# PHISHPIN: AN INTEGRATED, IDENTITY-BASED ANTI-PHISHING APPROACH

Hicham Tout
*School of Computer and Information Sciences, Nova Southeastern University*
*3301 College Avenue, Fort Lauderdale, Florida, U.S.A.*

Keywords: Phishing, Spam, Information security, Identity theft, Social engineering, Encryption, Hash algorithms, One time password, Digital certificates, Online scams, Web, Pharming.

Abstract: Phishing is a social engineering technique used to fraudulently acquire sensitive information from users by masquerading as a legitimate entity. One of the primary goals of phishing is to illegally carry fraudulent financial transactions on behalf of users. The two primary vulnerabilities exploited by phishers are: Inability of non-technical/unsophisticated users to always identify spoofed emails or Web sites; and the relative ease with which phishers masquerade as legitimate Web sites. This paper presents Phishpin, an approach that leverages the concepts of mutual authentication to require online entities to prove their identities. To this end, Phishpin builds on One-Time-Password, DNS, partial credentials sharing, & client filtering to prevent phishers from masquerading as legitimate online entities.

## 1 INTRODUCTION

Phishing is a social engineering technique used to fraudulently acquire sensitive information from users by masquerading as a legitimate entity. Phishing is typically carried over electronic communications such as email or instant messaging (Kirda and Kruegel, 2006). One of the primary goals of phishing is to illegally acquire sensitive information, such as passwords or social security numbers, in order to carry fraudulent transactions on behalf of the victim. Using a forged email that contains a URL pointing to a fake Web site—masquerading as an online bank or a government entity, a phisher may lure a victim into giving his/her Social Security Number, full name, & address, which can then be used to apply for a credit card on the victim's behalf. According to McCall (2007), phishing attacks escalated in the 12 month period ending August 2007 to impact 3.6 million adults and cause losses worth approximately $3.2 billion.

The first publicized phishing attacks occurred at AOL in the early 90s where phishers posed as AOL staff members to lure victims into giving their sensitive account information (Wikipedia, 2008). Since then the number of phishing attacks have substantially increased. Attacks have also evolved to become more sophisticated and malicious targeting large number of users dealing with financial institutions. Much effort has gone into the development of anti-phishing techniques. These techniques fall into 4 categories: content filtering, blacklisting, symptom-based prevention, & domain binding. Many of these techniques focus on enabling clients to recognize & filter various types of phishing attacks. While many of these techniques have proven effective in a number of scenarios, they also have put much of the burden of proof on either the online user or client filter or both. Yet online entities have had few techniques that enable them to prove their identities without forcing online users to deploy complex bi-directional authentication mechanisms.

This paper presents Phishpin, an anti-phishing technique that integrates One-Time-Password (OTP), DNS, partial credentials sharing, & client filtering techniques to prevent phishers from easily masquerading as legitimate online entities. One of the primary goals of the proposed approach is to enable both parties—online users and entities—to prove their identities—mutual authentication—without having to divulge sensitive information. Another goal of the proposed solution is to build an effective, yet simple mutual authentication mechanism that runs seamlessly within the client

Tout H. (2009).
PHISHPIN: AN INTEGRATED, IDENTITY-BASED ANTI-PHISHING APPROACH.
In *Proceedings of the International Conference on Security and Cryptography*, pages 369-374
DOI: 10.5220/0002222503690374
Copyright © SciTePress

browser. The proposed approach is made up of 3 components: DNS TXT record to store the legitimate entity's certificate; one-way hash algorithm; and client/server plug-in to verify the authenticity of both online entities and users.

This paper is organized as follows. Section 2 introduces related work and contrasts it with the approach in this paper. Section 3 discusses the proposed approach. Finally, conclusions and future work are summarized in section 4.

## 2 RELATED WORK

This section enumerates some of the known anti-phishing techniques. It's meant to provide a brief overview of some of the best known efforts in this area of research. Anti-phishing techniques fall into 4 major categories: content filtering, blacklisting, symptom-based prevention, & domain binding. Content/email filtering relies on machine learning methods, such as Bayesian Additive Regression Trees (BART) or Support Vector Machines (SVM), to predict and filter phishing emails (Abu-Nimeh et al., 2007; Fette, Sadeh, and Tomasic, 2006). Since email is normally the first step in a phishing attack, the advantage of this technique is that it intercepts & eliminates suspected phishing emails before they reach the user. Contents of the email, the sender/source, and other attributes are analyzed by this technique. The main disadvantage of this technique is that it cannot guarantee that all phishing emails are filtered (Wu, Miller, and Little, 2006). Phishers have come up with alternative semantics that are capable of bypassing these filters. Phishers have also in certain cases resorted to the use of images instead of text, which makes the filtering process more challenging. It's important to note that while the majority of phishing attacks are initiated by email, there has been a surge of new types of attacks that are initiated by instant messaging or by hacked Web pages. These types of attacks cannot be intercepted by email-based solutions.

