Phishing Defense against IDN Address Spoofing Attacks

Viktor Krammer [1,2]
[1] E-Commerce Competence Center
[2] Vienna University of Technology
Vienna, Austria
vkrammer@acm.org

ABSTRACT

Address spoofing is a common trick used in phishing scams to confuse unsuspecting users about a Web site's real origin. With the introduction of Unicode characters into domain names, also known as Internationalized Domain Names (IDN), the risk has significantly increased even for the most cautious users. The author explores the various types of address spoofing attacks focusing on IDN, and presents a novel client-side Web browser plug-in Quero which implements several techniques—including highlighting—to protect the user against visually undistinguishable address manipulations.

Categories and Subject Descriptors

D.2 [**Software**]: Software Engineering; H.5.2 [**Information Systems**]: Information Interfaces and Presentation—*User Interfaces*

General Terms

Security, Algorithms, Design

Keywords

Internet Security, Phishing, Internationalized Domain Names, Unicode, Usability, Web Browsers, Internet Explorer

1. INTRODUCTION

Phishing has become a widespread problem of today's Internet by tricking unsuspecting users into revealing sensitive information, such as login credentials, on fake Web sites. The URL displayed in the address bar of the Web browser is for many users a trustworthy reference point for their current location in the World Wide Web. Furthermore, the domain name has the potential to indicate the origin of the content, a circumstance that is commonly exploited by "phishers" and referred to as *address spoofing*.

Other techniques typically used in phishing attacks include content spoofing and spamming, the former referring to their resemblance to original Web sites and the latter to fake mails sent to potential victims. Phishing has been already discussed one decade ago under the more general term Web spoofing in literature [15].

A real-world phishing incident[1] was recently reported in Austria targeting the Bank Austria Web site located at `www.ba-ca.com` by registering the fake domain `ba-cq.com`.

While this classic address spoofing attack can be discovered by carefully looking at the current URL, the recently introduced concept of Internationalized Domain Names (IDN) can be taken advantage of to spoof addresses which are no longer visually distinguishable from their legitimate counterparts. Simplified IDN extends the repertoire of registrable characters to (a subset of) Unicode [1] (version 3.2). However, due to compatibility considerations, this is not done by a modification to DNS, but left to a client-side resolver referred to as IDNA (Internationalizing Domain Names in Applications), which transparently maps Unicode domain labels to ASCII-only, DNS-compatible labels and vice versa.

Although the authors of IDNA had security in mind, *Gabrilovich* and *Gontmakher* [16] discovered a possible attack scenario, called the homograph attack, in which the characters might be replaced by visually identical ones (homographs)[2] found in Unicode. Attempts to show only the ASCII representation of IDN domains in Web browsers or restrict the use of Unicode characters depending on the top-level domain have so far enjoyed only moderate success. "Users can not reliably parse domain names" [12], a reality we must face given the steadily growing number of inexperienced Web users.

In this article I focus on IDN address spoofing attacks and propose techniques to protect the user and help her detect suspicious characters in domain names by following the recently published Unicode Security Considerations [11]. I have implemented an IDN enabling plug-in for Internet Explorer, called Quero Toolbar[3] [26], and experimented with the proposed anti-spoofing techniques. In section 2 I discuss the different types of address spoofing in more detail, followed by a list of requirements that an efficient solution should take into account in section 3. Section 4 presents my solution, implemented in Quero, which is briefly described in section 5. Related work is discussed in section 7 and concluding remarks are given in section 8.

[1] `http://futurezone.orf.at/it/stories/86570/`
[2] Unicode defines for example the Latin small letter 'a' (U+0061) and the Cyrillic small letter 'a' (U+0430) as different code points.
[3] Quero Toolbar is freeware and has been downloaded over 10,000 times between 01/2005–01/2006

2. TYPES OF ADDRESS SPOOFING ATTACKS

Basically, what all address spoofing attacks have in common is that they lead the user to think she is on a legitimate Web site while actually viewing a fake page. This is achieved by crafting a URL that resembles so closely or is even indistinguishable from the real address so that the user does not notice the trick. Address spoofing attacks can be classified into attacks that rely on user confusion and mistakes, vulnerabilities in either the Web browser or the legitimate Web site and visual ambiguity between the encoding of the address and its visual representation. The focus in this article is mainly on the last type of spoofing, involving security issues that were introduced or aggravated by the adoption of Internationalized Domain Names. Attacks based on software vulnerabilities or IDN encoding ambiguity are considered to be the most dangerous, because they can also fool the most Internet-savvy user. Below I give a summary of the various types of address spoofing attacks.

2.1 User Confusion-based Attacks

2.1.1 *Confusion by Name Similarity*

The attacker registers a slightly different domain name than the real one. For instance, `southtrustonlines.com` instead of `southtrust.com`. This works because registrars do not manually review each registration and do not perform similarity checks. Additionally, the attacker might use a fake or stolen identity for the registration.

