Journal of Network and Computer Applications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Using one-time passwords to prevent password phishing attacks

Chun-Ying Huang [a,*], Shang-Pin Ma [a], Kuan-Ta Chen [b]

[a] Department of Computer Science and Engineering, National Taiwan Ocean University, No. 2 Pei-Ning Road, Keelung 202, Taiwan
[b] Institute of Information Science, Academia Sinica, No. 128 Academia Road, Section 2, Nankang, Taipei 115, Taiwan

ARTICLE INFO

Article history:
Received 13 July 2010
Received in revised form
20 December 2010
Accepted 22 February 2011

Keywords:
Anti-phishing
Identity management
In-band password delivery
One-time password
Web security

ABSTRACT

Phishing is now a serious threat to the security of Internet users' confidential information. Basically, an attacker (phisher) tricks people into divulging sensitive information by sending fake messages to a large number of users at random. Unsuspecting users who follow the instruction in the messages are directed to well-built spoofed web pages and asked to provide sensitive information, which the phisher then steals. Based on our observations, more than 70% of phishing activities are designed to steal users' account names and passwords. With such information, an attacker can retrieve more valuable information from the compromised accounts. Statistics published by the anti-phishing working group (APWG) show that, at the end of Q2 in 2008, the number of malicious web pages designed to steal users' passwords had increased by 258% over the same period in 2007. Therefore, protecting users from phishing attacks is extremely important. A naïve way to prevent the theft of passwords is to *avoid using passwords*. This raises the following question: *Is it possible to authenticate a user without a preset password?*

In this paper, we propose a practical authentication service that eliminates the need for preset user passwords during the authentication process. By leveraging existing communication infrastructures on the Internet, i.e., the instant messaging service, it is only necessary to deploy the proposed scheme on the server side. We also show that the proposed solution can be seamlessly integrated with the OpenID service so that websites supporting OpenID benefit directly from the proposed solution. The proposed solution can be deployed incrementally, and it does not require client-side scripts, plug-ins, nor external devices. We believe that the number of phishing attacks could be reduced substantially if users were not required to provide their own passwords when accessing web pages.

1. Introduction

Phishing is a malicious activity whereby an attacker (phisher) tries to trick Internet users into providing confidential information (Dhamija et al., 2006). It is a serious problem because phishers can steal sensitive information, such as users' bank account details, social security numbers, and credit card numbers. To achieve this goal, a phisher first sets up a fake website that looks almost the same as the legitimate target website. The URL of the fake website is then sent to a large number of users at random via e-mails or instant messages. Unsuspecting users who click on the link are directed to the fake website, where they are asked to input their personal information. Although the process of setting up a fake website sounds complicated, reports show that it is much easier than before as there are now "phishing kits" (McMillan, 2006; Danchev, 2008) that can create a phishing site in a very short time. Users believe that responsible enterprises should protect them from phishing attacks; thus, in addition to the risk of personal information leakage, successful phishing attacks can seriously damage business enterprises, especially a company's brand reputation (McDonnell, 2006; O'Brien, 2006).

1.1. Anti-phishing techniques

As phishing is a serious threat to both users and enterprises, several anti-phishing techniques have been developed. In general, the techniques can be classified as either list-based or heuristic-based technologies. List-based techniques maintain a black list or a white list, or both. Many anti-phishing mechanisms use a black list to prevent users from accessing phishing sites. However, the effectiveness of black list filtering depends on the coverage, freshness, and accuracy of the list. The URLs are usually reported by Internet users or collected by web crawlers, and list maintainers are responsible for verifying whether or not the listed URLs are really phishing sites. Though a well maintained black list can filter most well-known phishing sites, it obviously cannot filter unreported, uncollected, or unanalyzed URLs. No list can

* Corresponding author.
E-mail addresses: chuang@ntou.edu.tw, huangant@gmail.com (C.-Y. Huang), albert@ntou.edu.tw (S.-P. Ma), swc@iis.sinica.edu.tw (K.-T. Chen).

1084-8045/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.jnca.2011.02.004

guarantee 100% coverage and up-to-date freshness; and list-based filtering techniques often generate false negatives.

Some anti-phishing mechanisms use white lists that contain the names of trusted domains. If a user visits an unlisted website, a white-list-based filter may block the site immediately or require the user to make decisions on the fly. The drawback of this method is that the user may become annoyed if some sites are blocked or if the system constantly requests confirmation. Sometimes, it may even be difficult for the user to make a decision. In the end, the user may loose patience with having to validate unlisted sites and decide to disable the filter mechanism.

Heuristic-based mechanisms employ several criteria to determine whether a website is a phishing site. The following are some frequently used criteria.

- *Domain name*. A phishing site may register a similar domain name to that of the target site. For example, `paypal.com` and `paypa1.com` look the same, but the latter is actually `paypa` plus the numeral `1`. Several metrics can be applied to measure the "distance" between two strings, e.g., the Levenshtein (1965) distance, the Needleman–Wunch (1970) distance, and the Smith–Waterman (1981) distance. The measured distances can be used as an index to identify possible phishing sites.
- *URL*. A phisher may attempt to mislead users by including the `@` symbol in a URL. Browsers treat the text before the `@` symbol as the name used to access a website. This allows a phisher to redirect users to a fake site using a URL like www.paypal.com@123.123.123.123. The user thinks he is visiting `www.paypal.com`, but he is actually being redirected to another site with the IP address `123.123.123.123`. Thus, it is important to check whether a URL contains special symbols.
- *Image similarities*. Some methods try to detect phishing sites by checking image and visual similarities. There are various ways to do this. For example, it can be done strictly by comparing the hash values of image files (Venkatesan et al., 2000), or loosely by comparing the distribution of colors in images or web pages (Fu et al., 2006; Chen et al., 2009; Huang et al., 2010). A website that looks the same or similar to a well-known site but uses a different domain name may be a phishing site.
- *Specific input fields*. As phishing sites usually require users to input their personal information, a phishing site has some input fields for personal information, such as passwords, social security numbers, and credit card numbers. Thus, if a web page has such fields, it may be a phishing site.
- *Keywords*. Keywords can be used to distinguish between phishing and non-phishing activities. Zhang et al. (2007) propose a technique to identify frequently used terms on a web page by computing the term frequency and inverse document frequency (TF-IDF) (Salton and McGill, 1986). The identified terms are then submitted to a search engine to obtain a list of matching sites. The checked page may be a phishing site if its URL is not included in the site list returned by the search engine.

Heuristic-based mechanisms may use only one criterion to assess web sites. For example, the basic CANTINA filter (Zhang et al., 2007) only calculates the TF-IDF score. In contrast, the advanced CANTINA filter and the SpoofGuard filter (Chou et al., 2004) use a weighted score based on several criteria. Given a set of predefined weights for each criterion, the overall score used to evaluate a site is calculated by

$$s = \sum w_i P_i, \tag{1}$$

where w_i and P_i are, respectively, the weight and the probability of a given criterion i.

List-based and heuristic-based methods can not detect and block all phishing sites (Cranor et al., 2007). Moreover, users are not always aware of alerts displayed by anti-phishing toolbars (Wu et al., 2006a). Therefore, to prevent password phishing, we believe *it would be better to develop methods that tackle the root of the problem*, i.e., methods that authenticate a user on the Web.