Blacklisting depends on public lists of known phishing Web sites/addresses published by trusted entities such as (Phishtank, 2008). It requires both a client & a server component. The client component is implemented as either an email or browser plug-in. that interacts with a server component, which in this case is a public Web site that provides a list of known phishing sites. In the case of an anti-phishing email plug-in, the client component compares URLs embedded in every incoming email to one or more publicly provided lists of suspected phishing sites. Should it find a match, the email is either discarded or flagged as a phishing/spam email. In the case of an anti-phishing browser plug-in, the client component compares every URL loaded into the address field of the browser to one or more publicly provided lists of suspected phishing sites. Should it find a match, a warning message, in the form of a popup window is displayed. The advantage of this technique is the ability of the plug-in to reference a frequently updated, reliable public list of known phishing Web sites. This technique however, suffers from many of the same problems as signature-based prevention methods—almost always outdated as phishers continuously use new Web sites and addresses. In fact most phishing Web sites are only available online for few hours (Zhang, Hong, and Cranor, 2007). It's important to note that blacklisting have, in certain cases also been used as a component/step in email filtering solutions since it runs as an email plug-in in most cases.

Symptom-based prevention analyses the content of each Web page the user visits and generates phishing alerts according to the type and number of symptoms detected (Chou et al., 2004). Symptoms generated are the result of parsing the contents—text of the Web page and the URL/address. Symptom-based prevention uses learning and identification techniques similar to email filtering such as SVM and BART. The difference between the two techniques is that one operates on the contents of the email, while the other operates on the content of the Web page being visited. It's important to note that unlike email-based filtering, both symptom-based & blacklisting techniques are not invoked until after the user presses Web link contained in the email. The advantage of this technique is that it parses the content of the visited Web site using machine learning techniques to conclude whether it's a phishing site. This technique may provide a higher level of detection rate since it parses the content of the actual visited site and not just the text in the phishing email. Disadvantages of this technique include its inability to detect phishing attacks that use client-side JavaScript and its reliance on warning messages, which have proven ineffective with most users (Wu, 2006). It's important to note that this technique should be viewed as complementary to rather than competing with email-based phishing techniques. The combination of both methods may enable a defence in depth strategy.

Trusted domain binding is a browser-based technique that binds sensitive information—mostly credentials—to a specific domain (Raffetseder, Kirda, and Kruegel, 2007). Should the user enter sensitive information in a Web form that belongs to a different/un-trusted Web site, the browser plug-in either blocks the transmission of data or warns the user of the consequences. This technique runs as part of the Web browser workflow, disables all Web form fields identified as sensitive, and presents the user with a specialized form where credentials are entered. This technique establishes a one-to-one relationship between a set of credentials and a trusted domain. Should those credentials be used with a different domain that is not yet considered trusted, submission of information is blocked and an alert is sent to the user. The design principles behind this technique rely on the assumption that preventing the user from directly submitting sensitive information can eliminate most if not all phishing attacks. The approach builds on top of a survey conducted by (Wu, 2006), which concluded that the use of only warning messages did little to sway users from proceeding forward with what they perceived to be a trusted site. This approach provides a high detection rate and does not suffer from many of the disadvantages associated with email filtering or blacklisting. However this technique requires a manual process to identify & bind sensitive information and to identify trusted domains for initial binding. It also uses a domain-based binding process (Wu, Miller, and Little, 2006) and requires a one-to-one relationship between credentials and the intended domain. In addition, the methods used by this technique to distinguish between trusted and non-trusted domains/Web sites is similar to the one used by blacklisting.

In contrast, Phishpin combines client/server-based filtering and domain-based identity techniques to prevent phishers from masquerading as legitimate online entities. It integrates PKI, DNS, OTP, & filtering to enforce the authenticity of online entities based on primary attributes associated with both legitimate online entities and online users. One such attribute is the legitimate entity logo or an imitation of it, which is used by phishers as a visual deception tactic intended to trick end-users into believing they're connected to a legitimate online entity (Downs, Holbrook, and Cranor, 2007). Other attributes may include online user's name, address, phone, password, social security number, or pin number.

