The Methodology and an Application to Fight against Unicode Attacks

Anthony Y. Fu[1,2] ayf@mit.edu

Xiaotie Deng[2] csdeng@cityu.edu.hk

Liu Wenyin[2] csliuwy@cityu.edu.hk

Greg Little[1] glittle@mit.edu

[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

ABSTRACT

Unicode is becoming a dominant character representation format for information processing. This presents a very dangerous usability and security problem for many applications. The problem arises because many characters in the UCS (Universal Character Set) are visually and/or semantically similar to each other. This presents a mechanism for malicious people to carry out Unicode Attacks, which include spam attacks, phishing attacks, and web identity attacks. In this paper, we address the potential attacks, and propose a methodology for countering them. To evaluate the feasibility of our methodology, we construct a Unicode Character Similarity List (UC-SimList). We then implement a visual and semantic based edit distance (VSED), as well as a visual and semantic based Knuth-Morris-Pratt algorithm (VSKMP), to detect Unicode attacks. We develop a prototype Unicode attack detection tool, IDN-SecuChecker, which detects phishing weblinks and fake user name (account) attacks. We also introduce the possible practical use of Unicode attack detectors.

Categories and Subject Descriptors

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

General Terms

Security, Legal Aspects, and Verification.

Keywords

Unicode, Spam, Phishing, and Secure Web Identity

1. INTRODUCTION

The globalization of information processing systems pushes for greater use of Unicode, which allows people of different nationalities to represent character based information in their native tongue. Unicode is convenient for many people, but brings with it potential security risks. Many visually or semantically similar characters coexist in the UCS. UCS has a large set of characters. It covers the symbols of almost all languages in the world. Figure 1 shows a set of samples that are similar to the four characters “s”, “o”, “u”, and “p” in Arial Unicode MS font (where the hexadecimal number under each character is the character code of that character). We see that there are at least two other characters in UCS that look exactly the same as the character "s", "o" and "p". There are even more if we include semantically similar characters, e.g., "a" and "A".

People do not usually look into the code of every Unicode string they see to evaluate its validity. This opens a door for malicious people to spoof characters by replacing them with visually or semantically similar (or even visually identical) characters from the UCS. We call this a “Unicode attack”.

We classify Unicode attacks into three categories: (1) *Spam attacks*: Many machine learning techniques used in anti-spam filters view an email as a sequence of characters, and look for patterns commonly associated with spam. If spammers replace characters in these common patterns with similar characters from UCS, they may bypass some of these filters. (2) *Phishing attacks*: Malicious people can use similar characters as replacements in IRI/IDN[8] to create visually similar domain names. Ordinary users may not look into the code under the domain name strings to verify their validity. User studies in [6] show that almost all users judge the validity of a website by the domain name, which makes this attack particularly hazardous. (3) *Web identity attack*: There are countless systems in the world using Unicode strings to represent user names (or accounts). Here again, people tend to identify each other through the appearance of their user names, and not through the underlying Unicode representation of their user names. This allows malicious people to imitate other people by registering a user name that looks like the username of somebody else.

s	s	s	S	S	S	Ȣ
0073	FF53	0455	10BD	FF33	0405	03E8
o	o	o	o	º	ο	o
006F	03BF	043E	FF4F	00BA	FFB7	047B
u	⊔	ս	∪	∪	IJ	ṷ
0075	2294	03C5	22C3	222A	0132	1E75
p	p	p	p	ρ	⍴	P
0070	0440	FF50	01BF	03C1	05E7	0420

Figure 1. Characters similar to “s”, “o”, “u”, and “p” (in Arial Unicode MS Font).

Unicode provides many possible mutations for strings. For example, the simple Unicode string "citibank" has 24(c) * 58(i) * 21(t) * 58(i) * 24(b) * 22(a) * 21(n) * 14(k) -1=263,189,025,791 potential mutations. It may not be surprising that we have found no registration systems (including domain name registrars, chatting applications, BBSes, etc.) which attempt to detect Unicode attacks. As a matter of fact, we have easily registered domain names which are visually similar to several prominent web sites. For instance, we registered, “www.中国银行.com” (“中国银行” is Chinese for “bank of China”, and “国” is a very similar character to “国”), “www.中国銀行.com” (“中国銀行” is Japanese for “bank of China”), which is similar to “www.中国银行.com”. We also registered “www.和記黃埔.com” which is

similar to "www.和记黄埔.com" ("和記黃埔" is Japanese for "Hutchison", a company in Hong Kong). We were able to link these domain names to our Anti-Phishing Group website [3]. Hence there is nothing preventing a malicious person from linking such a domain to a phishing site. This makes the detection of Unicode attacks an important direction for researchers in user interface design, and computer security & privacy to look into.

The concept underlying Unicode attacks is not limited to straightforward character replacement. It can be more complicated. For instance, spammers can add noisy symbols to spam content, as well as replace words with semantically similar words in order to throw off spam filters. In general, the problem of detecting Unicode attacks may require layers beneath and above the character similarly level, including preprocessing to denoise the data in order to know which characters to compare, and higher level language processing to detect similarities at a word or semantic level.

In this paper, we propose a methodology for detecting Unicode attacks. We analyze both the character-character similarity and word-word similarity, use string similarity algorithms to evaluate the similarity of two given Unicode strings, and follow the methodology to carry out experiments in a demo implementation. In the implementation part, we first build a Unicode Character Similarity List (UC-SimList) which can easily retrieve the visual and semantic similarity of any given pair of characters in UCS. We implemented a tool, IRI/IDN SecuChecker, to detect similar/fake IRI/IDN, and demonstrate several possible uses to domain name registrars, user name (account) registrars, and web browser based phishing detection add-ins.

The rest of this paper is organized as follows. Section 2 introduces the related work and background of this research. In Section 3, we address the methodology for Unicode attack detection. In Section 4, we provide a case study of the methodology and discuss the experiments based on a demo implementation to show the effect of Unicode attack detection as well as the associated computation time. In Section 5 we introduce a tool, IRI/IDN SecuChecker, which can be used to detect IRI/IDN based Unicode attacks. We also discuss Web Link Illustrator, a possible improvement for the address bars of web browsers. In Section 6, we discuss problems related to using and deploying these systems. Finally, we present concluding remarks and future work in Section 7.