2.1.2 *Confusion by Address Complexity*

The simplest and most common form found in phishing scams is a URL with an IP address. According to the phishing archive of the Anti-Phishing Working Group[4], a real-world attack used for example the following URL:

`http://61.129.33.105/secured_site/www.skyfi.com/index.html?MfcISAPICommand=SignInFPP&UsingSSL=1`

Often the attacker tries to confuse the user by the sheer length of the address by adding a cryptic query string and mentioning the original name of the victim's Web site somewhere in it to make the address look more familiar. A variant of this attack type is subdomain spoofing, which takes advantage of the fact that the host name is displayed in least significant label first order. By crafting a sufficiently long chain of subdomains the important part of the address, usually the second/third level domain label, may no longer fit into the visible part of the address field, fooling gullible users into thinking that the domain name is the first part of the URL.

Example: `http://www3.ebay.com--login.dll-from-email-satitle-update-user-information.secure-connection.attackersdomain.com/...`

Another variant relies on the user:password URL scheme[5] [3], which allows one to supply a user name and password (in plaintext) in front of the host name. From a security point of view, I regard this as bad design since the cleartext password can leak in various ways (gathered by passers-by, possibly stored in the browser's URL history or even transmitted to other Web sites through the referer HTTP request field).

[4]`http://www.antiphishing.org/`

[5]Microsoft has decided to disable such URLs in Internet Explorer by default.

Example: `http://www.ebay.com:8080@login.dll-from-email-satitle-update-user-information.secure-connection.attackersdomain.com/...`

2.1.3 *Confusion by Random Addresses*

Although this is not really an address spoofing attack, it is often encountered in phishing scams. The attacker either does not try to disguise the address at all or relies on a serious-sounding address that has nothing in common with the legitimate one.

For instance, `http://secure-user-survey.com/exec/obidos/subst/home/` may mislead users into disclosing personal information. This attack works best if the user does not look at the address at all.

2.2 Vulnerability-based Attacks

This type of address spoofing attack relies on security flaws in either the Web browser (client-side) or the Web application of the victim's Web site (server-side), and is therefore very dangerous and hard to detect.

2.2.1 *Client-side Vulnerabilities*

An historic example is for instance the vulnerability documented in the Secunia Advisory SA12304 [33], where Internet Explorer fails to update the address bar after a sequence of actions is performed on a named window. Although marked as unpatched by Secunia, IE 6.0 SV1 is not affected by this vulnerability. Quero's pop-up blocker blocks this attack as well.

Another client-side vulnerability in IE known as chrome UI spoofing allows an attacker to spoof elements of the browser's UI by creating chromeless pop-up windows [5]. Actual phishing attacks use this to overlay the address bar with the original address of the Web site being spoofed. IE 6.0 SV1 limits the ability of chrome UI spoofing by restricting the position where such pop-up windows may appear [2].

2.2.2 *Server-side Vulnerabilities*

In these attacks, mainly based on cross-site scripting vulnerabilities, malicious content is injected into the victim's Web site. Strictly speaking, this kind of attack is not an address spoofing attack but the symptoms are the same, leading the user to believe she is viewing content from the legitimate site when in fact she is not.

2.3 IDN-based Attacks

While the former address spoofing attacks are on a word or more abstract level, we define IDN-based attacks as being on a lower, character-based level, particularly exploiting the encoding subtleties of Unicode. There have been no real-world IDN-attacks reported so far. This may be because at the moment only a minority of Internet users have access to IDN domains[6]. However, without effective anti-spoofing techniques, such attacks will pose a very serious security threat when IDN becomes widely available and popular, as the following attack types based on the Unicode Security Considerations [11] demonstrate.

2.3.1 *Mixed-script Spoofing*

This attack relies on the circumstance that some characters are defined more than once while belonging to different

[6]Microsoft Internet Explorer 6 and below do not support IDN natively and Firefox 1.0.x shows only the ACE (ASCII Compatible Encoding identifiable by the `xn--` prefix) form of the address.

script groups, despite the fact that one of Unicode's primary goals is unification. Reasons for this decision were compatibility with legacy encodings, varying behavior with respect to case folding, different character categories (letter vs. number) and different numerical values for homographs depending on the intended writing system. For instance, the Latin small letter 'o' U+006F can be confused with the Cyrillic small letter 'o' U+043E, the Greek small letter omicron 'o' U+03BF and the Myanmar letter wa 'o' U+101D. With this knowledge an attacker can already easily compose 15 different combinations of `google.com` by replacing the `o`s with the above-mentioned homographs. The risk is especially multiplied with East Asian scripts having thousands of characters with only minor distinctions.

2.3.2 *Whole-script Spoofing*

Due to historical reasons or mere coincidence, different scripts share identical or similar-looking characters and symbols. It may be possible to compose an entire domain name in Cyrillic that spoofs a Latin (ASCII-only) one or vice versa.

Example: `caxap.ru` ASCII domain that if interpreted in Cyrillic means "sugar" in Russian.

2.3.3 *Single-script Spoofing*

In single-script spoofs the similarity between characters within one script or characters common across scripts (such as numbers and symbols) is exploited. ASCII-only domain names are also well-known to be vulnerable to this kind of address spoofing attack by replacing 'o' with Zero '0' (working example `http://www.micros0ft.com/`), 't' with 'l', 'm' with 'rn' and so on. Unicode characters that are shared among different scripts such as numbers, symbols and punctuation marks, are also problematic.