1.2. Motivation

Lists of active phishing sites provided by PhishTank (2009) show that 70% of the sites are designed to obtain users' login names and passwords. Once a phisher obtains a valid username and password, the phisher can login to the phished account and retrieve further valuable information about the user. As phishers target users' account names and passwords, a naïve way to reduce the number of such attacks is to authenticate a user without having to use a preset password. Instead of using a preset password, a user is given a new password every time the user wishes to utilize the web service. However, to do this, *we need a reliable secondary channel to deliver the password.*

The rationale behind the proposed solution is quite simple. We propose that users can be authenticated with one-time passwords (OTPs) delivered via a reliable secondary communication channel on demand. The user database at the server side matches a user's login name with its corresponding identity on another secondary channel. When a user wishes to access a web site, the server sends an OTP to the user through the secondary channel. On receipt of the OTP, the user can login before the password expires. The proposed solution provides three levels of protection. A phishing attack can only succeed if the attacker knows (1) the user's account name; (2) the identity of the secondary channel through which the user receives the one-time password; and (3) the password used to access the secondary channel. These constraints complicate the phishing attack process. Moreover, as preset passwords are not used, phishers can only obtain users' login names.

There are several kinds of secondary communication channels, for example, e-mails, short message services, and instant message services. We consider that *an instant messaging service is the best secondary communication channel for our solution.* In addition to delivering messages in real-time, such *services are almost ubiquitous and the cost of using them is negligible*. Although an instant messaging service is not totally secure, with proper design and configuration, it can be a good medium for delivering passwords. The infrastructure for instant messaging services is already available on the Internet; and downloading the client side program and obtaining a user account are free of charge. Therefore, to utilize such services, a website only needs to set up an identity management database and run an instant messaging bot program to send one-time passwords.

1.3. Scope and limitations

It is probably impossible to prevent phishing attacks completely. Thus, the purpose of the proposed solution is to *reduce the number of password phishing attacks*. That is, a website that applies our solution can reduce the probability that password phishing attacks will be successful. However, as we leverage instant messaging services to deliver one-time passwords, those services may also become targets for phishers. We discuss this issue further in Section 3. Apart from phishing, other types of attacks try to steal users' account names or sensitive information. Many websites allow a user who has logged in to maintain his/her on-line status for a period of time by storing persistent cookies; however, the cookies might be stolen via cross-site script (XSS)

Please cite this article as: Huang C-Y, et al. Using one-time passwords to prevent password phishing attacks. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.004

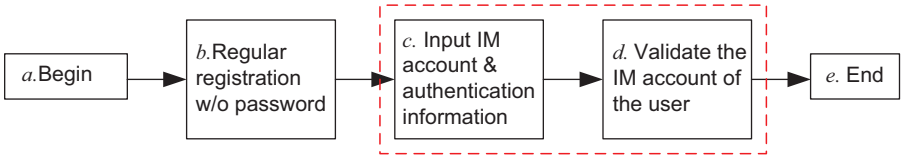

Fig. 1. The modified user registration process. The two extra steps required in our solution are enclosed in the dashed rectangle.

attacks or cross-site request forgery (CSRF) attacks. These two types of attacks are beyond the scope of this paper. Moreover, we do not discuss attacks that compromise users' computers, e.g., root kits and key loggers, which can gain total control of a user's personal computer or monitor a user's interactive behavior. As all user inputs can be monitored, stored, and passed on, attackers can obtain a great deal of sensitive information, including user names and passwords. Finally, we do not discuss attacks designed to compromise a website. If such attacks are successful, attackers can access any sensitive data stored on the server.

1.4. Paper organization

The remainder of this paper is organized as follows. In the next section, we discuss the proposed solution in detail. Section 3 provides an analysis of the security of our approach. In Section 4, we consider issues that may arise when implementing the solution. Section 5 contains a review of related works, and Section 6 summarizes our conclusions.

2. The proposed solution

The proposed solution involves two processes: a registration process and a login process. First, we introduce the parties and assumptions used in the solution. There are four parties: websites, instant messaging (IM) service providers, users, and phishers. As we leverage a secondary channel to deliver secret passwords, we assume the primary channel, i.e. the HTTP protocol, is secure. This assumption is reasonable because most web browsers and commercial websites use the SSL protocol to encrypt HTTP traffic. To use the authentication service, we assume that a website has joined one or more instant messaging networks and uses the network(s) to communicate with users. Similarly, a user who wishes to utilize a service must install at least one IM client that the website supports. Obtaining an IM account is quite simple, and we assume that an Internet user can have multiple IM network accounts. As instant messaging services are free, the cost of joining a social network is zero for both websites and users. It is even unnecessary for a user to install an IM client, since many operating systems have built-in clients, e.g., Microsoft's Windows Messenger (Microsoft Corporation, 2009a) and Apple's iChat in Mac OS X (Apple, Inc., 2009). Most IM clients support automatic startup and sign-in; thus, with the proper setup, a user can join an IM network seamlessly when the user is on-line. We also assume that IM service providers are aware that their services are used to deliver one-time passwords. Finally, we assume that phishers are aware of the existence of the solution.

2.1. The registration process

Usually, the registration process for a web service involves the following steps. A user must choose a unique account name, select a login password, fill in all the required information fields, and provide at least one type of personal contact information (usually an e-mail address). Our goal is to eliminate the use of a fixed or preset password to access a website, so we modify the above registration process, as shown in Fig. 1. Most of the registration steps are the same, but users are not required to choose a login password. Instead, they must complete an additional IM account registration process to take advantage of our solution.

Figure. 2 shows the IM account registration process. Note that if one step fails during the IM registration process, both the regular registration process and the IM registration process will fail. If a user continues to register his/her IM account name, the website will ask for two items of IM authentication information. The first is the *instant messaging account* used for receiving one-time passwords. The website should list all the IM services that it uses to deliver one-time passwords so that the user can choose the preferred service. The user is then asked to provide an IM account name. As mentioned earlier, a user must use an IM account that is supported by the website; otherwise, the user will not be able to use the authentication service. The second item of authentication information is the answer to a predefined or user-defined *security question*, such as the user's middle name, birthday, a relative's name, or a pet's name. The use of a security question is not mandatory. It depends on the website provider's policy or the user's wishes. However, such questions make the authentication process more secure. We discuss this issue further in Section 3.

With the user-provided IM authentication information, the website has to confirm the validity of the user's IM account. To do this, it sends the user a confirmation page containing a completely automated public Turing test (CAPTCHA test), which can distinguish between computers and humans (Naor, 1996; von Ahn et al., 2003). This prevents the confirmation process from being abused by robots because, at this stage, the website has to add the user-provided IM account name to its contact list and send an authentication message for account validation. If the user passes the CAPTCHA test, a series of exchanges take place between the website, the IM service, and the user. The steps of the process are as follows:

1. An IM *account validation page* is sent to the user. The page contains the name of the IM account used by the website and *a short random string t* (called a session token). The page also includes a password field for the user to input the one-time password the user is assigned.
2. If the user's IM account is new to the website, the user is asked for permission to add his/her IM account name to the website's contact list. This guarantees that the website can always send instant messages to the user.[1] The step can be skipped if the website's IM account has been approved by the user and vice versa.
3. After the IM account has been approved by both the website and the user, the website sends an account validation message to the user via the designated IM service. The message contains the session token t and a one-time password p.
4. On receipt of the message, the user verifies that the sender's IM account name and the session token t are the same as those

[1] Although some IM clients allows users to accept incoming instant messages from anonymous or unknown senders, users may wish to disable this feature for security or privacy considerations.

