Analysis and Improvement of Anti-Phishing Schemes

Dinei Florêncio and Cormac Herley

Microsoft Research, One Microsoft Way, Redmond, WA

Abstract. The problem of phishing has attracted considerable attention recently, and a number of solutions and enhanced security measures have been proposed. We perform a detailed analysis of several anti-phishing schemes, and attacks and improvements. While several anti-phishing technologies address commonly observed phishing tactics, the space evolves rapidly, and a good prevention technique should be robust to anticipated as well as observed attacks. We present a number of attacks and techniques that might be easily employed by phishers and examine the robustness of a recently proposed password re-use anti-phishing system. We compare with other proposed phishing prevention techniques and find that it withstands several attacks that render current anti-phishing approaches obsolete and fares better in a large scale deployment than others.

1 Introduction

The problem of phishing has been well documented in the popular press. A phisher who wishes to attack, say BigBank.com, spams many users with an email purporting to come from Bigbank. The email says that there is a problem with their BigBank account, and directs them to a website to login and fix the problem. The email and the phishing website look as though they belong to BigBank, but in fact have no affiliation. Users who "login" are parting with their BigBank identity, which can be harvested by the phisher. See [13] for details of recent attacks and statistics on the enormous growth in the number of attacks since the phenomenon first emerged.

The problem differs from many other security problems in that we wish to protect users from themselves. The difficulty of finding a solution is compounded by a number of issues. Both false positives (where we falsely suspect a phishing attack) and false negatives (where we fail to detect one) are very expensive. False positives erode trust in the system and cause inconvenience and possible loss to websites that are erroneously classified as phishing. False negatives allow a phisher to grab a user's credentials in spite of our efforts. Recent approaches to phishing prevention have struggled with the effort to keep both kinds of error rates low.

A further difficulty is that of warning the user (or taking other action when phishing is detected or suspected). Halting the browser connection (*i.e.* refusing to connect to the site) is usually not acceptable unless it is absolutely certain

that the site is phishing. Warnings and pop-ups have been found to be of very questionable use in getting users to alter their behavior [15].

We explore some of the proposed solutions to this problem and examine attacks and improvements. An interesting recently proposed scheme [7] claims to address all of these problems, and stop essentially all phishing attacks. We present some tactics that phishers might employ against this system and explore it's robustness. We also document the strength and weaknesses of other phishing prevention technologies. The next section covers related work. Section 3 gives a brief overview of the Password Re-Use scheme. Section 4 analyzes the attacks on many of the anti-phishing solutions and the PRU scheme. Section 5 concludes.

2 Related Work: Existing Anti-Phishing Approaches

Broadly speaking, solutions divide into those that attempt to filter or verify email, password management systems, and browser solutions that attempt to filter or verify websites.

2.1 Password Management Systems

Since phishing primarily targets the user's password several proposed approaches address the problem by using a password management system. Such systems enable users to manage all of their passwords from a single secure location. An early example proposed by Gaber *et al.* [8] used a master password when a browser session was initiated to access a web proxy, and unique domain-specific passwords were used for other web sites. These were created by hashing whatever password the user typed using the target domain name as salt. Similarly, Microsoft's Passport [1] allows users to sign in to the Passport site, which remembers their credentials and authenticates them at other sites that participate in the Passport program. Neither of these systems aimed to address phishing directly.

Ross *et al.* [14] propose a phishing solution that, like [8], uses domain-specific passwords for web sites. A browser plug-in hashes the password salted with the domain name of the requesting site. Thus a phisher who lures a user into typing her BigBank password into the PhishBank site will get a hash of the password salted with the PhishBank domain. This, of course, cannot be used to login to BigBank, unless the phisher first inverts the hash. Their system has no need to store any passwords.

Halderman *et al.* [9] also propose a system to manage a user's passwords. Passwords both for web sites and other applications on the user's computer are protected. In contrast to [14] the user's passwords are stored, in hashed form on the local machine. To avoid the risk of a hacker mounting a brute force attack on the passwords a slow hash [12] is employed.