Example: `http://www.ba-ca.com/` encoded with the Unicode hyphen symbol U+2010 instead of U+002D.

2.3.4 *Syntax Spoofing*

In IDN syntax spoofing, characters with a special syntactic meaning in URLs such as the slash '/' (U+002F) are substituted with a Unicode variant (U+2044 or U+2215)[7] to trick the user into thinking that the domain name ends at the pseudo-slash character.

2.3.5 *Numeric Spoofing*

This type of spoofing attack is a special case of mixed-script spoofing where Arabic numbers are replaced by similar-looking characters such as the Bengali digit zero U+09E6 or the Bengali digit four U+09EA, which looks like the Arabic digit eight U+0038.

2.3.6 *Invisible Character Injection*

Unicode has a large repertoire of invisible characters such as control, formatting, tagging and spacing characters that could have been deliberately inserted into a domain name. Fortunately, those characters are either prohibited or removed by the IDN Nameprep string preparation function [19, 20].

2.3.7 *Bidirectional Text Spoofing*

Since Unicode supports right-to-left writing systems (such as Hebrew or Arabic), it is easy to create many strings whose characters are in different orders (resulting in different identifiers) but are displayed identically. Unless manually overridden Unicode stores characters in logical order.

[7]Both characters are not prohibited by IDNA.

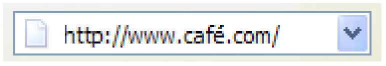

http://www.café.com/

Figure 1: Inadequate Rendering

Example: <U+202E (right-to-left override), g, o, o, g, U+202C (pop directional formatting), l, e, ., c, o, m>

This spoofing attack, however, is eliminated in RFC-compliant IDNA implementations because the specification [19, 20] requires that:

- Each label of a host name must not use both right-to-left and left-to-right characters.
- A label using right-to-left characters must begin and end with right-to-left characters.
- Manual bidi modifiers are prohibited.

Note: For each Unicode character the Unicode Character Database [34] defines a bidirectional character type (e.g. left-to-right) which is used by the Unicode Bidirectional Algorithm [8] to determine the ordering of the displayed characters.

2.3.8 *Combining Mark Order Spoofing*

Another encoding-specific threat is combining mark order spoofing. Unicode has defined a rather large number of combining diacritical marks but has assigned some common combinations their own character at the same time. For instance the German umlaut 'ö' can be expressed either with the combining diaeresis character as the sequence <U+006F, U+0308> or by the composed form Latin small letter o with diaeresis <U+00F6> as a single code point. While Unicode Normalization NFKC does exactly this transformation for <U+006F, U+0308>, the order of multiple combining marks within the same canonical combining class is undefined.

Example: under NFKC
<U+006F, U+0302, U+0300> $\rightarrow$ <U+1ED3>
<U+006F, U+0300, U+0302> $\rightarrow$ <U+00F2, U+0302>
Note:
U+0300 combining grave accent
U+0302 combining circumflex accent
U+1ED3 Latin small letter o with circumflex and grave
U+00F2 Latin small letter o with grave
Both U+0300 and U+0302 are in canonical combining class 230.

2.3.9 *Inadequate Rendering Support*

An important role in visual spoofing is played by the underlying text-rendering engine and the font used for displaying the address. The smaller the font size, the harder it is for the user to notice differences in the glyphs. For instance, the default font size in the IE address bar is set to only 8 points, which is rather small, especially for ideographic characters. Another problem is missing glyphs since they are all mapped to a default symbol—an empty rectangle in Windows. Also important is how combining marks are handled and whether bidirectional text is correctly rendered.

Example: Repeating combining marks
<c, a, f, e, U+0301, U+0301> looks like café in the address bar (cf. figure 1).

3. REQUIREMENTS

In this section I would like to outline the requirements which I believe any good and usable defense against the above-mentioned attack types should fulfill. Although some of the attack types can be addressed by domain name registration policies, an additional security layer in the Web browser is inevitable. The drawback of registration-based policies is that they put an additional burden on domain registrars to implement them. In all reality it is unlikely that the existing and rather sophisticated guidelines for IDN implementation and administration [22, 21, 25][8] will be fully adopted by all registrars in the near future. Moreover, attack types such as subdomain name spoofing and confusion by address complexity are beyond the scope of such policies. Thus I focus mainly on requirements and solutions for client-side defense. The following list is derived from the Unicode Security Considerations [11] and an online article about phishing by the former Mozilla intern *Markham* [27].

3.1 RFC Compliance

Any solution must adhere to the IDNA-related RFC specifications [19, 14, 20, 7]. These specifications basically require the deletion of certain characters and that all domain labels be folded to lower case, normalized under NFKC, do not contain prohibited characters and fulfill certain requirements for bidirectional text, i.e. text directions must not be mixed within one label. Case folding and normalization significantly decrease the available label space. Furthermore lower-case letters are thought to be more distinguishable than their upper-case counterparts. With Unicode normalization NFKC compatibility characters such as U+FB01 (ligature 'fi') are decomposed (U+0066,U+0069), followed by their canonical composition. Problematic characters like the soft hyphen U+00AD, the zero width space U+200B or other control, formatting and invisible characters are either removed or prohibited. However the homograph problem, as well as other spoofing attacks, remain unsolved by mere RFC compliance.