Please cite this article as: Huang C-Y, et al. Using one-time passwords to prevent password phishing attacks. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.004

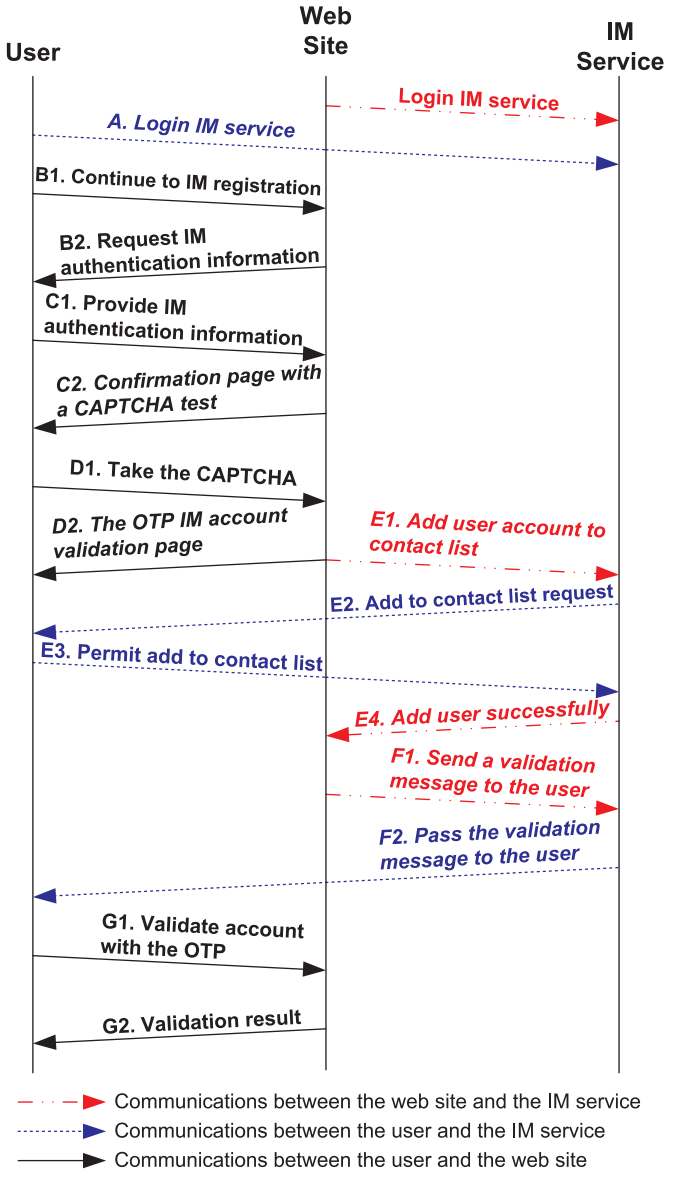

Fig. 2. The instant messaging account registration process.

shown on the account validation page. If the message is verified, the user then adds the OTP received in step 3 to the page and submits it to website.

The website completes the validation process after it receives the correct OTP from the user. For example, if the user enters p' in the password field of the account validation page, the validation process will be approved if p' is identical to p. Validation of the user's IM account completes the user registration process. The account can then be used for authentication whenever the user wishes to sign into the website in the future.

2.2. The login process

Once a user has registered successfully, the user can log in with the OTP assigned by the website. The steps of the process are shown in Fig. 3. We assume that the website has already logged into the instant messaging (IM) service. The login procedure is comprised five steps, *A–E*, as indicated in the figure. First, the user logs into his/her instant messaging account to obtain an OTP from the website. As many IM clients can start up and sign in

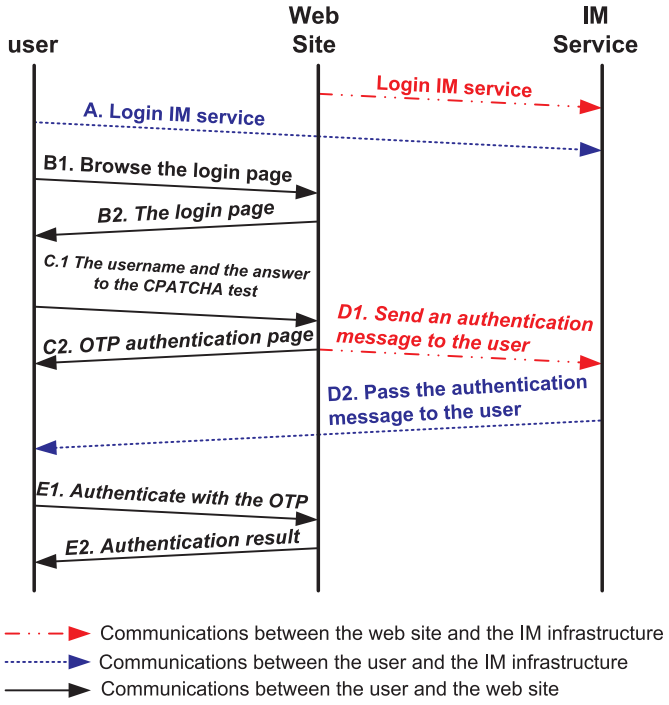

Fig. 3. The user login process.

> You have requested to log into the website holding the session token DwZ6EXgBREGT. You are connected from the IP address 11.22.33.44. Your pet's name is Max. Please log in with the one-time password gREduqmYz5QB.

Fig. 4. An example of an authentication message.

automatically, the first step can be performed in the background. Next, the user starts the actual login process by browsing the login page, which contains an input field for the user's account name and the CAPTCHA test. If the user passes the test, the website displays the input page for the user's OTP in the third step. This page must be displayed even if the user's account name is not recognized by the website. In addition to an OTP input field, the page also contains a randomly generated *session token t* and the user's *IP address* to connect to the Internet. If the user's account name is valid, the website checks the user's registered IM account and *sends an authentication message to that account*. The message must contain the previously generated session token t, and the randomly generated one-time password p. It may also include the user's IP address and the optional security question with the answer provided by the user. An example of an authentication message is shown in Fig. 4.

Finally, on receipt of the authentication message, the user can verify its validity of the authentication message by (1) comparing the session token received by the IM client with the one shown on the OTP input page; (2) comparing the IP address received by the IM client with that shown on the OTP input page, and (3) checking the answer to the security question. If the authentication message is valid, the user enters the assigned OTP on the OTP input page. On receipt of the OTP, the website has to make sure that the user submits the OTP from exactly the same IP address as the user requests to log in. It then authenticates the user by comparing the OTP provided by the user with the previously assigned OTP. For example, if the user enters p' in the password field of the OTP input page, the login is deemed successful if p' is the same as the OTP p generated by the website.

Please cite this article as: Huang C-Y, et al. Using one-time passwords to prevent password phishing attacks. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.004

As we use one-time passwords in both the instant messaging account registration process and the login process, the life span of passwords must be limited. If a user inputs an invalid OTP more than n times or a delivered OTP has not been used within m seconds, the website will invalidate the current password and stop the process. The user must then start the process again to generate a new OTP. Note that n must be a very small number and m must be a suitably short time, for example, $n=3$ and $m=30$ s. In addition, each OTP should comprise a variety of letters (both upper- and lower-case) as well as numerals to reduce the probability that it could be guessed.

3. Security analysis

In this section, we examine the strengths and weaknesses of the proposed solution in terms of security and consider possible improvements.

3.1. Mutual authentication

Mutual authentication is a security feature that enables a server (the website) to authenticate a client (the user) and vice versa. In our solution, a website authenticates a user by asking him/her to input the one-time password already assigned by the website. As the instant message account used to receive OTPs is only known to the user, other users cannot guess the correct OTP. Meanwhile, the user authenticates the website by first checking the sender of the authentication message. We assume that an IM client can block messages from anonymous or unknown senders; thus, only authorized senders are able to send authentication messages. The user can also check the correctness of the security question and the corresponding answer in the authentication message. The absence of a security question, an incorrect security question, or an incorrect answer to the security question will cause authentication of the website to fail.