2.2 Browser Plug-ins

A number of browser plug-in approaches attempt to identify suspected phishing pages and alert the user. Chou *et al.* [5] present a plug-in that identifies many of the known tricks that phishers use to make a page resemble that of a legitimate site. For example numeric IP addresses or web-pages that have many outbound links (*e.g.* a PhishBank site having many links to the BigBank site) are techniques that phishers have used frequently. In addition they perform a check on outgoing passwords, to see if a previously used password is going to a suspect site. Earthlink's Scamblocker [2] toolbar maintains a blacklist and alerts users when they visit known phishing sites; however this requires an accurate and dynamic blacklist. Spoofstick [3] attempts to alert users when the sites they visit might appear to belong to trusted domains, but do not. Trustbar [10] by Herzberg and Gbara is a plug-in for FireFox that reserves real estate on the browser to authenticate both the site visited and the certificate authority.

Dhamija and Tygar [6] propose a method that enables a web server to authenticate itself, in a way that is easy for users to verify and hard for attackers to spoof. The scheme requires reserved real estate on the browser dedicated to userid and password entry. In addition each user has a unique image which is independently calculated both on the client and the server, allowing mutual authentication. A commercial scheme based on what appear to be similar ideas is deployed by Passmark Security [4]. The main disadvantage of these approaches is that sites that are potentially phishing targets must alter their site design; in addition users must be educated to change their behavior and be alert for any mismatch between the two images.

3 Review of the Password Re-Use Scheme [7]

3.1 Client Piece: Browser Plugin

The Password Re-Use (PRU) architecture involves a plug-in running in the user's browser that detects when credentials from a *protected list* are entered in the browser, and a server that receives an alert when this has occurred, and also stores the credentials. The credentials are (*uid*, *pwd*, *dom*), where *uid*, *pwd* and *dom* are the userid, password and domain respectively. When the user enters *uid*, *pwd* at any site dom' where $dom' \neq dom$ and dom' is not on the *whitelist*, that fact is reported to the server by sending (`hash(`*uid*`)`, *dom*, dom'). The server is in a position to aggregate the reports from many users, and distinguish a phishing attack from the reports that we expect due to the fact that users commonly use a few passwords at all the sites they visit.

The credentials are extracted from the data POST'ed for any page that contains a password. A separate thread tracks the user's keystrokes to detect when a previously POST'ed password is retyped at an unfamiliar site. Whenever a hit is generated, the server is informed. What is reported to the server is the fact that a password from dom_1, on the protected list, was typed at dom_R, which is not on a pre-populated whitelist. Users commonly use the same password at

several legitimate sites. This is not a problem, since only an accumulation of reports will generate suspicion at the server.

3.2 The Server Piece

The server's role is that of aggregating information from many users. A Password Re-Use report is in itself of course, not proof that an attack is in progress. It means either that the user is knowingly using the same password at a protected and a non-whitelisted site, or that she has been fooled into entering the data by a phishing site. Thus it is important to distinguish innocent cases of password re-use from phishing attacks. This task is much simplified by the fact that the server aggregates the data across many users. Having a single user typing her BigBank credentials into an unknown site may not be alarming, but having several users type BigBank credentials into *the same* unknown site within a short period of time is definitely alarming.

One of our goals is to examine how robust this scheme is to evolving attacks, which we examine in the next section.

Notifying the Target A key difference between the PRU scheme and the anti-phishing browser plug-ins [5, 2, 3] is the action taken when a site is suspected of phishing. When a threshold number of users report using their dom_R password at another site *dom*, the server determines that an attack is in progress. Rather than inform the user, at this point the server notifies the institution under attack. There are two very important components to the information the server can now provide dom_R :

- The attacking domain *dom*
- The hashes h(*uid*) of already phished victims.

First, by getting the attacking domain to the institution under attack they can proceed with "Cease and Desist" orders or web-site takedown (a process that can involve legal as well as Denial of Service measures). Next, by providing the institution with h(*uid*) for phishing victims, they can disable withdrawal facilities on the accounts in question.