3.2 Easy User Interface

Crucial for the success of any anti-phishing solution is the design of its user interface, which is also decisive for its effectiveness and user acceptance. The security-related UI should be easy to use and simple to understand. There should be no unnecessary extra work required from the user (e.g. traversing trough complicated submenus or reading complex dialogs) to access security features. The UI should also be as consistent as possible, avoiding for instance separate UI widgets and different dialogs for each attack type where a unified one would be more feasible. Because security is often neglected by users and thought of as a burden, the UI should not waste precious space in the browser. For example, Internet Explorer displays an optional information bar instead of a modal dialog when it detects a security problem.

3.3 Avoiding Discrimination

Shortly after the Shmoo Group released their IDN homograph proof of concept attack in February 2005 [23], the Mozilla team announced that it would disable all IDN support for the time being. This inconsiderate decision, which ignored all countries with an legitimate interest in non-ASCII domains, was later revoked due to the enormous number of "flame wars" that took place as a consequence. Hopefully the lesson was learned that discriminating IDN domains is not an option.

[8]ICANN currently reviews the IDN guidelines, cf. `http://forum.icann.org/lists/idn-guidelines/msg00005.html`

3.4 Preferring Self-contained Solutions

A self-contained solution should not rely on an extra server in order to work. In fact, many anti-phishing and anti-malware solutions are based on a database, that is either remotely queried or regularly downloaded to the client, listing malicious Web sites or code respectively. Despite the privacy implications that arise when every visited Web site is remotely checked against a database, this practice is also considered by *Ranum* to be one of the "Six Dumbest Ideas in Computer Security" [31], namely "enumerating badness".

3.5 Alerts

The Web browser should alert the user in an understandable way about reasonable suspicion of a spoofing attack attempt. In order to deal with false positives the user should be able to maintain a whitelist of trusted sites.

3.6 Appropriate Rendering Support

One cause of visual spoofing may also be inappropriate font rendering support. The rendering system should correctly display bidirectional text and handle combining marks properly. A distinctive font and sufficiently large font size should be chosen that make it easier for the user to notice differences in similar-looking characters. If there is no glyph for a specific character available in the font the rendering system should never just show a '?' or omit the character.

3.7 Preferences

Users (especially power users) like to configure software to their needs, so a good solution should allow the user to fine-tune warnings and opt out of automatic alerts.

4. PROPOSED TECHNIQUES

Considering the requirements identified above, the proposed client-side address spoofing defense is based on visualization techniques, Web browser UI improvements that both help the user to recognize more easily suspicious addresses used in phishing scams, and on security alerts triggered when an elevated risk is detected. The proposed techniques are implemented (cf. section 5) and demonstrated in Quero [26], a Web browser extension for Internet Explorer.

4.1 Visualization Techniques

4.1.1 Digit Indication

Domain names with Arabic digits or URLs with IP addresses are marked by a small icon displayed in the right corner of the address box. The digits in the host name are highlighted when the user clicks and holds the mouse button down on the icon. Digit indication mitigates attack type 2.3.3 involving Arabic digits.

4.1.2 IDN Indication

Similar to digit highlighting, non-ASCII domain names are marked by a small icon with the caption IDN. The icon is also displayed in the address box to inform the user unobtrusively that she is visiting a non-ASCII domain. When clicking on

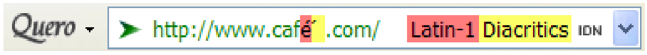


Figure 2: Quero highlighting <c,a,f,é,U+0301>

the icon, the host name is highlighted as described in the section below.

4.1.3 *IDN Highlighting*

In order to reveal special and suspicious characters in the host name, the host name is highlighted in such a way that characters from different script groups receive different background colors. Basic Latin letters and syntax characters (e.g. the dot or slash) always have a white background.

Additionally, the names of the script groups that occur in the host name are displayed to the left of the IDN icon and highlighted with the same color as that of the characters in the host name belonging to the script. The names of the script groups are important because they help detect whole-script attacks. In order to better visualize and detect the abuse of combining marks, Quero renders them separately, combining them with the space character. This is shown for the café example in figure 2.

The idea of IDN highlighting was also proposed by *Hoffman* [18], one of the authors of the IDNA standard, shortly after the homograph attack was rediscovered in 2005. I would like to note that Quero does not highlight non-ASCII domain names by default to avoid IDN discrimination. IDN indication and highlighting mitigate attack types 2.3.1, 2.3.2, 2.3.3, 2.3.4, 2.3.5, 2.3.8 and 2.3.9 (involving inadequate combining mark rendering).