3.2. Compromising the proposed solution

The strength of the proposed solution lies in the *hidden relationships* between three components and the fact that it is difficult to compromise the components. The components are the user's website account, the website's IM account, and the user's IM account. Therefore, to compromise the solution, an attacker must at least know which of the components are associated with one another, and then target the components accordingly. We discuss how the proposed solution can be compromised in two scenarios. The first involves logging into a website with the identity of a user; the second involves tricking a user to log into a fake website. For ease of discussion, we call the attacker *Alice* and the user *Bob*.

In the first scenario, if *Alice* wants to log into a website with *Bob*'s identity, she must know *Bob*'s web account name[2] and the correct IM account that *Bob* uses to receive OTPs from the website. However, as we explained in the third step of the login process in Section 2.2, the website always shows an OTP input page, irrespective of whether the user's account name is recognized by the website's database. Therefore, *Alice* cannot determine the validity of any account on the website. Moreover, if *Alice* tries to *guess Bob's account name*, the incident will be reported to *Bob* via an authentication message immediately.[3] Even if *Alice* knows the web account that *Bob* uses when visiting the website, she still has to discover about *Bob*'s associated IM account and compromise it; otherwise, she will not be able to log into *Bob*'s web account. If she tries to compromise *Bob*'s IM account, *Bob* will be warned about the attack.

In the second scenario, if *Alice* wants to trick *Bob* to log into her own fake website, she needs to know the correct IM account name that *Bob* uses to receive OTPs and the correct IM account name that the website uses to send OTPs. She also needs to know the security question provided by *Bob* as well as the answer to that question. However, the above requirements about *Alice*'s knowledge can be relaxed so that she can launch an attack if she only knows one of *Bob*'s IM accounts used to reach *Bob*, and one of the approved IM accounts of *Bob*'s friends that is authorized to send messages to *Bob*. In this scenario, if *Bob* is tricked into logging into the fake website, he will be asked to input his Web account name and complete the CAPTCHA test. Then, the fake site will display an OTP input page and wait for *Bob* to input the correct one-time password. At this stage, *Alice* sends an authentication message to *Bob*'s IM account using either the website's IM account or the IM account of one of *Bob*'s friends. Suppose *Alice* can send instant messages using any identity she chooses. If *Bob* is aware of security problems, he will be able to identify a false authentication message if the security question and the answer are incorrect or if the IM account used to send or receive the authentication message is incorrect. Otherwise, the attack may be successful.

Under the proposed OTP solution, users do not require a preset password to log into a website; thus, no passwords can be stolen. However, a phisher can still steal a user's web account name. Recall that we assume an IM client can block messages from anonymous or unknown senders. Hence, in both of the above scenarios, *Alice* must compromise some IM accounts to pass the authentication process. We discuss the security issues related to instant messaging services in Section 3.6.

3.3. Authentication in an untrustworthy environment

Sometimes users have to sign into their web accounts in an untrustworthy environment, e.g., accessing a personal bank account using an Internet cafe's publicly shared computer. Our solution is also applicable to such cases. However, to facilitate the login process the user must have another trusted Internet-connected device, such as a smart phone. In this case, the user must sign into his/her IM account via the trusted device, and then just follow the OTP login instructions. The required OTP is still delivered to the IM account registered for the web account. As the IM account is now accessed via the trusted device, the user can read the authentication information from the device and then log into the website on the untrustworthy device (i.e., the publicly shared computer in the above example). Some IM services allow users to redirect instant messages to their mobile phones via the short message service (SMS) when they are off-line. Therefore, users can also receive authentication messages via their mobile phones.

3.4. The man-in-the-middle attack

In a man-in-the-middle (MITM) attack, a malicious user located between two communication devices can monitor or

[2] Here, we assume that a user's web account name is not necessarily identical to his/her real name, as it is very common for people to have the same name. Moreover, a website may have its own requirements for the account name, e.g., it must be two words separated by a dot.

[3] Many IM services allow users to receive messages when they are off-line. Thus, even if users are off-line when their accounts are attacked, they will be notified as soon as they go back on-line.

Please cite this article as: Huang C-Y, et al. Using one-time passwords to prevent password phishing attacks. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.004

record all messages exchanged between the two devices. Suppose an attacker tries to launch an MITM attack on a user and a website, and that the attacker can monitor *all messages* sent to or received by the user. In the proposed solution, we assume that the primary channel, which is used to deliver the user's login name, is encrypted; thus, an eavesdropper cannot associate an OTP with a user's web account name. In addition, even if an OTP is discovered, it is only valid if it is sent to the website via the user's computer, which is identified by the IP address. After the OTP has been used for authentication, it becomes meaningless to both the user and the website. Now, suppose the attacker can discover both the user's web account name and the OTP assigned for the current session. Since the life span of the OTP is very short, it would be of little use to the attacker. In Section 2.2, we suggested that the life span, m, should be no longer than 30 s, which is sufficient for the user to log into the website after receiving the OTP. Thus, the only potentially useful information that might be stolen via an MITM attack is the optional security question and its answer. We discuss the prevention of information leakage from the authentication message in Section 3.6. Here, we consider the scenario where an MITM attacker may be able to retrieve sensitive information after the user has been authenticated. Such attacks cannot be prevented by authentication mechanisms and are thus beyond the scope of this paper. Websites should apply proper encryption techniques to prevent the theft of sensitive data via MITM attacks. An alternative solution involves re-authenticating a user multiple times. The cost of delivering OTPs is minimal. Therefore, to ensure that an attacker cannot use a stolen session password to access and compromise the system, a web service could re-authenticate the user before a critical operation is performed.

3.5. IP-spoofing attacks

It is difficult to attacks the proposed solution using IP-spoofing technique since the primary communication channel is encrypted, as we assumed in Section 2. If an attacker plans to launch an IP-spoofing attack, the attacker has to break the encrypted primary channel first. Then, the attacker must be able to intercept the OTP message received by the user. Finally, the attacker has to use the OTP before the user uses it. For the first step, although network traffic flows through the encrypted primary channel can be captured, it is not able to be decoded nor replayed. For the second step, the attacker must be either a man-in-the-middle or another user within the same local network. This restricts the locations that are able to launch IP-spoofing attacks. If the attacker uses the same IP address as the user in the same local network concurrently, it can be detected by the user. Finally, the lifetime of an OTP and the corresponding session token is only a few seconds. After the OTP has been sent to the website, both the OTP and the session token become invalid. Therefore, it is not possible for an attacker to use captured OTPs and session tokens to login a protected website even if the attacker is able to access the website via the same IP addresses.

3.6. Security enhancements for instant messaging services

A number of security enhancements could be made to strengthen the proposed solution. They are not essential, but they would make the solution more robust to advanced attacks. We discuss the enhancements below.

Encryption of instant messages. The proposed solution does not require IM clients to encrypt messages sent between two communication devices. As far as we know, most IM clients do not have built-in message encryption features; hence, they can be targeted by MITM attacks, as discussed in the previous subsection. Currently, it seems that Skype is the only IM system that supports end-to-end message encryption (Mannan and Oorschot, 2004; Baset and Schulzrinne, 2004; Skype Limited, 2009). However, some third-party companies, such as Binary Inertia, LLC (2009) and Secway France (2009), are developing client-side proxy software that intercepts and encrypts IM messages, and then sends them to IM networks. As a result, instant messages can be exchanged securely between two end clients that have installed the software. We suggest that an IM client should have a built-in message encryption feature so that the user's privacy can be guaranteed.

