertoire using Unicode for all characters we may use. The most popular version of UCS uses a 16 bit number to represent a character code. There are a lot of visually similar characters coexisting in the UCS.

The possibility of using similar characters to generate fake domain name using national alphabets was firstly reported in [4] and named *Homograph Attack* as a general concept. Our efforts focus on the survey of UCS which is the most populated character set for the internationalization of Web information, thus we would like to call it *Unicode Attack* in particular. The *Unicode Attack* is more than just faking domain names. We classify possible attacks into three categories: (1) Spamming Attack: malicious people (spammers) may create numerous spams while keeping the appearance of the original email. (2) Phishing Attack: malicious people (phishers) could use visually similar characters to mimic a real Internationalized Resource Identifier (IRI) [3] [5]. Another possibility is that an original Webpage could be mimicked by similar characters such that certain existing Anti-Phishing systems (e.g., [6]) would fail to catch this kind of attack because they must find sensitive word(s) in emails or webpages before actual comparison. (3) Web Identity Attack: malicious people (pretenders) may use similar user names to pretend another user's identity. Many Web based systems (Website, Email, Instant Message, Blog, Wiki, etc.) utilize text string to represent the user names/accounts. There will be no problem if only ASCII characters are permitted to use as a user name. However, if Unicode strings are allowed to represent user names, the system is vulnerable to such attack, especially when the user name is the only way for users to identify each other, and malicious people may successfully gain the victims' trust.

Original String	e	b	a	y
	0065	0062	0061	0079
Fake String1	e	b	a	y
	FF45	0062	FF41	0443
Fake String2	e	b	a	y
	FF45	FF42	0061	0079

Figure 1. Samples of *Unicode attack*. The first string "ebay" is the original part of a real weblink and the rest two strings are mutated/faked ones. Code under each character is the character code in the hexadecimal form.

2. RELATED WORKS
The basic idea of carrying out *Unicode attack* is to generate the mutations of the original (Unicode) string (such as spam content, domain name, and user name, etc.) by replacing similar characters, as shown in Figure 1, and the basic idea of safeguarding the (Web) systems from *Unicode attack* is to evaluate the similarity of the suspected string(s) to the original one(s). A generic methodology for the counter measure against *Unicode attack* has been reported in [5], where the construction of the UC-SimList is considered as a critical part of the Unicode string similarity evaluation.

2.1 About UC-SimList
The first UC-SimList construction method was proposed in [5], however, no detail was given out. UC-SimList is a matrix, which stores the similarities of pairs of characters. To construct UC-SimList, we first need to find the similar characters in UC-SimList_s for a given character, e.g., we find two characters, "a" and "A", are semantically similar to "a" (including "a" itself). We use the semantically similar characters as a source and find all of the visually similar characters of the source from UC-SimList_v, e.g. we find " a " (U+0430), " a "(U+FF41), "A"(U+0491), "A"(U+FF21), "A"(U+0410) are 100% similar to either " a " (U+0061) or "A"(U+0041) in UC-SimList_v. We calculate the similarity of a given pair of characters by multiplying their visual similarity and their semantic similarity.

WWW 2006, May 23–26, 2006, Edinburgh, Scotland.
ACM 1-59593-323-9/06/0005.

We have ever used the pixel overlapping evaluation method to construct UC-SimList [1]. However, the method does not perform well when certain amount of shift of the glyph contour exists. Hence, we use the method of kernel density estimation (KDE) to construct UC-SimList in this paper.

2.2 About KDE

In this approach, a character is represented and measured in 2D kernel densities around the sample points on its contour. Therefore, characters' similarity can be intuitively measured by the similarity of their 2D densities. Kullback-Leibler (KL) divergence [2] is a useful dissimilarity measure of two densities. Let the density of one character be $U(x)$, that of the other be $V(x)$, the dissimilarity between the two characters, $Dis(U,V)$, can be defined as $Dis(U,V)=½\ (KL(U(x),V(x))+ KL(V(x),U(x))$, where U and V can be estimated by Gaussian functions.

3. UC-SIMLIST GENERATION AND APPLICATIONS

The UC-SimList generation includes the UC-SimList_s construction, UC-SimList_v construction, and a process of generating UC-SimList using the two constructed lists.

The construction of UC-SimList_s needs a complete survey on all languages used UCS. In many cases, we can find corresponding replacement to one character, such as "a " to "A" (English in lower-case and upper case), "银" to "銀" (Chinese in simplified-form and traditional-form), "あ" to "ア" (Japanese in hirakana and katakana). The investigation on constructing UC-SimList_s is heavily depending on the language usage and character representation at the semantic level of each language character set in UCS, such as "一" and "壹" (Chinese, stands for "one"), and there is no automatic way to construct it. Hence, UC-SimList_s has to be constructed manually. We constructed the basic version of UC-SimList_s which includes English, Chinese, and Japanese.