Since additional victims are likely to be phished in the interval between the server notifying dom_R and the phishing site *dom* actually going offline. Each of these victims will generate a report to the server, and thus h(*uid*) for each victim can be routed to dom_R without delay. Thus, even though the scheme provides no notification to users, their accounts are protected.

The mechanism for notifying dom_R that it is under attack is simple. A domain BigBank that wishes to be protected under this scheme sets up an email account `phishreports@bigbank.com`. Reports will be sent to that address. For verification purposes the email header will contain the time, the domain (*i.e.* "Bigbank") and the time and domain signed with the server's private key. In this manner any spam or other email that does not come from the server can be immediately discarded. Scale of deployment is a key factor here. It is unlikely

that BigBank would set up special procedures to save just a few users. However, if the proposed scheme were deployed to a majority of users it is more likely that financial institutions would take the steps to use this information to protect their users. In other words scale works in favor of the scheme, instead of against it.

4 Attacks on Anti-Phishing Schemes

Anti-phishing solutions must observe both security and usability constraints. A peculiarity of the space is that for many solutions the degree of protection it offers is related to the scale of deployment. If the installed base for a particular solution is small enough, it is very likely that phishers will simply ignore those users and target only the bulk of users, i.e., users not using that technology. However, as the installed base grows, phishers will no longer ignore those users. Thus, the efficacy of some technologies decreases with the scale of deployment. We will try to illustrate this in our enumeration of attacks that follows.

A rather obvious objection that is common to the client plug-ins [5, 3, 2], the PRU scheme [7], and password management schemes [14] is that they require the user to instal software or change their behavior. Phishing, as opposed to pharming or keylogging, probably finds most of its victims among the less technically savvy. Once a user understands what phishing is, and how it works she is no longer at great risk. The paradox is that those who know enough to download a plug-in probably no longer need the plug-in. Nonetheless these technologies are worth examining in detail, since those that are effective may become incorporated in the popular browsers.

4.1 Attacks on Anti-Phishing Client Plug-ins

A sample of the client plug-ins that offer protection from phishing schemes are [3, 2, 5].

Broken Links and Delaying Pageload Attack Client plug-ins that perform some test to determine whether a site is phishing or not are amenable to an attack that delays completion of the pageload. For example, Internet Explorer provides and event handler `DocumentComplete`, which might seem the perfect place to introduce the test. Spoofguard [5], for example, performs several checks only when this event occurs. However, a phisher then need merely place one broken link on the page to prevent this event from ever occurring. This prevents the plug-in from even executing the phishing test. To avoid this, the plug-in might execute the test a fixed amount of time after the URL was first contacted. This is better, but again open to attack: if the plug-in waits, say 10 seconds, before performing the test, the phishing site need merely delay any items that are likely to cause suspicion until after the test is performed. For example, the part of the page that collects a password can be withheld until after the test is performed.

Thus, quite apart from the question of *what* a phishing detection plug-in should check, there is a considerable question of *when* those checks should be performed. An improvement might be to execute any checks when the user types the first keystroke on the page. This avoids wastefully executing the phishing detection when a user is navigating to sites that involve no typing (which is likely a majority of pages for most users), while allowing the check to be performed before the password is typed (assuming that the phishing page asks for the userid first).

Problems with Blacklist Approaches Blacklisting approaches attempt to inform clients of phishing sites by either pushing an updated list to the client plug-ins [2] or having the clients check with a server to request information on a URL it is visiting. Both of these approaches have difficulties. If a blacklist server broadcasts updated lists of phishing sites there is a definite latency issue. Even if the broadcasts occur once a day a phisher can have a full 24 hours to harvest passwords without fear of discovery. Greater frequency increases the load on the server; this might be feasible if the plug-in is used by a small percent of the population but becomes unwieldy with scale. If each client contacts a blacklist server for information on each URL the latency issue is moot, but the scale problems for the server are even worse.