Customized interpretation of authentication messages. The authentication messages in the proposed solution are always sent in plain text. Even if a plain text message is encrypted by the underlying IM service, the IM service provider may still learn the content of the message. If a website requires a higher security level, i.e., it does not want the IM service provider to understand what is being transmitted, a customized interpreter of authentication messages must be installed. Some IM clients allow developers to install customized plug-ins on the IM client (AOL, LLC, 2009; Skype Limited, 2008; Yahoo! Inc, 2008; Microsoft Corporation, 2009c). With plug-in support, a website can send obfuscated authentication messages over IM networks and only allow the customized plug-in to recover the messages. This also reduces the probability that sensitive information will be leaked to attackers. Hence, we suggest that an IM client should support the development of customized plug-ins for interpreting messages.

Always block messages from unknown users. Most users have received annoying instant messages from anonymous or unknown users. This happens because many IM clients allow the delivery of unauthorized messages by default; that is, the message senders are not listed on the recipients' contact lists. As a result, it is easy for an attacker to forge an authentication message and trick a user to log into a fake website. In most IM clients, the feature that allows receipt of messages from unauthorized users can be disabled; thus, we suggest that users disable the feature if it is not disabled by default.

3.7. Anti-phishing for instant messaging services

If login passwords are delivered via IM networks, the networks may become popular targets for phishers. This is to be expected. As user-designated passwords are not used during the authentication process, phishers cannot imitate a website that adopts the proposed OTP solution. For this reason, attackers may focus on phishing IM accounts rather than regular website accounts. Therefore, it is important to develop anti-phishing measures for IM services. In this subsection, we discuss possible solutions and strategies to protect IM users against phishing attacks.

We believe that protecting IM users against phishing attacks is easier than protecting regular websites for a number of reasons. As most IM services have their own platform-dependent IM clients, the chances of an attacker phishing an IM account successfully are lower. To ensure an attack is successful, the phisher must (1) insert a crafted IM client into an IM user's personal computer, or (2) trick the IM user into opening a phishing web page. Detailed discussion of the defense mechanisms for the first case is beyond the scope of this paper. In general, if an attacker can insert a program into a user's computer, the attacker can launch any type of attack on the computer. However, there are more efficient ways to obtain private information from a user if a program can be inserted. The second case may occur if an IM service has a web-based client. A phisher may imitate the web page of the client and trick IM users into

Please cite this article as: Huang C-Y, et al. Using one-time passwords to prevent password phishing attacks. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.004

logging into their accounts via that fake page. This type of attack cannot be eliminated completely, but it can be detected by existing anti-phishing solutions for regular websites. An additional benefit is that the number of well known IM services is limited. An anti-phishing solution can be developed specifically for IM services only and thereby maximizing phishing detection rates.

4. Implementation issues

The proposed solution can only be implemented at the web server side. As many users already use a preferred instant messaging network, the following question arises: How can a website systematically contact users in IM networks? The solution is that a website must implement an instant messaging (IM) bot program, which is responsible for logging into the IM network automatically, handling IM contact list operations, and sending authentication messages to users. There are several instant messaging (IM) networks. In general, an IM bot that contacts users in multiple IM networks can be implemented by three models: the protocol specification model, the protocol wrapper model, and the client wrapper model. In the protocol wrapper and client wrapper models, an IM bot can be implemented using an application programming interface (API) provided by the IM service provider or a third party. From a programmer's point of view, the two models look similar, but they are completely different. Next, we discuss the properties of the three models and some issues that may be encountered when implementing IM bots with the models.

4.1. The protocol specification model

The most direct way to implement an IM bot is to follow the protocol specification of the targeted IM network. However, it is not so simple. As most commercial IM services do not release their protocols to the public, it is not easy to implement a customized IM bot program. Although some unofficial IM clients support proprietary IM protocols, most of them are implemented based on reverse engineered IM protocols (Pidgin Project, 2009; Mintz, 2004; MSNPiki, 2007). To the best of our knowledge, AOL's AIM is the only proprietary IM protocol that has been released to the public (Wikipedia, 2009; AOL, LLC, 2008b). In addition to commercial IM protocols, there are several open standards. The most well-known standard may be the extensible messaging and presence protocol XMPP (a.k.a. Jabber previously) (Saint-Andre, 2004), which has been adopted by Google's Talk IM service (Google, Inc, 2009). Although this is the most direct way to implement a bot, in practice, it is difficult to achieve with only a protocol specification. A developer should consider using the protocol wrapper model instead, if it is available.

4.2. The protocol wrapper model

We think this is the best model for implementing an IM bot. With a protocol wrapper library, a developer can easily implement any customized IM clients, including an IM bot. Moreover, the developer does not need to know how the underlying protocol works because all operations are performed transparently by the API. The implemented IM bot can also access the IM network directly. The API of AIM's SDK is a representative protocol wrapper model that can be downloaded free of charge from AIM's official developer website (AOL, LLC, 2008a). It is quite simple to write an AIM bot with SDK. A simple bot, which can send and receive instant messages, can be written with only a few hundred lines of codes. A number of unofficial protocol wrappers are provided by third parties, such as those developed by open source communities. However, some of these implementations are also based on reverse engineered IM protocols. A single IM protocol may have multiple protocol wrappers from different communities. Hence, a developer can choose a preferred variant if an official protocol wrapper is not available.

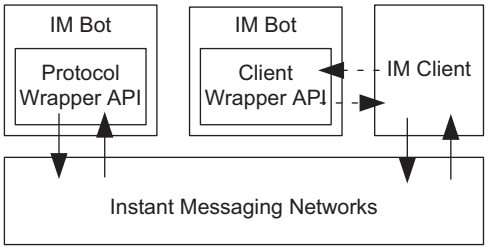

Fig. 5. The protocol wrapper model vs. the client wrapper model.

4.3. The client wrapper model

An IM service provider may only allow its own clients to access the network in order to protect their core technology and restrict the use of their IM network. The third model can be used to find a balance between protection and functionality. The difference between the client wrapper model and the protocol wrapper model is that, in a client wrapper API, an IM bot cannot access the IM network directly. Instead, as shown in Fig. 5, it must *control an official IM client* to access the network. Skype is an example of this type of network, and the company has released its developer API to the public (Skype Limited, 2008). However, there are no protocol documents available and it seems that no one completely understands how the underlying Skype protocol operates. Developers, except Skype staff, who want to join Skype's network or handle Skype messages *must* command an official Skype client. In other words, a Skype bot is an external program that controls the behavior of a running Skype client and handles events received by that client. Since a Skype bot cannot run in the background, it must run in the foreground desktop environment, but this is not suitable for writing a service. Another restriction is that the Skype API cannot ask a client to sign into the Skype network. The sign-in action can only be performed by the user or by the client's "sign me in when Skype starts" option. If a Skype client logs out of the network, the IM bot will lose the connection to the network completely.

A client wrapper API usually controls an IM client via inter-process communications (IPC). For example, in Microsoft Windows operating systems, the Skype wrapper API talks to a Skype client via the `WM_DATA` messaging service. By contrast, in Linux operating systems, the API talks to a Skype client via the `D-BUS` or `X11` messaging service. We find that the performance and stability of such APIs are not as good as those in the other two models. However, if a developer wishes to build a Skype IM bot, the client wrapper model is the only choice.