2. RELATED WORK AND BACKGROUND

We use symbols to record and represent languages for information processing. We used illusive 0/1 strings to represent data in computers at the very beginning. Later, people invented ASCII [1] to encode textual information. This made it easier for people to interact with computers, but it only includes the necessary Latin character symbols. People who use other languages need to install additional character sets to satisfy their requirements, such as GB2312 and HZ for simplified Chinese, BIG5 for traditional Chinese, EUC and Shift-JIS for Japanese, etc. With the development of symbol technology and the requirement of information exchange, people wanted a unified character system to represent all of these characters—hence, Unicode. The most popular version of Unicode (Ver. 2.1 [26]) uses 16 bits and can represent up to 2^{16}=65,536 characters (the most updated version uses 32 bits to represent a character) and can represent almost all standard characters/symbols in the world. Unicode is widely used all over the world. We see it in emails, webpages, resource identifiers, various user (account) registration systems, etc… However, there are too many similar characters in the UCS and it is quite easy to generate numerous similar/fake Unicode strings from a given one to carry out Unicode attacks. Some fake Unicode strings look exactly the same as the original one. We consider this a very dangerous usability and security problem because the computer screen can hide its users from the fact that a string they see may not be exactly what it appears to be. Even before Unicode IRIs, there was a real case involving the website of Industrial and Commercial Bank of China, "WWW.ICBC.COM", which was mimicked by a phisher using a very similar domain name, "WWW.1CBC.COM" ("1" is the number "one" rather than uppercase the letter "i"), to trick people. Similar attacks are also reported in [12], which are called "homograph attack". The potential for abuse increases as Unicode becomes a trend in modern information processing and we call such abuses "Unicode attacks". We classify Unicode attacks into *spam attacks*, *phishing attacks*, and *web identity attacks*, among which phishing attacks turn out to be the most typical and motivating research aspect to fight against.

Phishing is a kind of criminal activity in our modern Internet society where someone forges the webpage of a real company or organization to trick their clientele into divulging sensitive information. Unwary Internet users may be deceived by their scams and follow their instructions to leak private information, such as bank account numbers, passwords, and credit card numbers. Phishing attacks appear to be increasingly common. In the phishing attack prospering period, it was reported that the number of phishing attacks increased 50% each month and 5% of the phishing email receivers respond to them (Anti-Phishing Working Group[3] Phishing Attack Report of July 2004). People are progressively notified or alerted to such scams, however phishers are always trying to use more sophisticated techniques to circumvent detection. This includes making the appearance of their web links and the content of their webpages increasingly similar to the real ones.

Many anti-phishing measures have been carried out. Some address the more general problem of document duplication detection. Along these lines, the collection statistics based approach is proposed by Chowdhury et al. in [5], structure based repetition detection by Nanno et al. in [23], and a general evaluation of different plagiary detection measures is discussed by Hoad et al. in [14], etc. These works focus on plain text documents and use text level features as similarity measurements. Nevertheless, a more effective strategy for phishing detection is proposed by Liu et al. in [20] based on visual comparison of the DOM [27] generated from HTML. Another vision based phishing detection approach is proposed by Fu et al. in [11], where the visual similarity is detected at the image level. Researchers have also sought solutions along different lines. Garfinkel et al [13] and Wu et al [28] have worked on anti-phishing through improving software usability. People also proposed methodologies for anti-phishing from the cryptography view [7] [16], and SSL [25] is now a widely used technology in security critical websites. However, the investigation of web links themselves has been neglected. A survey of similar characters in UCS and the problem of IRI/IDN based phishing was introduced in [10], however no IRI/IDN oriented anti-phishing technique have ever been formally discussed. However, IRI/IDN based phishing attacks could be a critically severe problem for the Internet in the near future. It could be disasterous to delay solving this problem until the usage

of IRI/IDN becomes popular and phishers start using similar characters in UCS to carry out attacks.

In the past, people used IP addresses to access different hosts and resources. Nowadays, most internet resources are identified by ASCII based Uniform Resource Identifiers (URIs) [4]. However, URIs are cumbersome and inefficient for people speaking languages other than English. These people would who would like more familiar character scripts to identify their web resources. Another problem with using ASCII based URIs is that of conflicting URIs when different languages are converted to Latin character representations. Most non-Latin language scripts have a mapping from their characters to Latin characters, such as Chinese Pinyin and Japanese Romaji. People are using these methods to represent URIs in their languages. However, it can happen that different companies or organizations want the same domain name, while they want it for different semantic meanings. This is called URI confliction and semantic ambiguity. Nevertheless, with globalization of information technology, people are using localized operating systems, applications, etc. People are eager to use these systems with their native scripts, including the activity of locating universal resources. IRI/IDN is proposed as a complement to URI. It is a sequence of characters from a subset of UCS. UCS uses 16 bits to represent a character and allows up to 65,536 characters to be represented. This permits most non-Latin scripts to be freely represented in IRI/IDN. This allows Chinese people unfamiliar with English to input "花旗银行.公司" ("花旗银行" is pronounced "Hua Qi Yin Hang" and stands for "Citibank"; "公司" is pronounced "Gong Si" and stands for "Company") rather than "citibank.com". Whereas Japanese people may enter "シテイバンク.会社" ("シテイバンク" is pronounced "Shi Tei Ban Ku", and stands for "Citibank"; "会社" is pronounced "Kai Shya" and stands for "Company") to access the webpage of CitiBank.

These developments make the Internet more accessible as a global resource, but we must address the potential threats in terms of Unicode attacks. These same issues also arise as systems allow Unicode string based web identities, e.g., if email systems allow Unicode accounts, then someone might be able to register an account and pretend to be Bill Gates (such as billg@microsoft.com for MSN Messenger) and send a message to his CEO, "Hi, Steve, I finally decided to open the source code of Vista and give Google a billion dollars. Please do it asap!"

3. METHODOLOGY OF UNICODE ATTACK DETECTION

We propose a general methodology to assess the similarity of a pair of given Unicode strings. We also present a demo tool that can be used by people in academic and industrial areas to do research and develop systems to fight against Unicode attacks.

We organize the methodology from views of string similarity on several levels. The lowest level is character similarity, which includes visual similarity and semantic similarity between two characters. The second level is semantic similarity between words. The highest level is string similarity, and it is based on either of the previous two levels or both. We also note that spammers can add noise characters into the similar/fake Unicode strings; hence, we do string preprocessing to reduce or eliminate noise symbols.

3.1 Preprocessing

UCS contains many so-called symbol characters (e.g., '|' and '\' in the string "y0U|HaVE/A |FrEe \G|fT ++"). We consider these to be noise, which make it difficult for us to detect similar/fake Unicode strings. Hence, we have to do preprocessing to replace the symbol characters with empty strings or space characters (depending on the string similarity evaluation requirements). The symbol character list can be constructed by referencing the specification of Unicode [26] manually. Unicode string preprocessing is quite useful for phishing IRI/URI detection, especially for detecting spam emails, erotic content, and dirty words. Phishers will generally not add noise characters to their fake IRI/IDNs since they want to make them as visually similar as possible, so the preprocessing step is primarily aimed at spam attack detection.