## 3 PHISHPIN

While users are required to use one or more authentication methods to prove their identities, online entities have done little to prove they are who they claim they are. With little Web development or design experience, a phisher can masquerade as almost any legitimate online entity. The ease with which online entities can be spoofed may be considered one of the primary vulnerabilities in the fight against phishing attacks. One of the primary goals of the proposed approach is to address this vulnerability by enabling both parties—online users and entities—to prove their identities without having to divulge sensitive information. Another goal of the proposed solution is to build an effective, yet simple mutual authentication mechanism that runs seamlessly within the client browser.

Phishpin is divided into three major components: DNS, client plug-in, and server plug-in. The DNS TXT record is used to store the online entity's certificate. The primary purpose of storing the cert is to enable the client plug-in to validate the certificate chain and match the distinguished name in the certificate with the domain name being visited. This step ensures that not only the certificate chains to a known and trusted certificate authority but also that it's associate with the domain being visited. This step is considered as the first line of defence against phishing attacks. It mostly focuses on the legitimacy of the credential being presented by the online entity. However, well known DNS attacks that may spoof legitimate certificates, have been documented. Therefore it's important to note that this must not be the only line of defence used. It's also worthwhile noting that such validation must be executed seamlessly by the browser plug-in without must input from the online user. Should certificate chain validation or distinguished name matching fail, the client plug-in would lock the Web form/page and inform the online user that authentication of the site has failed. Displaying warnings without disabling Web forms has shown to be ineffective as most online users tend to ignore warning messages.

The role of the client plug-in is to block phishing attacks by validating the online entity's certificate stored in the DNS TXT record; applying content filtering rules; validating 2nd half of the OTP received from the server; and generating the 1st half of OTP. On the other hand, the role of the server plug-in is to validate the 1st half of the OTP received from the client and generating the 2nd half of the OTP. This is in addition to using the selected hash algorithm to build the initial hash based on user attributes stored during the account setup process.

The DNS component includes the DNS record to store the online entity's certificate. The certificate is stored in the TXT record.

The server plug-in includes the following:

• An algorithm to generate OTP.
• A method to validate 2nd half of OTP received by the client.
• A method to generate 1st half of OTP.
The client filter includes the following:
• An algorithm to generate OTP.
• A method to validate 1st half of OTP received by server.
• A method to generate 2nd half of OTP.
• A method to disable Web form fields.
• A Method to validate online entity's certificate.

The initial account setup process typically requires online users to enter general and personal information. This information may include user id, account number, user name, address, phone, password, and potentially social security number, which is mostly used for financial accounts setup. In addition to the typical account setup, the proposed approach would require both online users & entities to determine the hash algorithm and the sequence of attributes to be used in the OTP hash. While MD5 has been amongst the most commonly used hash algorithms, successful attacks against MD5 have been documented. Therefore it's recommended that stronger hash algorithms such as SHA-256 be considered. As illustrated in figure 1 below, besides enabling an online user to input account or personal information, the initial account setup process would also include the following steps:

• Register selected hash algorithm with both client filter and server plug-in.

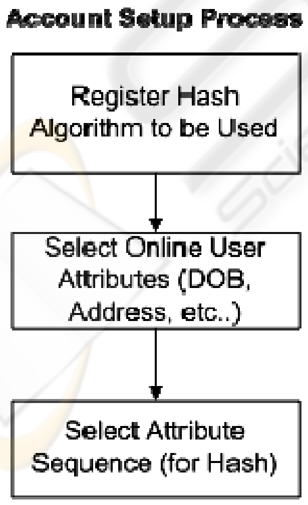
Account Setup Process
Register Hash Algorithm to be Used
Select Online User Attributes (DOB, Address, etc..)
Select Attribute Sequence (for Hash)

Figure 1: Initial Account Setup.

• Define online user attributes that will be used as part of the OTP. It is recommended that at least one or more private attributes are included in order to ensure that that the resulting hash is built based on multiple unknown strings that contains half of the password hash & challenge phrase.

- Define the order in which attributes are concatenated into the source string for the hash function.
- Compute the hash of the online user's password; divide it in half; store the second half of the hash in a secure repository—entitlement store—at the online entity site; store the first half of the hash on the online user's device. The second half of the hash must be accessible by the server plug-in while the first half of the hash must be accessible by the browser plug-in. Since this step is performed during account setup, which also includes setting up the password, access to the user password would be made available by the setup script or application being used to create the initial account.