4.4. Managing contact lists

An IM bot's most important function is to manage contact lists. Although many IM protocols restrict the number of users on the contact list, this is not a drawback for the proposed solution. In fact, we suggest that the number of users should be limited even if the protocol does not impose a restriction, as it enhances both portability and security. To ensure portability, we can set the number of users to levels that comply with the restrictions of various IM protocols. In terms of security, it would be better if the maximum number of users is set at a small number. Then, an IM bot has to control several IM accounts simultaneously and ensure that it

Please cite this article as: Huang C-Y, et al. Using one-time passwords to prevent password phishing attacks. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.004

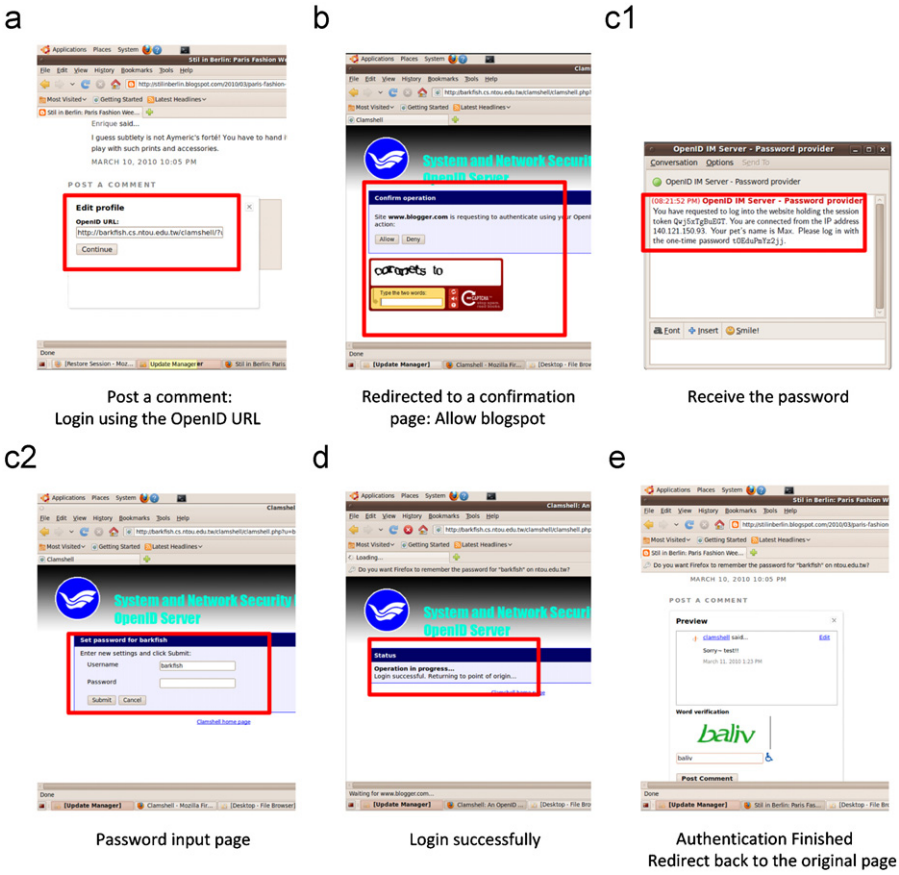

Fig. 6. Example: a user leaves messages on the blogspot website using the proposed service.

reaches all the website's users. Suppose a website has 1,000,000 registered users and the maximum number of users on the contact list is 500. The IM bot must control at least 2000 different IM accounts simultaneously in order to handle login requests from any of the website's users. As mentioned in Section 3.2, tricking a user to log into a fake website requires knowledge of the exact website IM account authorized by the user. When a website has to manage a large number of IM accounts, it becomes difficult to launch such an attack. In the above example, even if the attacker knows all the IM accounts controlled by the website's IM bot, in the worst case scenario, the attacker would have to try all the accounts to find the correct one authorized by a single user.

It is possible to write an IM bot that controls a large number of IM accounts at the same time. However, for the above reasons, we strongly recommend limiting the maximum number of users allowed on the contact list.

4.5. Integrated with existing services

It is one major obstacle for a new service to deploy and integrate it with existing services. Hence, to minimize the deployment cost, we have tried to integrate the proposed solution with the OpenID service. OpenID Recordon and Reed (2006) is an open standard that allows users to be authenticated in a distributed manner. The OpenID protocol does not rely on a central authority to authenticate a user. Moreover, neither services nor the OpenID standard limit a specific means to authenticate users, allowing for approaches ranging from the common (such as static passwords) to the novel (such as smart cards or one time passwords). The proposed solution can be implemented as an OpenID provider and hence seamlessly integrated with websites that support OpenID. A user registered to the proposed service is able to receive randomly generated one-time passwords by simply logging in with the URL of the OpenID provider. An integrated example given in Fig. 6 shows how a user leaves messages on the Blogger (2010) website using the proposed service.

4.6. Improvement of usability

We have shown that it is possible to mitigate password phishing attacks by delivering one-time password via a second real-time channel. A proof-of-concept implementation also shows that it works well with existing websites. However, before input a password, a user have to verify that the session token and the IP address appeared in the received OTP message are both correct. It might be a demanding task for a normal user to perform such a verification process manually. Hence, to simplify the task, we suggest a possible improvement to automate the process. To do this, a proxy can be implemented and run on the local computer of the user to intercept messages received by the web browser and the instant messenger. Therefore, the proxy is able to know the session token given by the website and then compare the token against that one received along with the OTP message. In addition, the IP address of the request originator can be verified as well since the proxy is able to detect the IP address used by the computer. If an OTP message is verified, the proxy can show only the password for the user to login the visiting website.

5. Related work

There are various methods for protecting users from phishing attacks as they surf the Web. Most web browsers have built-in anti-

phishing solutions (Mozilla Project, 2009; Microsoft Corporation, 2009b) that block phishing sites based on well maintained black lists and white lists. For example, Firefox blocks malicious web sites based on lists from Google, Inc (2007) and StopBadware.org (2009). Clearly, the effectiveness of phishing detection depends on the coverage, freshness, and the accuracy of the employed list. Many other solutions can be implemented as add-ons, plug-ins, or extensions for web browsers (Chou et al., 2004; Google, Inc, 2007; NetCraft, Ltd, 2004; Zhang et al., 2007; Wu et al., 2006b). As mentioned in Section 1, in addition to using black lists, some approaches employ heuristics to distinguish between phishing and non-phishing sites (Levenshtein, 1965; Fu et al., 2006; Chou et al., 2004; Zhang et al., 2007; Salton and McGill, 1986; Fette et al., 2007). Cranor et al. (2007) evaluate several popular anti-phishing toolbars and conclude that the phishing detection rate is not good enough. While the better implementations can detect more than 75% of phishing attacks, the least effective ones can detect less than half of such attacks. Furthermore, a toolbar with a high detection rate may also have a high false positive rate, i.e., it blocks non-phishing sites.

Researchers have also tried to modify the authentication process so that it is more difficult for phishers to compromise a network. The user login process may be made more secure with the assistance of external devices. Wu et al. (2004) propose using a short message service to confirm a user's login request via WAP-enabled mobile phones; and Parno et al. (2006) suggest storing authentication tokens in a trusted device, e.g., a mobile phone. When a user visits a website, the authentication messages between the web server and the trusted device are exchanged via the web browser. However, these solutions may only work with *sophisticated* external devices, e.g., those that support WAP or have built-in bluetooth interfaces. In addition to increasing the cost of devices, such solutions may be not scalable if they are deployed worldwide.