However, preprocessing for general Unicode text strings is not simple work. First of all, we do not have a complete list of symbol characters. The UCS is a big, complicated and growing list. Also, we cannot conclude that all symbols are noise; for instance, "|" can be used by malicious people to replace "I" in the word "GIfT" (changing it to "G|fT"). Therefore, potential future work can concentrate on Unicode preprocessing alone.

3.2 Character Level Similarity

The basic trick of a Unicode attack is to replace some characters with similar ones. We address character similarity in two dimensions: visual similarity and semantic similarity.

A visual similarity list can be constructed automatically by comparing the glyphs in UCS. If necessary, we can optimize it manually. However, we are expecting to have an algorithm to construct the list completely automatically without additional manual work, since UCS is a huge database and the manual work could be overwhelming.

A semantic similarity list can only be constructed manually by referencing the specification of the Unicode repertoire because we cannot find an algorithm or a tool with the necessary knowledge to do it. We still do not have a complete semantic similarity list, and we list it as future work.

The overall character-character similarity matrix can be constructed by combining the visual and semantic similarity lists. We consider multiplication as a good combination method. That is, if the visual similarity of "ạ" (1EA1) and "a" (0061) is 0.9 and the semantic similarity of "a" (0061) and "A" (0081) is 1, we can calculate the overall similarity between "ạ" (1EA1) and "A" (0041) as 0.9×1=0.9. We also use this method in Section 4.1 to calculate the character level similarity. Other combination methods can also be attempted based on more investigation.

The character-character similarity matrix is a good resource for assessing the similarity of Unicode stings and recovering an original string from its noisy versions, or other similar tasks. For instance, we can do noise reduction to the example string in Section 3.1 and retrieve the intended message: "you have a free gift". We can then use the denoised string to perform word level similarity assessments with any candidate strings as addressed in Section 3.3.

3.3 Word Level Similarity

Unicode attacks can be carried out by replacing words with other semantically similar ones (e.g. synonyms). The following four types of semantic substitutions are most likely to occur in the near future:

3.3.1 Phonetic Substitution:

A malicious person may change some part of the string but still

keep the original pronunciation, e.g., "BankForYou" can be changed to "Bank4U", "中国银行" to "ZhongGuoYinHang", "日本銀行" to "にほんぎんこう" or "NiHonGinKou".

3.3.2 Acronym Substitution:
A malicious person may use the acronyms of the keywords of the original Unicode strings to carry out attacks, e.g., "BankOfChina" to "BOC", "中国银行" (Bank of China) to "中银", and "とうきょうだいがく" (the University of Tokyo) to "とうだい", etc.

3.3.3 Tongue Shifting Substitution:
A malicious person may translate some words in the original Unicode string to another language to carry out attacks. E.g., "BankOfChina" to "中国银行" or "中国バンク"

3.3.4 Synonym Substitution:
The words in a Unicode string could be replaced with their synonyms, e.g., "this is spam" to "this is junk mail", or "Hi, buddy" to "Hello, friend", etc.

These four types of word level obfuscations could be used together in many ways to make a single string even more complicated and difficult to detect, while still allowing humans to understand it.

The solution to detect word level obfuscations is to establish a word-word semantic similarity matrix to assess the similarity of strings. However, the matrix could be very complicated and large. We have constructed one based on word-word semantic similarity assessment algorithms. However we are still on the way toward constructing a complete one. It turns out that we need to use excellent word-word similarity algorithms and a lot of human intervention. It is also a growing matrix since new words are invented continuously. The matrix data structure should be well constructed because it quite large.

3.4 String Similarity Algorithms

We propose the methodology of using character-character similarity, as addressed in Section 3.2, and word-word similarity, as addressed in Section 3.3, to calculate the similarity of Unicode strings (which can be domain names, user names (accounts), sentences, passages, or even documents).

There are many standard string similarity evaluation algorithms from information retrieval (IR), natural language processing (NLP), and even bioinformatics which can be applied, such as edit distance [19], KMP[18], Needleman-Wunch distance [24], and n-gram etc. Many of them are based on character level similarity. Hence, we can apply them directly to evaluating the character level similarity. Since we have the word-word similarity, we could simply consider each word as a character, and then use character level similarity algorithms to calculate string similarity. We need to consider time complexity when we choose specific algorithms to calculate the string similarity. We provide an application based on implementing VSED and VSKMP at the character level as an example in Section 4 for demonstration.

4. A DEMO IMPLEMENTATION STUDY ON THE METHODOLOGY

The methodology in Section 3 turns out to be abstractive. Here we carry out a case study by applying part of the methodology. We organize this section in three subsections. In Section 4.1, we introduce our method of generating a Unicode similarity list (UC-SimList). The word-word similarity matrix is an ongoing study and we would rather omit this part since it does not affect the discussion here. In Section 4.2, we present a possible string similarity algorithm; this is implemented as the algorithm for use in our experiments. In Section 4.3, we address the problem of associating relatively long similar/fake Unicode strings with the genuine ones. We use the strings to imitate spam attacks. In Section **Error! Reference source not found.**, we address the problem of detecting phishing IRI/IDNs in a set of protected ones. It is a special and critical issue, so it should be considered specifically. In Section 4.3 and **Error! Reference source not found.**, we discuss aspects of experimental data generation, classification effects and time performance evaluation.

4.1 Unicode Character Similarity List (UC-SimList) Generation

Since online calculation of character similarity is expensive, a lookup table (character-character similarity matrix) can be pre-constructed offline, which consists of a list of similar characters for each individual character in Unicode. We refer to the lookup table as UC-SimList in this paper. The construction of UC-SimList is based on Arial MS Unicode [21] [22] simply because it covers more character glyphs than any other existing font. The visual similarity is calculated by calculating the similarity of each pair of characters. We denote the visual similarity of character c_1 and c_2 with $vs(c_1,c_2)$, which can be calculated using Eq. 1.

$$vs(c_1,c_2)=\frac{|OverlapPix(c_1,c_2)|}{p|Pix(c_1)|+(1-p)|Pix(c_2)|}, \quad (1)$$

where $|OverlapPix(c_1,c_2)|$ is the number of overlapping pixels of the bitmaps of c_1 and c_2, $|Pix(c)|$ is the number of pixels of character c, and $p\in[0,1]$ is the factor for tuning the similarity computation validity. Experiments show that p performs the best when it is defined as Eq. 2.

$$p=\begin{cases}1 & if\ |Pix(c_1)|\geq|Pix(c_2)|\\ 0 & otherwise\end{cases} \quad (2)$$