4.1.4 *Secure Connection Indication*

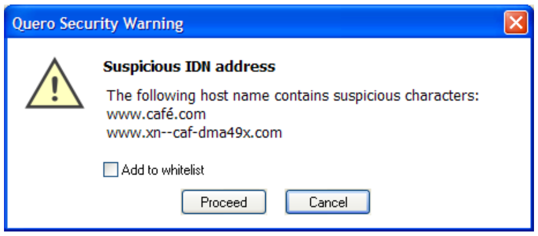
Digitally signed Web servers and securing connections by encryption are some of the fundamentals of Web security in combating phishing and the eavesdropping of sensitive information. Nevertheless, as the Shmoo Group has shown, even SSL certificates have been affected by IDN spoofing attacks. I propose that the certificate show both the Unicode and the ACE form of the domain name, and that the identity of the company or organization, the certificate was issued to, be clearly displayed and not hidden in complicated dialogs. To better visualize secured connections I adopted the Mozilla approach, turning the address box gold and displaying a small lock icon to the side. The placement of the lock icon near the URL should also encourage users to view the certificate.

4.1.5 *Core Domain Highlighting*

The “core domain” of the host name is, according to our definition, the most relevant part of the address. For domain names this is usually the second or third level domain name, depending on which one is more distinctive.

Example: for `http://www.dbai.tuwien.ac.at/` the core domain is `tuwien.ac.at`.

Because URLs contain the domain labels in least significant label first order, highlighting or rendering the core domain in bold print is very interesting because it can mitigate the more general attack types 2.1.2 and 2.3.4 assisting the user in parsing the address.



Suspicious IDN address

The following host name contains suspicious characters:
www.café.com
www.xn--caf-dma49x.com

Figure 3: Quero Security Warning

4.2 UI Improvements

4.2.1 *Address Bar Integration*

I argue that it is important to integrate all security-related information in only one place instead of spreading it between the address bar (current location), the status bar (lock icon, core domain, blocked content) or other places in the browser. I therefore suggest displaying this information consistently in a prominent place such as the address bar. A side effect of this integration may also be to save space in the browser. Quero Toolbar was designed to replace the standard address bar of IE 6 and below and promote this integration.

4.2.2 *Support for Larger Font Sizes*

In popular Web browsers the standard font size used for displaying the address is relatively small, especially for ideographic characters. In IE the text size of the address bar corresponds to the icon title font size. Quero Toolbar allows the user to increase the font size relative to this setting.

4.2.3 *Switching to ACE Form*

Although the RFC specification [14] tells us to avoid “exposing users to the raw ACE encoding”, it should be possible to access the pure ASCII domain name if necessary. Because the ACE form is what is actually stored by DNS and is guaranteed to be correctly rendered, it may be of interest. Quero displays both the Unicode and the ACE form of the domain name on security warnings and gives the user the possibility of switching to the ACE form by double clicking on the IDN indication icon.

4.3 Security Warnings

What the techniques just proposed have in common is that they do not actively interact with the user, but rather help her to verify addresses at will. The following techniques are meant to alert the user directly to problems or suspicions found in the address. I have implemented a whitelist to deal with false positives.

4.3.1 *Invalid Addresses*

Quero warns the user and actually blocks navigation to addresses that are not RFC-compliant. This includes checking for prohibited characters, length restrictions and bidirectional text requirements. An RFC-compliant implementation of IDNA mitigates attack type 2.3.6 and 2.3.7 fully and 2.3.8 partially.

4.3.2 *Suspicious Character Detection*

We define as suspicious any character that might be misleadingly used in an address to confuse the user. I developed some heuristics to detect such characters in practice. A very efficient approach is to alert the user to cases of mixed script and to display the domain name with IDN highlighting. Mixed-script detection is efficient thanks to two reasonable assumptions:

- It is much more difficult to exploit similarities within one script group than having the whole Unicode character base available. Additionally, each character from a particular script must be unique by definition.
- It is rather undesirable to mix characters in domain names belonging to different languages and using different scripts. This is because domain name owners usually want their domain name to be as short as possible and easy to input, read, recognize and memorize. Internationalized domain name labels are likely to be words, names or phrases that have a specific meaning in a language.

For the sake of simplicity I currently use script blocks [10] to partition Unicode into script groups. This is not entirely accurate because characters from the same script may be in several different blocks (not too much of a problem) and characters from different scripts may be in the same block (problematic especially for East Asian scripts). We should note that general characters (such as symbols) or combining marks used by different scripts reside in separate blocks. I also regard labels beginning with a combining mark as suspicious. The security warning for the café example is shown in figure 3. Suspicious Character Detection mitigates attack types 2.3.1, 2.3.4, 2.3.5 and 2.3.8.

4.3.3 *Missing Glyph Detection*

Because missing glyphs are all mapped to the same replacement glyph, I suggest warning the user about their presence. This security warning mitigates attack type 2.3.9 involving missing glyphs.

5. IMPLEMENTATION DETAILS

I have implemented the proposed anti-address spoofing techniques in Quero Toolbar [26], a Windows Internet Explorer Browser Helper Object (BHO) [13]. Quero is an address bar replacement that originally combined navigation and search functionality. Since Microsoft has postponed IDN support until IE7, I have decided to implement IDNA from scratch. The proposed techniques outlined in this article were successively incorporated into Quero and achieved full implementation in version 2.2.0. Quero is a COM object[9] written in C++ using the ATL/WTL framework. I chose the Internet Explorer platform because of its significant market share, excellent documentation and ease of extension. Below I describe some interesting aspects of my implementation.