As illustrated in figure 2 below, the client plug-in, which in this case is a browser plug-in, performs the following functions:

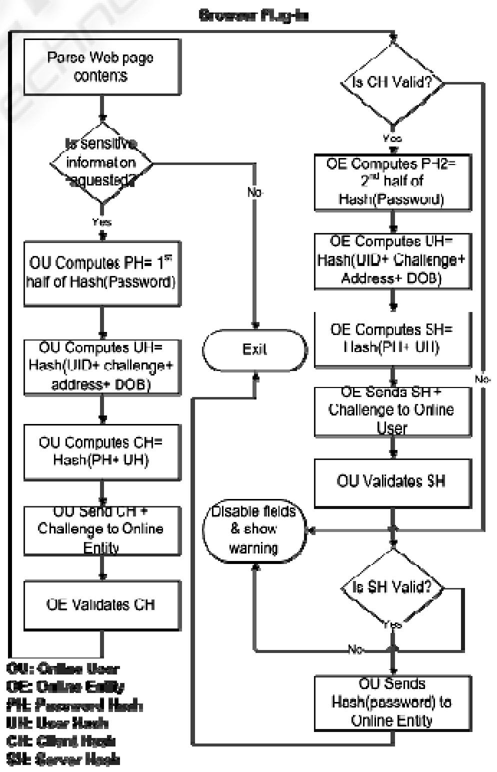
Browser Plug-in
Parse Web page contents
Is sensitive information requested?
Yes
No
OU Computes PH= 1st half of Hash(Password)
OU Computes UH= Hash(UID+ challenge+ address+ DOB)
OU Computes CH= Hash(PH+ UH)
OU Send CH + Challenge to Online Entity
OE Validates CH
Exit
Disable fields & show warning
Is CH Valid?
Yes
No
OE Computes PH2= 2nd half of Hash(Password)
OE Computes UH= Hash(UID+ Challenge+ Address+ DOB)
OE Computes SH= Hash(PH+ UH)
OE Sends SH + Challenge to Online User
OU Validates SH
Is SH Valid?
Yes
No
OU Sends Hash(password) to Online Entity
OU: Online User
OE: Online Entity
PH: Password Hash
UH: User Hash
CH: Client Hash
SH: Server Hash

Figure 2: Browser Plug-in.

- Parse Web forms embedded in the loaded Web page for fields that require the user to enter sensitive information such as user id, password, credit card number, or social security number. Should one of these form fields be detected, a certificate chain validation in performed on the certificate stored in the DNS TXT record. In addition the DN in the certificate is compared to the domain being connected to. Should validation fail, all form fields are locked by the browser plug-in using Java scripting and a warning is displayed to alert the online user of a possible phishing attack. Should the Web form include a Java Applet or a ActiveX control, the browser plug-in would intercept the data/stream submitted by the Applet/ActiveX control to check for the potential submission of sensitive data.
- Apply the selected hash algorithm to the original password then split the resulting hash into half—PH1 & PH2. This step may not be necessary since the initial account setup process would have hashed the original password and split it in half. In fact it's preferred that this step be performed by the initial setup since it has access to the original password. On the other hand, if performed by either the browser or the server plug-in then each would have to have access to the original password.
- Concatenate the selected user attributes, a challenge phrase, and PH1 into string S1.
- Generate HS1 by applying the selected hash algorithm to string S1.
- Append HS1 + challenge phrase to the cookie field in the HTTP request header.

Once received by the server plug-in, the following steps are taken:

- Validate HS1 by applying the selected hash algorithm to the same set of user attributes in the required sequence.
- If valid, then apply the selected hash algorithm to the original password then split the resulting hash into half—PH1 & PH2. Again this step may not be required since the initial setup process would have already created PH1 & PH2.
- Concatenate the selected user attributes, a challenge phrase, and PH2 into string S2.
- Generate HS2 by applying the selected hash algorithm to string S2.
- Append HS2 + challenge phrase to the cookie field in the HTTP response header.