Semantic similarity of two characters is the measurement of character similarity in terms of meaning. It is common that one character has more corresponding representations in the same or different languages. In our approach, we define the semantic similarity of two characters as a binary value, i.e. either 1 or 0. E.g., ss("a", "A")=1, and ss("a", "b")=0, where $ss(c_1,c_2)$ is the semantic similarity of character c_1 and c_2. UC-SimList is generated by first considering the semantically similar characters for each character in UCS, finding all visually similar characters of each semantically similar character, and finally ranking all these characters with their visual similarities for each character. That is, UC-SimList(c)=UC-SimList$_v$(UC-SimList$_s$(c)), where c is a given character, UC-SimList$_s$(c) denotes the semantically similar character set of c, UC-SimList$_v$(•) denotes the visually similar character set of character set • , and UC-SimList(c) denotes the similar character set of character c. We denote UC-SimListT as the Unicode Character Similarity List that only contains the characters with similarities larger than T (the similarity threshold), e.g., UC-SimList0.8 is a subset of UC-SimList that only contains the characters with similarities larger than 0.8. We also define the notation UC-SimList$_v$T in a similar way. We provide the UC-SimList and UC-SimList$_v$ online for free on our Anti-Phishing Group website. People can download them from www.mit.edu/~ayf/IRI.

4.2 Unicode String Similarity Algorithm for Experiments

We use edit distance [19] to calculate the dissimilarity of the pair of Unicode strings under evaluation. Edit distance represents the minimum editing operations needed to transform one string into another, where the only operations are insertion, deletion, and substitution. We define the cost (cost function) of insertion and deletion to be 1 and the cost (cost function) of substitution to be 1 minus the similarity in UC-SimList0.8. We use a standard dynamic programming algorithm to calculate edit distance for better efficiency and performance. Its time complexity is $\Theta(mn)$, where m and n are the respective lengths of the two Unicode strings. Experiments in Section 4.3 and **Error! Reference source not found.** show that this is sufficient for online similar/fake Unicode string detection. The edit distances are normalized by dividing by $Max(m,n)$, such that we can define a threshold to classify whether a given suspected Unicode string is too similar to a string in a set of protected Unicode strings.

4.3 Experiments with Normal Text Strings

We begin our experiments with three Unicode strings extracted from three webpages, namely the English, Chinese and Japanese versions of CitiBank, as shown in

Figure **2**. We denote these strings as US_E, US_C, and US_J, respectively. We then generate similar/fake strings based on each of these strings by substituting some characters with visually or semantically similar ones from UC-SimList$_v$ or UC-SimList. For each original string, we generate 5 sets of similar/fake ones using each of UC-SimListT and UC-SimList$_v$T, where T∈{0.8, 0.85, 0.9, 0.95, and 1}. Each of the original Unicode strings corresponds to 10 sets of similar/fake ones and each set contains at most 100 similar/fake Unicode strings. We denote the 10 sets of similar/fake Unicode strings of US_E as SUS_E(X), the 10 sets of US_C as SUS_C(X), and the 10 sets of US_J as SUS_J(X), where X∈{0.8, 0.85, 0.9, 0.95, 1, V0.8, V0.85, V0.9, V0.95, and V1}. SUS_E(0.8) is the similar/fake Unicode string set generated using UC-SimList0.8, and SUS_E(V0.8) is the similar/fake Unicode string set generated using UC-SimList$_v$0.8, and so on. We use the 83,198 Unicode strings in TREC-5 Confusion Track (part original-01) [17] as noise data and denote this set as *RUS*. We combine SUS_E(X) with *RUS* to form the set of suspected Unicode strings for US_E and calculate the precision and recall value of our similar/fake Unicode string detection algorithm for US_E with varying thresholds (from 0 to 1, with a step of 0.01). A partial result is shown in Figure 3 and the complete results are available in [2].

We also perform the same experiments on US_C and US_J, as shown in Figure 4 and Figure 5 respectively, and the complete results are also available in [2]. We calculate the precision value using Eq. 3 and the recall value using Eq. 4 respectively, where *TN* denotes the total detection number, *CN* denotes the correct detection number and *TF* denotes the total ground truth number of fake ones (phishing) in the corresponding test set.

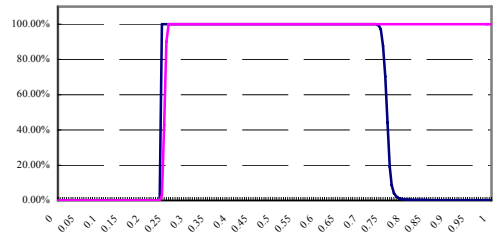
$$precision = \frac{CN}{TN} \times 100\% \quad (3)$$

$$recall = \frac{CN}{TF} \times 100\% \quad (4)$$

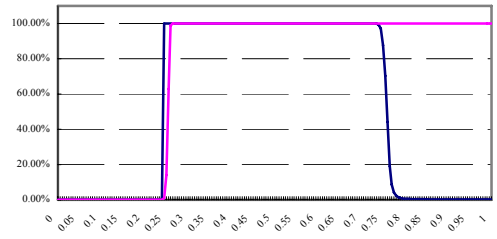
It is obvious that in a certain threshold range, we can achieve very good precision and recall values (both are approximately 100%) at the same time, i.e., 0.25~0.75 for US_E, 0.05~0.95 for US_C, and 0.05~0.97 for US_J. The wider the range is, the easier the classification of similar/fake Unicode strings is. Intuitively, similar/fake Unicode strings of Chinese and Japanese are easier to detect than those of English, and the reason is that English has far more similar characters in UCS than Chinese and Japanese do. We can observe two phenomena from the precision and recall figures. First, the threshold range is wider when the similar/fake Unicode strings are generated using UC-SimListT or UC-SimList$_v$T with higher T (similarity threshold) values. Second, the threshold range is wider when similar/fake Unicode strings are generated using UC-SimList$_v$T rather than UC-SimListT with the same T. However the second phenomenon is not very obvious in our experiments when we used UC-SimList0.8 as the cost function in Section 4.2. These phenomena also adapt to the experiment in Section **Error! Reference source not found.**. It takes 0.15 seconds to calculate the similarity of a Unicode string to US_E, 0.035 second to US_C, and 0.045 second to US_J on average (using a PC with P4 2.4G CPU and 512M RAM). The proportion of computation time, 0.15:0.035:0.045=1:0.23:0.3 is roughly equivalent to the character number proportion of the three original Unicode strings, 313:79:102=1:0.25:0.33, which verifies that the computational complexity grows linearly with the target string length as addressed in Section 4.2. All of the experimental data sets are available at [2].