Problems with Whitelist Approaches In approaches where the client contacts a blacklist server for each url, whitelists (i.e., a pre-compiled list of safe sites) can be used to reduce the traffic to the server. More specifically, if a user visits a site that is in the whitelist, there is no need to contact the blacklist server. Nevertheless, updating white-lists in the client is cumbersome and costly, and white-listing is subject to cross-site scripting, hacking, and personal sites located on large domains. For this reason, whitelisting can only be used in conjunction with sites that have high security standards, and do not host personal pages.

Redirection and Distributed Attacks The distributed phishing attack described in [11] poses an interesting challenge for all phishing protection systems. In this attack the phisher hosts the phishing page at many different sites; in the limiting case each victim might be directed to a page unique to them. If a client plug-in makes use only of information at the client (*e.g.* using rules based on the URL and html) then distributing the site among many pages makes no difference (since the individual clients are not sharing information). Of course protection systems like this are the most easily defeated: rules are easily evaded and these solutions suffer the problems of scale.

For plug-ins where individual clients make use of external information, for example by getting periodic blacklist updates, a distributed attack can destroy the value of the blacklist. Even if blacklists are updated very frequently the phisher can ensure that no victim is ever directed to a site on the blacklist, but

rather is sent to the next in a long series of host addresses. A suitably organize botnet can provide the phisher with more than enough hosting sites.

Problem of getting users to alter their behavior A very interesting recent study by Wu [15] confirms that users either tend to ignore or fail to act on security warnings. The study attempted to measure whether users notice the warnings provided by anti-phishing toolbars [3, 2] *even when the toolbar correctly detected the attack*. A large percentage of the participants did not change their behavior based on the warning. This point threatens the underpinning of several anti-phishing solutions. It appears that identifying a site as being a phishing site is not sufficient.

A clear advantage of the PRU scheme [7] is that it does not assume that users alter their behavior based on warnings.

The Problem of Scale Like many security technologies client-side detection systems face the problem of scale. Simple techniques that may work well when employed by only a small portion of the population. For example, each of [5, 3] filter for behaviors that are known to be in common use by phishers. Consider a plug-in phishing detector that is used by only 1% of the population. At that level of penetration phishers are unlikely to go to much effort to avoid the filters, so the 1% may indeed get worthwhile protection. Now consider the same plug-in used by 90% of the population. At this level of deployment phishers will be keenly motivated to evade the filters. In other words there are many rules or heuristics that can appear promising when run on training data. They can even perform successfully and offer worthwhile protection to small fractions of the overall population. However their efficacy is in inverse relation to their scale of deployment: the more people use them the less effective they are. An advantage claimed in [7] is that the efficacy grows rather than decreases with the scale of deployement.

4.2 Attacks on Password Re-Use Anti-Phishing Scheme

The simplicity of the PRU client means that it is relatively hard to prevent it from reporting password re-use events. Nonetheless there are a few approaches that an attacker might take to trick the client software into failing to report. The next four attacks we consider are of this form.

Flushing the protected list A Phisher might try to circumvent the protection by removing some (or all) of the passwords from the protected list. For example, since the protected list has only 256 entries, a phishing site could submit (using HTML forms) 256 strings as passwords to a random site. This would effectively "flush" everything from the protected list because of the Least Recently Used maintenance rule. To avoid this attack, before accepting a new entry, from the HTML form data, the password is matched with the contents of a keyboard

buffer, effectively requiring that the password have been actually typed at the site. It is unlikely that a Phisher can induce a victim to actually type hundreds of password-like strings.

Hosting on a whitelisted domain A phisher might attempt to host on an existing, legitimate site. For example putting the phishing page up on an ISP member site, like members sites on AOL or MSN, or a small site like a community group association, or by employing a Cross-Site Scripting (CSS) attack. Each of these is handled by proper design of the client whitelist.

It is easy to handle ISP member sites by including the ISP, but excluding certain sub-domains from the whitelist. Small groups like community associations cannot be depended upon to prevent break-ins. Thus the client whitelist should contain only very large commercial domains. Recall that a site can be on a users protected list, without being on the whitelist. CSS attacks actually host the phishers page on a target domain. For this reason, only sites that can be depended upon to maintain basic levels of security should be permitted to populate the client's whitelist.