5.1 The Unicode Database

Since IDNA requires Unicode normalization, I was faced with the challenge of finding the best way to integrate the Unicode Character Database into Quero. Since Internet Explorer will obviously come with its own IDNA implementation sooner or later, I wanted a solution with minimal overhead. Furthermore, I did not want to bloat Quero with an additional 1 MB overhead for the raw Unicode Database and IDNA-specific data tables. At the same time I searched for an efficient solution that would not affect IE's start-up and navigation speed.

[9]Actually, Quero comes with a second COM object for content filtering.

I singled out two basic types of functions performing with Unicode data as necessary for our purposes: The set membership and the mapping function. With the former, I observed that in many cases code points exhibiting a certain property appeared in consecutive ranges, so I utilized this fact to compress the data a little.

Finally I stored all the data in constant, ordered arrays of the following types:

```
typedef struct UnicodeInterval
{
        BYTE Plane;
        WORD Begin;
        WORD End;
} UnicodeInterval;

typedef struct UnicodeMapping
{
        BYTE Plane;
        WORD From;
        WORD To[MAPPINGSIZE];
} UnicodeMapping;
```

To save even more space I defined several `UnicodeMapping` types varying only in the `MAPPINGSIZE`. The Unicode `Plane` information is also factorized. Note that I split `Plane` into two nibbles to specify different planes for `To` and `From`, and that the `To` sequence of code points must reside completely in the same plane, which is actually the case. The membership and mapping functions use binary search to retrieve the data efficiently. In order to get the canonical composition and decomposition mapping, I store the canonical mapping table twice in different sorting orders. By reducing the Unicode data to only those code points that are really required (i.e. not prohibited or case-folded in IDNA), I was able to compress all the Unicode data into less than 100 KB.

5.2 IDNA Implementation

Having the Unicode Database and the data tables from [19] available, I implemented Nameprep for IDN [20], ToAscii, ToUnicode [14] and Punycode [7], the underlying encoding and compression algorithm. In order to implement Nameprep I have also implemented Unicode NFKC normalization [9], including algorithmic Hangul syllable processing. In ToAscii and ToUnicode Quero splits the host name into labels and performs the core domain identification.

5.3 IDN Highlighting

For IDN highlighting Quero linearly traverses the host name while checking each character for its associated script group. If the script group is different from its predecessor, Quero outputs the characters read since the last script change and highlights them with a distinct background color. Since Unicode has many more scripts than there are reasonably distinguishable background colors available, Quero dynamically assigns script groups with background colors and remembers the color used for that script in an array. To measure the extents of the outputted blocks Quero uses the Uniscribe API [4]. The mixed-script and suspicious character detection algorithm is similar to the above-mentioned one except that it stops processing as soon as it finds an unjustified script change within one label.

6. EVALUATION

Since there were no real-world IDN address spoofing attacks reported so far [29] I concentrated on exploring the whole space of possible attacks in section 2 and evaluated the mitigation impact of each technique in section 4.

7. RELATED WORK

To my best knowledge Quero is the first real implementation focusing on IDN address spoofing attacks that follows almost all user agent recommendations of the recently published Unicode Security Considerations [11]. Highlighting as a potential solution was already mentioned by *Gabrilovich* and *Gontmakher* after they discovered the homograph attack [16], and more recently by *Hoffman* [18].

Next we will take a look at major Web browsers and how they deal with IDN address spoofing attacks. Although there are several other IDN enabling solutions[10] available for Internet Explorer, they do not address IDN security issues explicitly except for those covered by RFC compliance.

Microsoft has announced native support only recently, shipping with IE7 which will impose restrictions [17] on the domain name to mitigate mixed-script and whole-script attacks based on a user-defined list of allowed languages. If any label of the domain name contains a mix of scripts or characters that do not appear in any of the allowed languages IE7 displays the raw Punycode form of the address.

Additionally, Microsoft is developing a Phishing Filter for IE7 [28] based on heuristics and a remote phishing database. A preview of the phishing filter is also available in the MSN Search Toolbar[11]. The collaborative method chosen by Microsoft has also earned criticism ranging from privacy concerns, inadequate reaction times to denial of service attack vulnerabilities both against MS phishing database servers and legitimate sites which were falsely marked as fraudulent.

Opera 8.x and Firefox 1.5 allow Unicode representation for top-level domains that are trusted to enforce a safe IDN registration policy. Besides TLD whitelisting Mozilla has added a character blacklist [30], while Opera 8.5 has implemented mixed-script detection outside of the trusted domains. There are several IDN indication extensions for Firefox available such as ShowACE, IDND[12] and IDN Info[13]. Safari, the default Web browser for Apple Macintosh OS X, leaves it up to the user to specify script groups that are allowed to be displayed in Unicode, otherwise Safari shows the ACE equivalent.