From	Original Unicode strings (attacked targets)
www.citibank.com	Every Internet user should know about spoof (a.k.a. phishing or hoax) e-mails that appear to be from a well-known company but can put you at risk. Although they can be difficult to spot, they generally ask you to click a link back to a spoof web site and provide, update or confirm sensitive personal information. (total: 313 characters)
www.citibank.com.cn	最近，电子邮件用户成为全球网络黑客的攻击目标。花旗银行相信让所有网上银行用户了解邮件欺诈是至关重要。因此，我们为您提供了一系列建议以防止您的金融信息受到攻击。(total: 79 characters)
www.citibank.co.jp	最近、シティバンクを装って送られる偽の電子メールが増えております。一般的にこのような電子メールにあるリンクをクリックすると、暗証番号や口座番号など個人の機密情報の入力を促す偽のウェブページがあらわれます。(total: 102 characters)

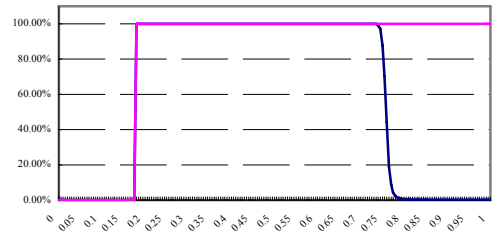
Figure 2. Original Unicode strings or English, Chinese and Japanese from the webpages of CitiBank



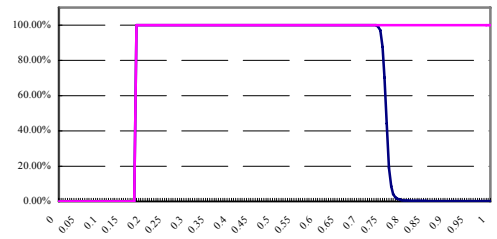
X=0.8



X=V0.8

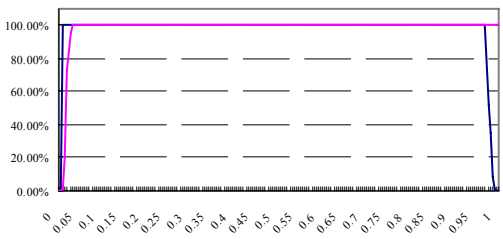


X=1

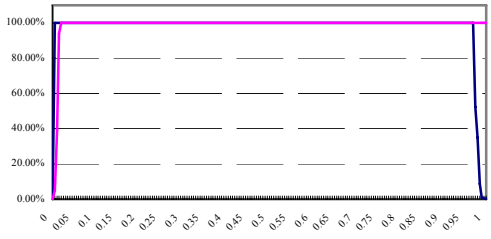


X=V1

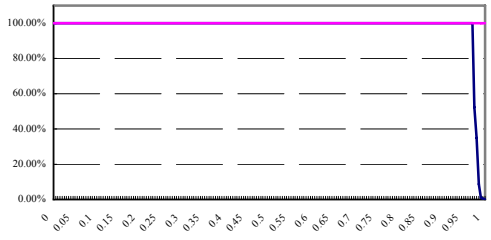
Figure 3. Precision and recall evaluation of detecting similar/fake Unicode strings to US_E (the purple curve is recall, the blue one is precision, the x-axis denotes the varying thresholds and the y-axis denotes the precision/recall percentage value)



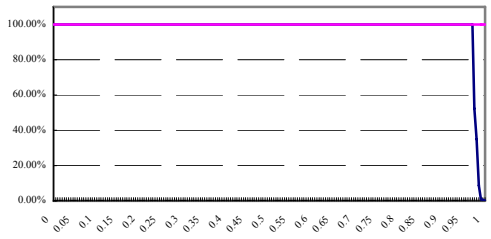
X=0.8



X=V0.8

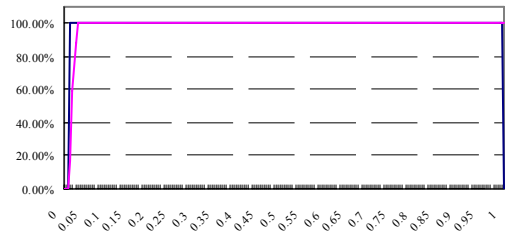


X=1

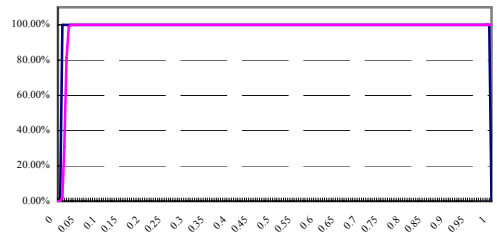


X=V1

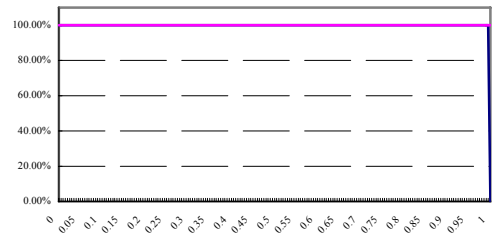
Figure 4. Precision and recall evaluation for detecting similar/fake Unicode strings to US_C (the legend is the same as in Figure 3)



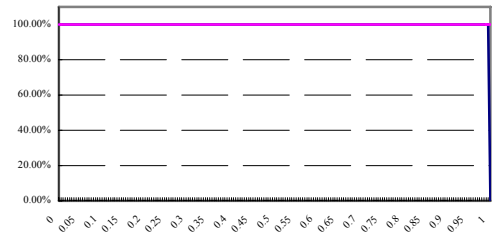
X=0.8



X=V0.8



X=1



X=V1

Figure 5. Precision and recall evaluation for detecting similar/fake Unicode strings to US_J (the legend is the same as in Figure 3)

4.4 Experiments with IRI/IDNs

Although IRI/IDN could be the complement or the replacement of URI in the near future, IRI/IDN is not used widely at present, and the number of real phishing URIs or IRI/IDNs is small. Hence, we also need to generate phishing IRI/IDNs for our experiments. Suppose we have 10 IRI/IDNs under protection as listed in Figure 6. We'll denote these as US_{IRI}. Next, we generate 100 similar IRI/IDNs for each of them by replacing similar characters in each original IRI/IDNs using UC-SimListT and UC-SimList$_v$T, where T∈{0.8, 0.85, 0.9, 0.95, and 1}. Note that we get 1,000 IRI/IDNs for each element of US_{IRI}, since there are 5 different T values, and we use each value in both UC-SimListT and UC-SimList$_v$T. We treat all of these generated IRI/IDNs as phishing IRI/IDNs, and we denote the 10 sets as SUS_{IRI}(X), following the conventions defined in Section 4.3.