Tricking the user into mis-typing the password An earlier version of the algorithm hashed the combined *uid/pwd*. This provided a way for a Phisher to circunvent the system, by forcing the user to mis-type the *uid*. Normally, as you type your userid, the letters show up at the screen. Suppose the Phisher introduces a page with a script where the third character you type do not show up in the screen. You'd think you did not pressed hard enough, and would re-type the character. As you do that, the keyboard buffer will have that character twice, and it will not hash to the protected entry. We note that this attack is not possible with the password, since the password do not show up in the screen as you type (only ****). If something goes wrong, most users simply delete the whole thing and re-start.

Visual Keyboard The PRU scheme only detects passwords type at the browser. Any scheme that obtains the password by other means than typing at the browser would be undetected. For example, a phisher may convince the victim to use a screen keyboard and click on each of the letters corresponding to the password. Or may convince the victim to type the password first on notepad, then copy and paste on the browser window. Or get the Phone and call an 1-800 number. Any of these methods would not trigger the defense. Nevertheless, we believe the further away the attack get from what the user expects, the less likely is the phisher to succeed.

Distributed Attack A possible approach for a phisher who seeks to evade detection is to distribute the attack. Rather than phish BigBank by directing victims to PhishBank the phisher may use many domains, or numeric IP addresses. Thus when clients report sending their BigBank credentials, it fails to

trigger an alert at the server. For this reason, we believe the server logic needs to adapt as phishers adapt. For example, while the destination for several BigBank credentials may be distributed, a large increase in the number of reports for a given whitelisted domain is in itself worthy of suspicion.

Redirection Attack Similar to the distributed attack is a Redirection attack where a phisher directs all victims to a single URL, but each victim is redirected to a different (possibly unique) address to be phished. For example the phishing URL might originally redirect to IP_1, but as soon as the first victim is caught it redirects to IP_2 and so on. This might appear to confuse the system by distributing the client reports one at a time among many addresses. To circumvent this the client report to the server includes any URLs visited in the last minute. By intersection, the redirecting (phishing) URL can be detected.

Also, we point out that many competing anti-phishing schemes would be highly susceptible to this kind of attack. In particular, any scheme that relies on blocking visualization of a page that has been detected as phishing would be susceptible to this attack. Similarly, this applies to any technology that relies on users reporting suspicious pages. Of course, the solution used in the PRU scheme can be easily adapted to work with these other anti-phishing schemes as well.

Looking at the Phishing html? Including a site on the block list should be taken only when no doubt remains about a site being a phishing site. For legal and liability reasons it may even be necessary to have a person actually look at a site before making a final determination on block list inclusion. If that's the case, rather than visit the suspect site it is better to receive directly from the reporting client the HTML content that it received. The reason for this is that a phisher might easily present each phishing victim with a unique url, so that the first visitor to the url (the victim) would see the page requesting userid and password, while the second and subsequent visitors (*e.g.* someone at the server checking the site) would see a completely innocent page. This is easily accommodated by adding the HTML as a record to C_{report}.

4.3 Denial of Service on a Site

We now explore the possibility that a vandal tries to fool the PRU system that a legitimate site is phishing. Specifically, suppose a disgruntled MyCornerStore customer types BigBank credentials at the MyCornerStore login site. Recall, the vandal need not use real BigBank credentials, since populating the protected list was very easy. What are the consequences if the server wrongly concludes that MyCornerStore is phishing and places it on the block list?

First, note that the system tracks the IP of the reporting client; a single client reporting the site repeatedly will have no effect. Second, no site that is on the client whitelist (of large institutions) or the server's much larger whitelist can ever be placed on the block list. The most powerful defense, however, is that if MyCornerStore does get on the block list, *it does not mean that nobody*

can navigate to that site. Rather, it means that BigBank customers those who login to MyCornerStore for the first time (*i.e.* it is not on their protected list) and use a password already used at at BigBank would have their h(*uid*) sent to BigBank for monitoring. Users who have existing MyCornerStore credentials in their protected list would be unaffected (since the client will not even consult the server). BigBank users logging in for the first time who choose passwords not on their protected list would be unaffected. So, the level of effort required to include a site in the block list seem significantly high for causing a reasonably small inconvenience to users of the site.