Wu et al. (2006b) present a web wallet tool that changes the process of submitting forms. The web wallet is a browser extension in a form of a sidebar. When a web page asks a user to provide his/her personal information, including the account name and password, it disables all input fields on the page, checks the validity of the web page, and then displays another interface for the user to enter the required information. As users must send their personal information via the web wallet interface, the probability of being phished is reduced. However, the effectiveness of the tool depends on its ability to identify valid web sites. Another tool, called passpet, focuses on login fields (Yee and Sitaker, 2006). It records a user's account name and password for each valid website that the user visits in a secure storage. When the user wants to log into a website, the user simply clicks on the passpet icon and the tool will complete the login fields automatically. Passpet only provides account information for a valid website, which is verified by its domain name and site certificates. The probability of passwords being phished is therefore reduced.

Ross et al. (2005) propose a solution called PwdHash is proposed. The authors observe that users often employ the same password to log into multiple websites. If the password used for one website can be phished, the attacker can use it to log into other websites visited by the user. PwdHash is designed to solve this problem. For each user password p, PwdHash always computes $p' = hash(p,salt)$ and sends p' to websites instead of sending p directly. The value of *salt* is usually derived from the domain name of the visited website. Thus, even if a user chooses the same password for all websites, the actual passwords sent to the website are varied for different domains. As phishing sites usually reside in different domains to the targeted websites, an attacker cannot phish the right password to log into a targeted website. Even if phishers can obtain the right password via pharming attacks (Karlof et al., 2007), the phished passwords can only be used to log into the targeted websites. The BeamAuth mechanism (Adida, 2007), a two-factor authentication mechanism with a bookmark, also tries to obfuscate passwords. When a user opens a website's login page, the user must click on a preset bookmark, which is generated during the registration process and delivered to the user via a secondary channel, such as email. The bookmark embeds the user's name and a fixed secret token in the *anchor* part of the bookmarked URL. On clicking the bookmark, the JavaScript code embedded in the login page reads the user's name and the secret token. When the user submits the login information, it is intercepted and the submitted password p is then substituted with HMAC$_{secret\text{-}token}(p)$. Consequently, only the user who knows the secret token can log into the website. One limitation of BeamAuth is that it is vulnerable to pharming attacks when obfuscated passwords are sent via unprotected connections.

Unlike the above approaches, our solution only requires server side deployment. It is not necessary to make customized modifications of web browsers as not all web browsers support user customization. Moreover, it is not necessary to enable script support in web browsers because such support introduces additional threats to users (Anupam and Mayer, 1998; Yu et al., 2007). Although using an external device enhances the security level, such devices are not mandatory.

6. Conclusion

The goal of password phishing is steal users' personal information, such as account names and passwords. Although there are a number of methods for detecting phishing behavior and protecting users from attacks, it is not possible to detect all phishing sites. In this paper, we propose a solution that tries to reduce the number of password phishing attacks by authenticating users with one-time passwords instead of fixed user-defined passwords. One-time passwords are delivered via ubiquitous communication infrastructures like instant messaging services. Thus, any website can take advantage of the proposed solution by installing instant messaging bots at the server side only. This reduces the costs of deployment to almost zero, and also improves the practicability of the proposed solution. A potential drawback of our solution is that the instant messaging accounts may become key targets for phishers. However, it is relatively easy to detect phishing activities with existing anti-phishing techniques if the attacks only target a small number of well-known websites; thus, the drawback is not a major issue. When users no longer have to log into websites with static passwords, the number of successful password phishing attacks will be minimized.

Acknowledgments

This research was supported in part by Taiwan Information Security Center at NTUST(TWISC@NTUST), National Science Council under the grants NSC 100-2219-E-011-002. We would like to thank Mr. Zun-Yu Yang for his effort on implementing the prototype of the proposed technique. We would also like to thank the anonymous reviewers for their valuable and helpful comments.

References

Adida B. Beamauth: two-factor web authentication with a bookmark. In: CCS '07: proceedings of the 14th ACM conference on computer and communications security. New York, NY, USA: ACM; 2007. p. 48–57.

Anupam V, Mayer A. Security of web browser scripting languages: vulnerabilities, attacks, and remedies. In: SSYM'98: proceedings of the 7th USENIX security symposium; 1998.

AOL, LLC. AIM custom clients, 2008a. [online] 〈http://dev.aol.com/aim/clients〉. URL: 〈http://dev.aol.com/aim/clients〉.

AOL, LLC. OSCAR protocol, 2008b. [online] 〈http://dev.aol.com/aim/oscar/〉. URL: 〈http://dev.aol.com/aim/oscar/〉.

AOL, LLC. AIM module plugin API reference, 2009. [online] 〈http://dev.aol.com/aim/plugins/module_plugin_reference〉. URL: 〈http://dev.aol.com/aim/plugins/module_plugin_reference〉.

Apple, Inc.. Apple—Mac OS X Leopard—Features—iChat, 2009. [online] 〈http://www.apple.com/macosx/features/ichat.html〉. URL: 〈http://www.apple.com/macosx/features/ichat.html〉.

Baset SA, Schulzrinne H. An analysis of the skype peer-to-peer internal telephony protocol, 2004. Arxiv preprint cs.NI/0412017. [online] 〈http://arxiv.org/abs/cs.NI/0412017〉. URL: 〈http://arxiv.org/abs/cs.NI/0412017〉.

Binary Inertia, LLC. AIM encrypt—free security certificate for AIM! 2009 [online] 〈http://www.aimencrypt.com/〉. URL: 〈http://www.aimencrypt.com/〉.

Blogger. Blogger: create your free blog, 2010. [online] 〈http://www.blogger.com/〉. URL: 〈http://www.blogger.com/〉.

Chen K-T, Chen J-Y, Huang C-R, Chen C-S. Fighting phishing with discriminative keypoint features. IEEE Internet Computing 2009;May:30–7.

Chou N, Ledesma R, Teraguchi Y, Boneh D, Mitchell JC. Client-side defense against web-based identity theft. In: NDSS'04: proceedings of the 11th annual network and distributed system security symposium; 2004.

Cranor L, Egelman S, Hong J, Zhang Y. Phinding phish: an evaluation of anti-phishing toolbars. In: NDSS '07: proceedings of the 14th annual network and distributed system security symposium; 2007.

Danchev D. DIY phishing kits introducing new features. ZDNet, May 2008. [online] 〈http://blogs.zdnet.com/security/?p=1104〉. URL: 〈http://blogs.zdnet.com/security/?p=1104〉.

Dhamija R, Tygar JD, Hearst M. Why phishing works. In: CHI '06: proceedings of the SIGCHI conference on Human factors in computing systems. New York, NY, USA: ACM; 2006. p. 581–90.

Fette I, Sadeh N, Tomasic A. Learning to detect phishing emails. In: WWW '07: proceedings of the 16th international conference on world wide web. New York, NY, USA: ACM; 2007. p. 649–56.

Fu AY, Liu W, Deng X. Detecting phishing web pages with visual similarity assessment based on earth mover's distance. IEEE Transactions on Dependable and Secure Computing 2006;3(4):301–11.

Google, Inc. Google safe browsing for Firefox, 2007. [online] 〈http://www.google.com/tools/firefox/safebrowsing/〉. URL: 〈http://www.google.com/tools/firefox/safebrowsing/〉.

Google, Inc. Open communications—Google Talk for developers, 2009. [online] 〈http://code.google.com/apis/talk/open_communications.html〉. URL: 〈http://code.google.com/apis/talk/open_communications.html〉.

Huang C-Y, Ma S-P, Yeh W-L, Lin C-Y, Liu C-T. Mitigate web phishing using site signatures. In: Proceedings of IEEE TENCON 2010; November 2010.