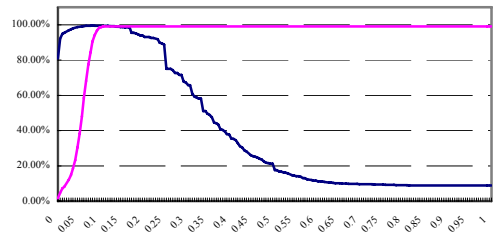
www.citibank.com	www.icbc.com
www.bank-of-china.com	www.wellsfargo.com
www.ebay.com	www.keybank.com
www.wamu.com	www.花旗银行.公司
www.usbank.com	www.シテイバンク.会社

Figure 6. The 10 IRI/IDNs under protection

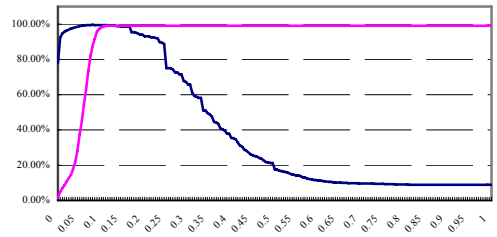
Now we mix the phishing IRI/IDNs up with 10,269 real web addresses that are randomly collected from the Internet. We then use the algorithm in Section 4.2 to perform phishing IRI/IDN detection. We use precision and recall values to evaluate our approach with varying edit distance thresholds. We also do experiments with UC-SimLists with various filtering thresholds (from 0 to 1, with a step of 0.01). We mix up SUS_{IRI}(X) with 11,269 IRI/IDNs in random order to form the suspected Unicode strings and calculate the precision and recall value of detecting similar/fake Unicode strings to the 10 protected IRI/IDNs with varying thresholds. The partial experimental results are shown in Figure 7**Error! Reference source not found.**, and the complete results are in [2]. It is obvious that there is a clear threshold range, 0.08~0.17, where both high precision and recall values (both approaching 100%) can be achieved. It takes about 10^{-3} second to calculate the similarity of one pair of IRI/IDNs using the same machine we used in Section **Error! Reference source not found.**. This performance is sufficient for online phishing IRI/IDN detection.

5. IRI/IDN SECUCHECKER

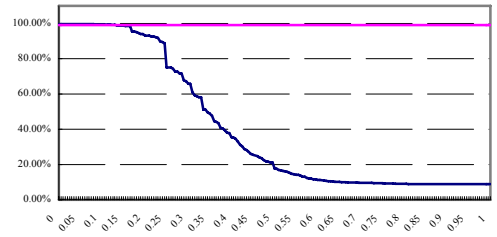
With the internationalization of information processing, the use of IRI/IDN is becoming a trend. However, we've shown that IRI/IDNs can be deceptive; in particular, it is possible to have two distinct IRI/IDNs which are hard (or impossible) to distinguish visually. This problem has already been reported for URIs in [12], which notes the possibility of mimicking English "microsoft.com" with Cyrillic "microsoft.com". However, there is no application or tool available for IRI/IDN detection. The domain name registration regulations are made by ICANN [15]. It can improve the domain name registration guidelines and ask its authorized registrar companies to follow them. ICANN first added the related section in "Additional Remark" in IRI/IDN Ver. 2.0, Nov. 8, 2005, and continued listing it in the "Additional Remark" of Ver. 2.1, Feb. 22, 2006. However no solution has been provided yet.



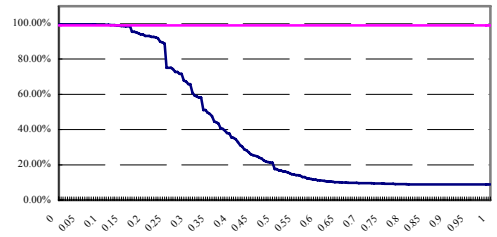
X=0.8



X=V0.8



X=1



X=V1

Figure 7. **Precision and recall evaluation of detecting phishing IRI/IDNs to US_{IRI} (the legend is the same as in Figure 3)**

We implement IRI/IDN SecuChecker, which provides a mechanism to help domain name registrars check the validity of new registered domain names. Figure 8 shows the interface of IRI/IDN SecuChecker. It includes a textbox to input a new registered IRI/IDN, a display showing the Unicode form of the inputted IRI/IDN, an option pane for selecting the Kernel Algorithm, a "Search" button, a "Clear" button, and a listbox to show the detected similar/fake IRI/IDN(s). There are two important lists in the database running behind the application: the UC-SimList and the registered IRI/IDN list. According the methodology in Section 3.4, various string matching algorithms (including string similarity algorithms and substring searching algorithms, etc.) could be applied. We have implemented visual and semantic based edit distance (VSED), where we use the UC-SimList to evaluate the cost of replacing one character into another. We choose the threshold to be 0.12, where we can achieve the best recall and precision values at the same time, as already shown in Figure 7**Error! Reference source not found.**. We also implement the visual and semantic based Knuth Morris Pratt (KMP) algorithm (VSKMP), where we use UC-SimList to assess the similarity between two given characters. We use a character similarity threshold of 0.08 to evaluate the two given characters—empirically, we find that this threshold can classify the characters well, as shown in Figure 9. Characters to the left of the thick vertical bars have similarity values to the given characters (GCs) of more than 0.8.

VSED shows better performance in phishing IRI/IDN detection. VSED can detect "www.bankofthevvest.com" (double "v" to mimic "w") from "www.bankofthewest.com", while VSKMP cannot, as shown in Figure 10. However VSKMP has better time complexity, namely $\Theta(m+n)$ compared to $\Theta(mn)$ for VSED (where m and n are the lengths of the two strings under assessment). In real experiments, VSKMP tends to be fast, and it also behaves well when the protected IRI/IDN is a substring of the new domain name under registration, e.g. VSKMP can detect the phishing IRI/IDN "www.citibank.com.info123.biz", while VSED cannot, as shown in Figure 11. Each pair of domain name evaluations takes about 0.0012 seconds with VSDE and about 0.0005 seconds with VSKMP (on average, on a P4 2.4G PC with 512MB RAM). So if the user thinks the system is fast enough, we recommend selecting "Both" in the kernel algorithm group before pressing the "Search" button. In this way, the phishing IRI/IDN(s) detected by any of VSED or VSKMP will be reported.

IRI/IDN SecuChecker is not just limited to checking the validity of domain names; it can also be used in any instance where visually and semantically unique Unicode identifiers are desired. For instance, it can be used in a user name (account) registration server, preventing users from spoofing the identities of other users.

In the new registration textbox, we used different colors to represent characters from different language character sets. It is another feature of IRI/IDN SecuChecker and we call it Web Link Illustrator. We can simply add this web address convention to web browsers' address bars. A demo of the Web Link Illustrator for Microsoft IE is available at [9] as shown in Figure 12. ICANN classifies characters that can be used in many different languages, which are listed in [15]. To implement a Web Link Illustrator, one can simply study and implement the character code ranges and program add-ins for IE, FirFox, etc. In this demo, the fake address for CitiBank contains an "a" from a different character set, which is highlighted in a different color (red) to remind users to be cautious (we choose to change the color because human eyes are sensitive to colors). We consider this a simple but effective potential feature for web browsers.