4.4 Attacks on other approaches

Two Problems with pwdHash [14] A difficulty, however, is that it is by no means simple to know when a password is being typed (*e.g.* phishers may use Javascript rather than HTML forms to gather passwords). To circumvent this problem [14] requires that users prefix all password by some special sequence or key (*e.g.* type 'F8' before any password). Thus, users must be "trained" to adopt the proposed system, and change their behavior. This might be an acceptable solution, but represents a non-trivial change in current practice.

A further difficulty with domain specific passwords is that different web-sites have very different password rules. Some sites require strong passwords, some accept only alphanumeric passwords, and some accept only numeric passwords. Many banks, for example, require that a user's online password be the same as the PIN they use to access their ATM. This creates the difficulty that the domain specific passwords generated will be unacceptable to the target web-site. To address this problem [14] suggests tabulating the password rules that different institutions accept. This is clearly not a very scalable solution.

Attack on Mutual Authentication Scheme [6] A main weakness with the image based mutual authentication scheme is that it represents an unfamiliar use model, and thus is probably more susceptible to social engineering attack. If a bank chooses to authenticate itself to users by presenting a user-chosen image it adds a level of security. It remains open to question whether users notice *an act on* the absence of the authenticating image. But further the problem of phishing derives from the ease with with users are manipulated into parting with sensitive information. A phishing attack on a server using image authentication would involve sending users an email saying there is a problem with their authenticating image and asking to login to fix it.

5 Conclusion

We have examined attacks and improvements on a number of anti-phishing technologies. We found that many of the client plug-in approaches that have been proposed have efficacy that decreases as the scale of the deployment increases. By contrast, we reviewed a powerful new method for detecting phishing attacks

and protecting users has been recently proposed in [7]. That method relies on two independent ideas: aggregating information at the server for detecting an attack, and back channel protection for saving users. The method grows strong with the more users using the system. We examined its ability to withstand various anticipated attacks as compared with other anti-phishing technologies. We find that while several anti-phishing approaches have sever difficulties coping with possible phishing tactics the PRU scheme exhibits great robustness.

References

1. `http://www.passport.com`.
2. `http://www.scamblocker.com`.
3. `http://www.spoofstick.com`.
4. `http://www.passmarksecurity.com`.
5. N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. Mitchell. Client-side defense against web-based identity theft. *Proc. NDSS*, 2004.
6. R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. *Symp. on Usable Privacy and Security*, 2005.
7. Dinei Florêncio and Cormac Herley. Stopping Phishing Attacks Even When the Victims Ignore Warnings. *MSR Tech. Report.* `http://research.microsoft.com/users/cormac/papers/NoPhish.pdf`.
8. E. Gaber, P. Gibbons, Y. Matyas, and A. Mayer. How to make personalized web browsing simple, secure and anonymous. *Proc. Finan. Crypto '97.*
9. J. A. Halderman, B. Waters, and E. Felten. A convenient method for securely managing passwords. *Proceedings of the 14th International World Wide Web Conference (WWW 2005).*
10. A. Herzberg and A. Gbara. Trustbar: Protecting (even naive) web users from spoofing and phishing attacks. 2004. `http://eprint.iacr.org/2004/155.pdf`.
11. M. Jakobssen and A. Young. Distributed phishing attacks. 2005. `http://eprint.iacr.org/2005/091.pdf`.
12. J. Kelsey, B. Schneier, C. Hall, and D. Wagner. Secure applications of low-entropy keys. *Lecture Notes in Computer Science*, 1396:121-134, 1998.
13. Anti-Phishing Working Group. http://www.antiphishing.org.
14. B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. *Proceedings of the 14th Usenix Security Symposium, 2005.*
15. M. Wu. Users are not dependable: How to make security indicators to better protect them. *Trustworthy Interfaces for Passwords and Personal Information*, 2005.