Karlof C, Shankar U, Tygar JD, Wagner D. Dynamic pharming attacks and locked same-origin policies for web browsers. In: CCS '07: proceedings of the 14th ACM conference on computer and communications security. New York, NY, USA: ACM; 2007. p. 58–71.

Levenshtein VI. Binary codes capable of correcting spurious insertions and deletions of ones. Problems of Information Transmission 1965;1(1):8–17.

Mannan M, Oorschot PV. Secure public instant messaging: a survey. In: PST'04: proceedings of the 2nd annual conference on privacy, security and trust; 2004. p. 69–77.

McDonnell S. Bank of Ireland—Statement. Bank of Ireland, September 2006. [online] 〈http://www.bankofireland.com/press_room/latest_releases/2006/press_releases_news_154758_11.html〉. URL: 〈http://www.bankofireland.com/press_room/latest_releases/2006/press_releases_news_154758_11.html〉.

McMillan R. 'Rock Phish' blamed for surge in phishing. InfoWorld, December 2006. [online] 〈http://www.infoworld.com/article/06/12/12/HNrockphish_1.html〉. URL: 〈http://www.infoworld.com/article/06/12/12/HNrockphish_1.html〉.

Microsoft Corporation. Discover windows messenger, 2009a. [online] 〈http://www.microsoft.com/windowsxp/using/windowsmessenger/getstarted/discover.mspx〉. URL: 〈http://www.microsoft.com/windowsxp/using/windowsmessenger/getstarted/discover.mspx〉.

Microsoft Corporation. What is SmartScreen filter? 2009b. [online] 〈http://www.microsoft.com/security/filters/smartscreen.aspx〉. URL: 〈http://www.microsoft.com/security/filters/smartscreen.aspx〉.

Microsoft Corporation. Windows live messenger activity SDK, 2009c. [online] 〈http://msdn.microsoft.com/en-us/library/aa751024.aspx〉. URL: 〈http://msdn.microsoft.com/en-us/library/aa751024.aspx〉.

Mintz M. MSN messenger protocol, 2004. [online] 〈http://www.hypothetic.org/docs/msn/〉. URL: 〈http://www.hypothetic.org/docs/msn/〉.

Mozilla Project. Firefox phishing and malware protection, 2009. [online] 〈http://www.mozilla.com/en-US/firefox/phishing-protection/〉. URL: 〈http://www.mozilla.com/en-US/firefox/phishing-protection/〉.

MSNPiki. MSNPiki: Unofficial MSN protocol documentation, 2007. [online] 〈http://msnpiki.msnfanatic.com/index.php/Main_Page〉. URL: 〈http://msnpiki.msnfanatic.com/index.php/Main_Page〉.

Naor M. Verification of a human in the loop or identification via the turing test, July 1996. [online] 〈http://eprints.kfupm.edu.sa/75319/1/75319.pdf〉.

Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 1970;48(3):443–53.

NetCraft, Ltd. Netcraft anti-phishing toolbar, 2004. [online] 〈http://toolbar.netcraft.com/〉. URL: 〈http://toolbar.netcraft.com/〉.

O'Brien, C. Bank of Ireland to refund phishing victims. ZDNet, September 2006. [online] 〈http://news.zdnet.co.uk/security/0,1000000189,39283133,00.htm〉. URL: 〈http://news.zdnet.co.uk/security/0,1000000189,39283133,00.htm〉.

Parno B, Kuo C, Perrig A. Phoolproof phishing prevention. In: Proceedings of the 10th international conference on financial cryptography and data security; 2006. p. 1–19.

PhishTank. PhishTank: join the fight against phishing, 2009. [online] 〈http://www.phishtank.com/〉. URL: 〈http://www.phishtank.com/〉.

Pidgin Project. What is libpurple? 2009. [online] 〈http://developer.pidgin.im/wiki/WhatIsLibpurple〉. URL: 〈http://developer.pidgin.im/wiki/WhatIsLibpurple〉.

Recordon D, Reed D. Openid 2.0: a platform for user-centric identity management. In: DIM'06: proceedings of the second ACM workshop on digital identity management. New York, NY, USA: ACM; 2006. p. 11–6.

Ross B, Jackson C, Miyake N, Boneh D, Mitchell JC. Stronger password authentication using browser extensions. In: SSYM'05: proceedings of the 14th conference on USENIX security symposium. Berkeley, CA, USA: USENIX Association; 2005.

Saint-Andre P. Extensible messaging and presence protocol (XMPP): core. RFC 3920, October 2004.

Salton G, McGill MJ. Introduction to modern information retrieval. New York, NY, USA: McGraw-Hill, Inc.; 1986.

Secway France. Encryption and security products—crypto for MSN messenger, Yahoo!, ICQ, AIM—secure your IM, 2009. [online] 〈http://www.secway.fr/us/products/all.php〉. URL: 〈http://www.secway.fr/us/products/all.php〉.

Skype Limited. Skype developer docs, 2008. [online] 〈https://developer.skype.com/Docs〉. URL: 〈https://developer.skype.com/Docs〉.

Skype Limited. Skype—security section, 2009. [online] 〈http://www.skype.com/security/security/〉. URL: 〈http://www.skype.com/security/security/〉.

Smith TF, Waterman MS. Identification of common molecular subsequences. Journal of Molecular Biology 1981;147(1):195–7.

StopBadware.org. Badware website clearinghouse, 2009. [online] 〈http://stopbadware.org/home/clearinghouse〉. URL: 〈http://stopbadware.org/home/clearinghouse〉.

Venkatesan R, Koon S-M, Jakubowski M, Moulin P. Robust image hashing. In: Proceedings of the international conference on image processing. IEEE; 2000. p. 664–6.

von Ahn L, Blum M, Hopper NJ, Langford J. CAPTCHA: using hard AI problems for security. In: EUROCRYPT '03: advances in cryptology EUROCRYPT 2003; 2003. p. 294–311.

Wikipedia. OSCAR protocol, 2009. [online] 〈http://en.wikipedia.org/wiki/OSCAR_protocol〉. URL: 〈http://en.wikipedia.org/wiki/OSCAR_protocol〉.

Wu M, Garfinkel S, Miller R. 2004. Secure web authentication with mobile phones. In: DIMACS workshop on usable privacy and security software.

Wu M, Miller RC, Garfinkel SL. Do security toolbars actually prevent phishing attacks? In: CHI '06: proceedings of the SIGCHI conference on human factors in computing systems. New York, NY, USA: ACM; 2006. p. 601–10.

Wu M, Miller RC, Little G. Web wallet: preventing phishing attacks by revealing user intentions. In: SOUPS '06: proceedings of the second symposium on usable privacy and security. New York, NY, USA: ACM; 2006. p. 102–13.

Yahoo! Inc. Yahoo! messenger plug-in SDK, 2008. [online] 〈http://developer.yahoo.com/messenger/〉. URL: 〈http://developer.yahoo.com/messenger/〉.

Yee K-P, Sitaker K. Passpet: convenient password management and phishing protection. In: SOUPS '06: proceedings of the second symposium on usable privacy and security. New York, NY, USA: ACM; 2006. p. 32–43.

Yu D, Chander A, Islam N, Serikov I. JavaScript instrumentation for browser security. In: POPL '07: proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on principles of programming languages. New York, NY, USA: ACM; 2007. p. 237–49.

Zhang Y, Hong JI, Cranor LF. Cantina: a content-based approach to detecting phishing web sites. In: WWW '07: proceedings of the 16th international conference on world wide web. New York, NY, USA: ACM; 2007. p. 639–48.

Please cite this article as: Huang C-Y, et al. Using one-time passwords to prevent password phishing attacks. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.004