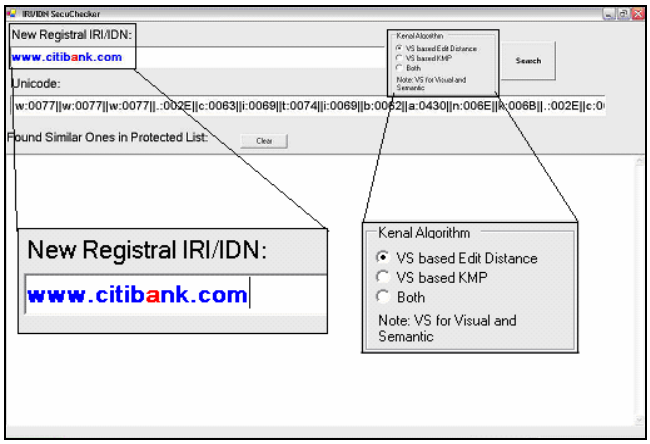
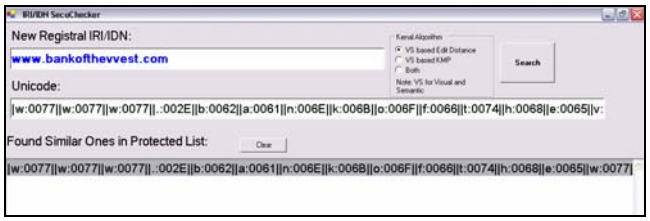


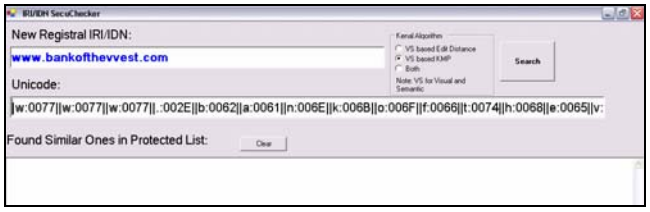
Figure 8. The interface of IRI/IDN SecuChecker

GC \ Rank	1	2	3	4	5	6	7
M	M	M	M	M	Ṃ	M̈	IX
004D	FF2D	039C	041C	216F	1E42	1E40	2168
N	N	N	Ṇ	Ṉ	N̲	N̂	Nj
004E	FF2E	039D	1E46	0145	1E48	1E4A	01CB
Z	Z	Z	Ẓ	Z̲	Ƶ	Σ	Σ
005A	0396	FF3A	1E92	1E94	01B5	01A9	03A3

Figure 9. The threshold selection for characters' visual similarity. Note: GC for given character. In each row, higher ranks indicate higher visual similarities to GC.

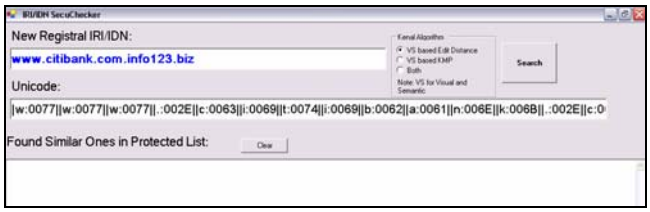


a) Phishing IRI/IDN detected using VSED

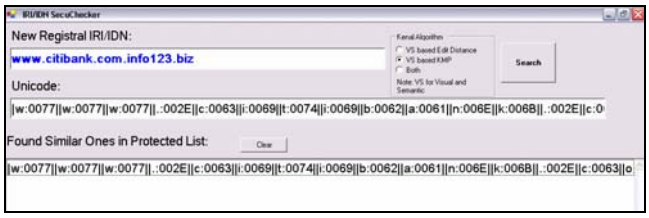


b) Phishing IRI/IDN not detected using VSKMP

Figure 10. VSED can detect the string"www.bankofthevvest.com " (double "v" to mimic "w"), while VSKMP cannot.

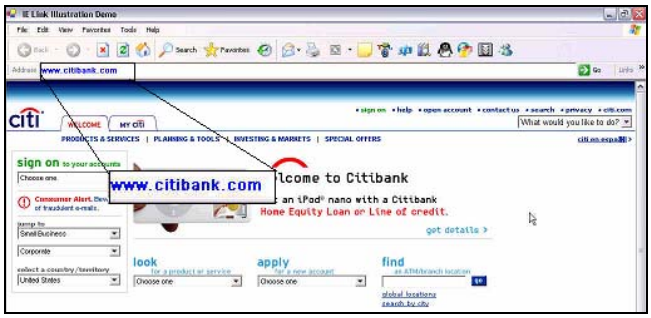


a) Phishing IRI/IDN not detected using VSED



b) Phishing IRI/IDN detected using VSKMP

Figure 11. VSKMP can detect the string "www.citibank.com. info123.biz", while VSED cannot.



a) Correct web link. Web Link Illustrator paints all characters blue, which is the color corresponding to Latin characters.

b) Web Link Illustrator paint the "a" red, since it is from another character set.

Figure 12. Demo for Web Link Illustrator.

6. DISCUSSION-PRACTICAL USE AND DEPLOYMENT PROPOSAL

We have demonstrated in Section 4 that the prototype application performs well for Unicode attack detection. All it needs is a list of domain names to protect and the attack detection algorithms. However, deploying the technology in real-world situations raises issues that we would like to address in this section.

6.1 Domain Name Server

When we need a domain name, we get it from domain name registrars. The registrars are relatively autonomous. However they are under the supervision of ICANN (Internet Corporation for Assigned Names and Numbers). ICANN makes regulations for the Internet domain name registration. It will be very helpful if ICANN can promote the pre-checking process before a malicious web page can be registered. People may worry that this limits their freedom to register any domain name, but it may protect more people than it hurts.. As a matter of fact, ICANN is realizing this problem. ICANN first added the related section in

"Additional Remark" in IRI/IDN Ver. 2.0, Nov. 8, 2005, and continued listing it in the "Additional Remark" of Ver. 2.1, Feb. 22, 2006. However we have not seen a solution yet. We hope that the method proposed in this paper can be used to address this issue.

Another solution is that domain name service registrars may provide the following service: once a domain name is registered, all similar domain names will automatically be registered to the same domain name owner. This service can be implemented by enumerating every way of substituting similar characters from UC-SimList for the characters in the domain name. The obvious drawback of this approach is that some domain names will have a prohibitively large number of similar names (since this number grows exponentially in the length of the name). However, some organizations may be willing to pay a premium for the resulting security if their name is small enough to secure in this manner.