All major browsers are still vulnerable to attack types 2.1.1, 2.1.2, 2.1.3, 2.3.2, 2.3.3, and those that render Unicode to 2.3.9. Due to the cryptic appearance of the underlying encoding legitimate non-ASCII domain names that are shown in Punycode for some reason still bear the particular risk of being spoofed.

Finally, we compare both academic and industrial solutions for fighting phishing but none of them handle IDN address spoofing attacks explicitly.

SpoofGuard [6] incorporates a holistic approach to detect Web spoofing that calculates a spoof score representing the possibility that the viewed page is fraudulent. A warning dialog is displayed if the spoof score exceeds a user-defined threshold. SpoofGuard checks the URL, domain name, links, images (for fake logos) and posted data (compared with previously hashed passwords) and also uses the browser history to find similarities (edit distance) to visited domains. The solution it suggests, however, can be circumvented if explicitly targeted by an attacker.

AntiPhish [24] is an input-based approach that protects sensitive information from being transmitted to unauthorized sites. But it is still possible to by-pass AntiPhish by using non-standard input fields built with Javascript or Java applets. Moreover, the user must manually start capturing sensitive information, an extra hassle that is necessary only once for each login site.

Another idea securing passwords on the Web is password hashing [32], a method to transparently produce unique passwords for every Web site based on a master password and the current domain name of the login page.

Dynamic Security Skins [12] are an imaged-based customization solution that require the user to remember only one personal photographic image and a "low entropy" password. To identify a trusted site, the user needs to visually match two images. The proposed solutions rely on both server and client-side modifications.

Bank of America has rolled out a system called SiteKey[14] which associates a customizable picture to an account to make spoofing more difficult. The SiteKey is displayed if the customer's account is either recognized by the SiteKey cookie or by answering a personal SiteKey challenge question.

eBay Toolbar[15] has an indicator which turns green if the user is on an eBay-affiliated site or red if a known phishing site is opened (both checked by URL matching).

Netcraft Anti-Phishing Toolbar[16] relies on a remote phishing database and displays the core domain name, the site's hosting location (country) and the domain registration date in its toolbar.

TrustBar[17] is a Mozilla extension that displays a text or a user-defined logo describing the site and its certificate authority for SSL secured sites. The drawbacks of this solution are that it works only for https sites and that the logos occupy much space.

SpoofStick[18] displays the core domain in a separate toolbar.

8. CONCLUSION

I have argued in this article that eliminating (IDN) address spoofing attacks cannot be achieved solely by enforcing sophisticated registration policies. Rather a combination of a feasible registration verification process and usable client-side protection mechanisms can tackle the elevated risks of IDN domains.

According to the requirements mentioned, I proposed and implemented a combination of visual techniques including highlighting, UI improvements such as the integration of all security-related information in the address bar, and security warnings which actively alert the user to cases of mixed-script domain labels, and other reasons for suspicion. While core domain highlighting can help the user overcome the in-

[10]VeriSign i-Nav plug-in (`http://www.idnnow.com/`), i-DNS iClient (`http://www.i-dns.net/`) and IDN-OSS (`http://idn.isc.org/`)

[11]`http://addins.msn.com/phishingfilter/`

[12]`http://lingvo.org/idnd/`

[13]`http://4t2.cc/mozilla/idn`

[14]`http://www.bankofamerica.com/privacy/sitekey/`

[15]`http://pages.ebay.com/ebay_toolbar/`

[16]`http://toolbar.netcraft.com/`

[17]`http://trustbar.mozdev.org/`

[18]`http://www.spoofstick.com/`

herent complexities of Internet addresses, a novice or incautious user may still be vulnerable to similar-looking domain names (attack types 2.1.1, 2.1.3) as well as to single-script (2.3.3) and whole-script (2.3.2) attacks.

I would also like to note that highlighting has the drawback that it is not suitable for visually impaired people. At least it can be improved by selecting larger font sizes and high-contrast colors. Future work will have to deal with the latent danger in East Asian scripts which contain a very large number of similar-looking characters. Distinguishing between Simplified and Traditional Chinese characters will be the next logical step in the right direction. Protecting users against address spoofing is one major factor in combating phishing.

9. REFERENCES

[1] J. Aliprand, editor. *The Unicode Standard, Version 4.0.* Addison-Wesley Professional, 2003.

[2] S. Andersen and V. Abella. *Changes to Functionality in Microsoft Windows XP Service Pack 2, Part 5: Enhanced Browsing Security.* Microsoft Corporation, September 2004. `http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/sp2brows.mspx\#EDAA`.

[3] T. Berners-Lee, L. Masinter, and M. McCahill. *[RFC 1738] Uniform Resource Locators (URL).* Network Working Group, IETF, 1994.

[4] F. A. Bishop, D. C. Brown, and D. M. Meltzer. *Supporting multilanguage text layout and complex scripts with Windows 2000.* Microsoft Corporation, January 2003. `http://www.microsoft.com/typography/developers/uniscribe/intro.htm`.

[5] [CERT VU#490708] Microsoft Internet Explorer window.createPopup() method creates chromeless windows. `http://www.kb.cert.org/vuls/id/490708`, September 2004.