The construction of UC-SimList_v needs the character similarity assessment metrics. KDE is an excellent character similarity evaluation as discussed in Section 2.2. Therefore, we use KDE to calculate similarity. Arial Unicode MS font 1.01 is the most complete font we can find, and it covers the largest number of characters among all available fonts in the world. Hence, we choose it for our experiments and the UC-SimList_v construction. Arial Unicode MS 1.01 is a true type font (TTF). Each character is represented with one or several contour(s); each contour comprises quadratic spline(s) (QS) and/or straight line(s) (SL); and each QS/SL is represented with critical points. We retrieve the font information (the critical points of each character) from Arial Unicode MS 1.01 and convert them with sets of points.

We denote N to be the number of points in the converted point sets. The larger the N is, the better the quality of the representation is. Experiment shows that, when N=100, it is sufficiently good to represent the visible characters in the range of U+0000 to U+00FF (ie., equal to the ASCII mapping). Experiment also shows that the process of calculating the KDEs for one character to the rest (2^16-1=65535 characters) takes about 1 hour when N=100, such that we need $\frac{1}{2^{16}-1}C_{2^{16}}^2$ hours (about 3.74 years) to finish the calculation (using a P4 2.4G PC with 1G memory). Hence, it is unrealistic to calculate the complete UC-SimList_v in a short time. As a matter of fact, it will take much longer to calculate if we concern about information lose and use N=200. Experiments shows that N=200 will be good enough to represent all characters in the UCS. We only calculate the characters in the range of U+0000 to U+00FF (ASCII). In fact, these characters are the most frequently used characters and the most probably targeted at to carry out *Unicode attacks*.

The utilization of KDE brings us the accuracy improvement comparing with the pixel-overlapping based assessment [1], where U+9512:银 is ranked the eleventh similar to U+94F6:银 because the two characters' glyphs have some offset to each other, such that the common area of the two characters are reduced. In comparison with the former method, the KDE based assessment method performs much better and rank U+9512:银 to be the second similar character to U+94F6:银.

We also developed an API package for determining whether two Unicode strings/documents are similar based on the constructed UC-SimList. It is available at [1] for free download and usage.

4. CONCLUSION AND FUTURE WORKS

In this paper, we discuss the *Unicode attacks*, which could be generally classified into three categories: (1) Spamming Attack; (2) Phishing Attack; (3) Web Identity Attack. These attacks are essentially based on the coexistence of visual similar characters in UCS. In addition, semantically similar characters in UCS should also be considered seriously. We need to assess the similarity of Unicode strings to evaluate the genuineness of a given one. Hence, one of the most basic cornerstones to detect/discover *Unicode attack* is to construct the UC-SimList. We constructed the prototype UC-SimList_s of English, Chinese, and Japanese. We also proposed an effective symbol similarity assessment measure, KDE, to construct UC-SimList_v. Finally, we put all of the lists and APIs available on the Web [1].

The UC-SimList_s is still under development because (1) there are character sets of many other languages in UCS than English, Chinese, and Japanese, (2) more semantic similarity relationships should be considered as well, for instance, we should consider "一" and "壹" as semantically similar. The UC-SimList_v construction can be done in an automatic way, however, the algorithm proposed in this paper is quite time consuming. Although we have calculated the visible characters in the ASCII, which are most frequently used, the UC-SimList_v should be consummated gradually with further efforts. We can redesign the algorithm to reduce the calculation time in the future.

5. REFERENCES

[1]. Anti-Phishing Group of City University of Hong Kong, http://antiphishing.cs.cityu.edu.hk

[2]. Cover T. and Thomas J., "Elements of Information Theory". John Wiley, 1991

[3]. Duerst M., Suignard M., *RFC 3987: Internationalized Resource Identifiers (IRIs)*, The Internet Society, 2005.

[4]. Gabrilovich E. and Gontmakher A., *The Homograph Attack*, Communications of the ACM 45(2), pp.128, 2002

[5]. Fu A. Y., Deng X., Liu W., *A Potential IRI based Phishing Strategy*, WISE2005, LNCS Vol. 3806, pp. 618 - 619, 2005

[6]. Liu W., Deng X., Huang G, Fu Y., *An Anti-Phishing Strategy based on Visual Similarity Assessment*, IEEE Internet Computing 10(2), pp. 58-65, Mar/Apr. 2006.