Once the browser plug-in receives the HTTP response, it authenticates the identity of the online entity by validating HS2. As a final confirmation, the browser plug-in will also perform a number of heuristic checks such as well-formed links/URL and the use of IP addresses in hyperlinks. The URL heuristic checks for symbols such as @ and the number of "Dots" in the URL [13]. Should both the client & server plug-ins successfully perform mutual authenticate, the browser plug-in would exit with success status and allow the user to input data into the Web form. It's important to note that once mutual authentication is performed, there would be not need for users to re-enter their credentials—original password. Therefore passwords are never exchanged between users and online entities except during the initial account setup process.

One of the advantages of Phishpin is that it enables both online users and entities to authenticate each other without revealing sensitive information. The use of OTP in combination with partial credentials and certificate chain validation makes it fairly challenging for phishers to obtain the user's credentials. Even in the unlikely scenario where a phisher is able to reverse the one-way hash, he/she would not be able to obtain the user's password since only half of the hash of the password was shared.

## 4 CONCLUSIONS

This paper presented the design of Phishpin, an integrated anti-phishing approach that combines client-based filtering and domain-based identity techniques to prevent phishers from masquerading as legitimate online entities. Phishpin integrates OTP, DNS, partial credentials, & filtering to enforce bi-directional authentication without revealing sensitive information.

One drawback of the proposed solution is the level of complexity associated with the original account setup, which requires online users to synch up attributes, selected hash method, sequence, & password with each online entity. That said, the initial setup process may be made easier by building one or more automated synch up tools. Another drawback is the added level of effort required by online entities to implement the server plug-in.

It's important to note that the focus of the proposed solution is on phishing attacks. Pharming, DNS poisoning, & malicious code attacks are not addressed by this solution. Should a hacker gain access to the client machine where the browser plug-in is running, she/he may be able to disable or even uninstall the browser plug-in.

## ACKNOWLEDGEMENTS

I would like to thank my Ph.D. advisor, Professor William Hafner for his support, time, ideas, and judgements that helped shape this paper.

## REFERENCES

Abu-Nimeh, S., Nappa, D., Wang, X., and Nair, S., 2007. A Comparison of Machine Learning Techniques for Phishing Detection. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, Pittsburgh, Pennsylvania, USA.

Chou, N., Ledesma, R., Teraguchi, Y., Boneh, D., and Mitchell, J., 2004. Client-side defense against Web-based identity theft. In *Proceedings of the 11th Network and Distributed System Security Symposium (NDSS'04)*, San Diego, California, USA.

Downs, S. J., Holbrook, M., and Cranor, L. F., 2007. Behavioral Response to Phishing Risk. In *proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit (eCrime'07)*, Pittsburgh, Pennsylvania, USA.

FDIC, 2004. *Putting an end to account-hijacking identity Theft.* http://www.fdic.gov/consumers/consumer/idtheftstudy/identity_theft.pdf.

Fette, I., Sadeh, N., and Tomasic, A. 2006. Learning to detect phishing emails. *Technical Report CMU-ISRI-06-112, Institute for Software Research, Carnegie Mellon University*. http://reportsarchive.adm.cs.cmu.edu/anon/isri2006/abstracts/06-112.html.

Kirda, E., and Kruegel, C. 2005. Protecting Users against Phishing Attacks. In *proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05)*, Edinburgh, UK.

McCall, T., 2007. *Gartner Survey Shows Phishing Attacks Escalated in 2007; More than $3 Billion Lost to These Attacks, Gartner, 2007.* http://www.gartner.com/it/page.jsp?id=565125.

Phishtank. *Phishing, 2008*. http://www.phishtank.org.

Raffetseder, T., Kirda, E., and Kruegel,C., 2007. Building Anti-Phishing Browser Plug-Ins: An Experience Report. In *proceedings of the 3rd International Workshop on Software Engineering for Secure Systems (SESS'07)*, Minneapolis, Minnesota, USA, 2007.

Wikipedia. *Phishing, 2008*. http://en.wikipedia.org/wiki/Phishing.

Wu, M. 2006. *Fighting Phishing at the User Interface*. http://groups.csail.mit.edu/uid/projects/phishing/minwu-thesis.pdf.

Wu, M., Miller, R. C., Little, G. 2006. Web Wallet: Preventing Phishing Attacks by Revealing User Intentions. In *proceedings of the Symposium On Usable Privacy and Security (SOUP'06)*, Pittsburgh, Pennsylvania, USA, 2006.

Zhang, Y., Hong, J. I., and Cranor, L. F. 2007. Cantina: a Content-based Approach to Detecting Phishing Web Sites. In *proceedings of the 16th International Conference on World Wide Web (WWW'07)*, Banff, Alberta, CA, 2007.