[6] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell. Client-side defense against web-based identity theft. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS '04).* The Internet Society, February 2004. `http://crypto.stanford.edu/SpoofGuard/`.

[7] A. Costello. *[RFC 3492] Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA).* Network Working Group, IETF, 2003.

[8] M. Davis. *Unicode Standard Annex # 9: The Bidirectional Algorithm.* The Unicode Consortium, 2005. `http://www.unicode.org/reports/tr19/`.

[9] M. Davis and M. Dürst. *Unicode Standard Annex # 15: Unicode Normalization Forms.* The Unicode Consortium, 2005. `http://www.unicode.org/reports/tr15/`.

[10] M. Davis and A. Freytag. *Unicode Standard Annex # 24: Script Names.* The Unicode Consortium, 2005. `http://www.unicode.org/reports/tr24/`.

[11] M. Davis and M. Suignard. *Unicode Technical Report # 36: Unicode Security Considerations (Revison 3).* The Unicode Consortium, 2005. `http://www.unicode.org/reports/tr36/`.

[12] R. Dhamija and J. Tygar. The battle against phishing: Dynamic security skins. In *ACM International Conference Proceeding Series; Vol. 93; Proceedings of the 2005 symposium on Usable privacy and security,* pages 77–88. ACM Press, 2005.

[13] D. Esposito. *[MSDN] Browser Helper Objects: The Browser the Way You Want It.* Microsoft Corporation, January 1999. `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebgen/html/bho.asp`.

[14] P. Faltstrom, P. Hoffman, and A. Costello. *[RFC 3490] Internationalizing Domain Names in Applications (IDNA).* Network Working Group, IETF, 2003.

[15] E. W. Felten, D. Balfanz, D. Dean, and D. S. Wallach. Web spoofing: An internet con game. In *Proceedings of the 20th National Information Systems Security Conference (NISSC),* pages 95–103, 1997. `http://www.cs.princeton.edu/sip/pub/spoofing.pdf`.

[16] E. Gabrilovich and A. Gontmakher. The homograph attack. *Communications of the ACM,* 45(2):128, February 2002. `http://www.cs.technion.ac.il/~gabr/papers/homograph_full.pdf`.

[17] V. Gupta. [IEBlog 12-19-2005] International Domain Names in IE7. `http://blogs.msdn.com/ie/archive/2005/12/19/505564.aspx`, December 2005.

[18] P. Hoffman. [LookIt 02-14-2005] IDN spoofing solutions with balance. `http://lookit.proper.com/archives/000302.html`, February 2005.

[19] P. Hoffman and M. Blanchet. *[RFC 3454] Preparation of Internationalized Strings ("stringprep").* Network Working Group, IETF, 2002.

[20] P. Hoffman and M. Blanchet. *[RFC 3491] Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN).* Network Working Group, IETF, 2003.

[21] IANA. *Registered IDN Language Tables,* 2005. `http://www.iana.org/assignments/idn/registered.htm`.

[22] ICANN. *Guidelines for the Implementation of Internationalized Domain Names, Version 1.0,* 2003. `http://www.icann.org/general/idn-guidelines-20jun03.htm`.

[23] E. Johanson. The state of homograph attacks. `http://www.shmoo.com/idn/homograph.txt`, February 2005.

[24] E. Kirda and C. Krügel. Protecting users against phishing attacks with AntiPhish. In *Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05),* pages 571–524. IEEE, July 2005. `http://crypto.stanford.edu/SpoofGuard/`.

[25] K. Konishi, K. Huang, H. Qian, and Y. Ko. *[RFC 3743] Joint Engineering Team (JET) Guidelines for Internationalized Domain Names (IDN) Registration and Administration for Chinese, Japanese, and Korean.* Network Working Group, IETF, 2004.

[26] V. Krammer. Quero Toolbar. `http://www.quero.at/`, November 2005.

[27] G. Markham. Phishing—Browser-based Defences (Version 0.3). `http://www.gerv.net/security/phishing-browser-defences.html`, 2005.

[28] Microsoft Phishing Filter: A New Approach to Building Trust in E-Commerce Content. `http://www.microsoft.com/downloads/details.aspx?familyid=B4022C66-99BC-4A30-9ECC-8BDEFCF0501D`, September 2005.

[29] J. Milletary. [CERT] Technical Trends in Phishing Attacks. `http:`

//www.cert.org/archive/pdf/Phishing_trends.pdf, October 2005.

[30] [MozillaZine Knowledge Base] Network.IDN.blacklist_chars. http://kb.mozillazine.org/Network.IDN.blacklist_chars, February 2006.

[31] M. J. Ranum. The Six Dumbest Ideas in Computer Security. http://www.certifiedsecuritypro.com/index.php/content/view/154/56/, September 2005.

[32] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *Proceedings of the 14th USENIX Security Symposium*. USENIX, August 2005. http://crypto.stanford.edu/PwdHash/.

[33] Secunia Advisory: Internet Explorer Address Bar Spoofing Vulnerability. http://secunia.com/advisories/12304/, August 2004.

[34] The Unicode Consortium. *Unicode Character Database, Version 4.1.0*, 2005. http://www.unicode.org/ucd/.