6.2 Anti-Phishing Client Application

There are solutions to protect end user computers. The Unicode attack method can be embedded into anti-phishing client applications or web browser plugins and installed in end user computers. These client applications and plugins act as filters for the web links that users try to use to access the Internet. If any suspected web link is found, then they can provide alert information. One possible problem is that it may be hard for client applications or plugins to maintain one complete list of legitimate domain names. However they can be designed as only maintaining the most security sensitive domain names, e.g. only maintain the list of banks, credit card companies, and online transactions services.

6.3 Registrar applications

As we discussed, Web Identity could become a big target for malicious people. They can use Unicode attacks to create similar user names and accounts on the Internet. We would like to propose that username/account registrars in various online systems use Unicode attack detection systems to overcome possible attacks. E.g. when someone wants to register a new username/account in a BBS system, if the BBS system allows users to register with Unicode strings (they will do this if it has many international users that prefer to use Chinese, French etc.), then Unicode attacks could happen. We propose using Unicode detection systems to verify the legitimacy of new usernames during registration.

6.4 Content Filtering

It is possible that phishers can carry out attacks through systems such as Email Servers, BBSs, Wikis, and Search Engines. It has been reported that people have directed phishing web pages through Google Searches. Such attacks not only cause network security problems, but also affect the reputations of decent service providers. Our method can be used by such systems to detect phishing Unicode attacks.

7. CONCLUSION AND FUTURE WORK

We have identified a severe security and privacy threat to various systems which use Unicode as text media, namely the Unicode attack. The problem arises from the fact the computer screens are hiding the truth from their users about which characters they are displaying. To resolve Unicode attack problems, we propose a methodology, which can be used to implement Unicode attack detection systems. Following this methodology, we provide a real demo implementation study using vision and semantic based algorithms to perform similar/fake Unicode string detection. Experiments show that both the classification effect and time efficiency of the proposed method are satisfactory. The threshold for VSED could reach an optimum around 0.12. We also implemented VSMKP, where we showed that the threshold for character-character similarity of 0.80 is an empirical optimization of VSKMP. We also built up IRI/IDN SecuChecker using the two algorithms to perform IRI/IDN based phishing and fake web identity detection. The demo of Web Link Illustrator [9] demonstrated a possible improvement for web browsers.

The establishment of UC-SimList$_s$ is a complicated work, as discussed in Section 4.1. The current version of UC-SimList$_s$ only includes characters in English, Chinese, and Japanese. It is expected to include more languages in our future work. We also would like to keep working on the generation of the word-word similarity matrix and apply it to the IRI/IDN SecuChecker. The algorithms, VSED and VSMKP, in IRI/IDN SecuChecker are the kernel parts to construct potential future similar/fake Unicode string detection systems. We expect that they could be implemented in email servers, BBSes, chatting rooms, and gateway filters to perform Unicode attack detection tasks.

REFERENCES

[1]. ANSI, *American Standard Code for Information Interchange*, www.ansi.org.

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

[3]. Anti-Phishing Working Group, http://www.antiphishing.org.

[4]. Berners-Lee T., Fielding R., Masinter L., RFC 3986: Uniform Resource Identifier (URI): Generic Syntax, The Internet Society (2005), Jan. 2005.

[5]. Chowdhury A., Frieder O., Grossman D., and McCabe. M. *Collection statistics for fast duplicate document detection*, ACM Transactions on Information Systems, Volume 20(2), pages 171-191, 2002.

[6]. Dhamija R., Tygar J. D., and Hearst M. *Why Phishing Works*, to appear in Proceedings of CHI-2006: Conference on Human Factors in Computing Systems, April 2006.

[7]. Dhamija R., Tygar J. D., *The Battle against Phishing: Dynamic Security Skins*, Symposium on Usable Privacy and Security 2005, pages 77-99, 2005.

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

[9]. Fu A. Y. Web Link Illustrator Demo for Microsoft IE, www.mit.edu/~ayf

[10]. Fu A. Y., Deng X. , Liu W., *A Potential IRI based Phishing Strategy*, in the Proceedings of 6th International Conference on Web Information Systems Engineering, LNCS Volume 3806, pages 618 - 619 , 2005

[11]. Fu A. Y., Liu W., Deng X., *EMD based Visual Similarity for Detection of Phishing Webpages*, in Proceedings of International Workshop on Web Document Analysis 2005, 2005.

[12]. Gabrilovich E. and Gontmakher A. *The homograph attack*. Communications of the ACM, 45(2):128, 2002.

[13]. Garfinkel, S., Miller, R., *Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express*. SOUPS, 2005.

[14]. Hoad T.C., Zobel J. *Methods for identifying versioned and plagiarized documents*, Journal of the American Society for Information Science and Technology, Volume 54(3), pages

203-215, 2003.
[15]. ICANN. http://www.icann.org/
[16]. Jakobsson M., *Modeling and Preventing Phishing Attacks*, Phishing Panel of Financial Cryptography, 2005.
[17]. Kantor P., Voorhees E., *The TREC-5 Confusion Track: Comparing Retrieval Methods for Scanned Text*, Information Retrieval, Volume 2(2/3), pages 165-176, 2000.
[18]. Knuth D., Morris J. H., and Pratt V., *Fast pattern matching in strings*. SIAM Journal on Computing, 6(2):323¨C350. 1977. Citations. Original publication.
[19]. Levenshtein V.I., *Binary codes capable of correcting deletions, insertions, and reversals*, Cybernetics and Control Theory, Volume 10, pages707-710, 1966.
[20]. Liu W., Deng X, Huang G., Fu A. Y., *An Anti-Phishing Strategy based on Visual Similarity Assessment*, IEEE Internet Computing, Vol. 10, No. 2, pp. 58-65, 2006
[21]. Microsoft, *Arial Unicode MS, Version 1.01*, 2000.
[22]. Microsoft, *Description of the Arial Unicode MS font in Word 2002*, http://support.microsoft.com/kb/q287247.
[23]. Nanno T., Saito S., and Okumura M., *Structuring Webpages based on repetition of elements*, in Proceedings of the 7th International Conference on Document Analysis and Recognition, 2003.
[24]. Needleman S. and Wunsch C., *A general method applicable to the search for similarities in the amino acid sequence of two proteins*, Journal of Molecular Biology, 48(3):443_453, 1970.
[25]. Netscape, http://wp.netscape.com/eng/ssl3.
[26]. The Unicode Consortium, http://www.unicode.org.
[27]. Wood L., Document Object Model (DOM) Level 1 Specification, http://www.w3.org/TR/ REC-DOM-Level-1.
[28]. Wu, M., Miller, R., Garfinkel, S., *Do Security Toolbars Actually Prevent Phishing Attacks?*, to appear in CHI 2006