PwdIP-Hash: A Lightweight solution to Phishing and Pharming Attacks

Baber Aslam, Lei Wu and Cliff C. Zou
School of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL, USA

***Abstract*—Phishing and Pharming, the leading threats to identity theft, result in losses of millions of dollars each year. Many solutions have been proposed to guard against these attacks. Among them, password-based solutions may require additional hardware and are still vulnerable to man-in-the-middle attack; multi-challenge/response based solutions are mostly complicated and may also be susceptible to denial-of-service attacks; and detection-based solutions are ineffective if users dismiss warnings generated by these solutions.**

In this paper, we present a novel lightweight password-based solution that safeguards users from Phishing and Pharming attacks. The proposed authentication relies on a hashed password, which is the hash value of the user-typed password and the authentication server's IP address. The solution rests on the fact that the server connected by a client using TCP connection cannot lie about its IP address. If a user is unknowingly directed to a malicious server by a Phishing or Pharming attack, the password obtained by the malicious server will be the hashed password tied to the malicious server's IP address and will not be usable by the attacker at the real server, and hence, the Phishing/Pharming attack will be defeated. The proposed solution does not increase the number of authentication messages exchanged, nor requires addition hardware tokens. The solution is also safe against denial-of-service attacks since no state is maintained on server side during the authentication process. We have prototyped our design both as a web browser's plug-in and as a standalone application. A comprehensive user study was conducted, and the results show that around 95% of users think the proposed solution is easy to use and manage; 79% of users have shown willingness to use the application to protect their passwords.

Keywords- design; web security; usability; Phishing; Pharming; password authenication.

I. INTRODUCTION

Convenience, efficiency, reduced cost and environmental friendliness have been the major driving forces for the increases in Internet usage. Today, every user has multiple online accounts to serve his different needs: email, social networking, online banking, remote working etc. All these accounts contain certain personal sensitive information which if stolen can be used by attackers for monetary or other purposes. Every year millions of dollars are lost due to Internet related crimes (or Identity thefts) [1]. Among various identity thefts attacks, the major threats are *Phishing* and *Pharming*. Both Phishing and Pharming aim at stealing a user's sensitive information by directing him to a malicious website; the stolen credentials are then used for malicious purposes. The utility of these stolen credentials to attackers depends on their validity, even if the credentials are one-time and valid for a short time period, they may still be used for malicious purposes. Phishing starts with a spam (but seemingly legitimate) email; it uses social engineering to obtain user's sensitive information either using forms within the email or luring a user to a malicious (but seemingly legitimate) website via a link within the email. Pharming, on the other hand, uses Internet (DNS servers, DNS resolvers, web servers etc) vulnerabilities to direct a user to a malicious (but seemingly legitimate) website where his credentials are stolen. Pharming is more dangerous since a user may be unknowingly taken to a malicious website even if he/she types the correct web address. Challenge/response type of authentication [8,9] usually safe against these replay attacks, can still be vulnerable to man-in-the-middle (MITM) attack: an attacker places himself between a user and the authentication server, and relays challenges/responses between the user and the server, thus capturing all the authentication credentials.

SSL/TLS is mostly being used to provide the authentication and confidentiality on the Internet [2]. It provides a mechanism to achieve mutual authentication via certificates. Current implementations use server side certificates to authenticate a server whereas client side authentication uses user name and password. The server side authentication is normally defeated because of human factor [3], e.g., a user may fail to differentiate between a HTTP and a HTTPS session either due to his lack of knowledge or due to attack sophistication. Further, most users are likely to dismiss warnings generated by web browsers when a server presents an incorrect certificate [3]. These are the major reasons for the success of Phishing and Pharming attacks.

Many solutions have been proposed to guard against these attacks. They can be classified as either active or passive. Active solutions, such as web browser add-ons [4], are not fully secure since they have false negatives and depend on

users to act on the warnings which users generally ignore [3]. Passive solutions can be password-based [5 - 7] or protocol-based [8, 9]. Protocol based solutions increase the number of messages exchanged between server and client, thus lengthening the authentication process. Further, these solutions require major modifications in existing authentication mechanisms. Multi-step authentication schemes may be vulnerable to denial-of-service (DoS) attacks, since the server needs to maintain state (thus commit its recourses) for each authenticating client at least till the completion of authentication. A number of password based solutions generate one-time-passwords using either a hardware token (which increases the cost and complexity) [5, 6] or a trusted application (that generates passwords or does authentication on user's behalf) [7]. These solutions increase the attack complexity (introducing timing constraints) and cannot eliminate MITM attack possibilities. Some solutions incorporate server names to generate server specific passwords, thus protecting against user behavior of using same passwords for all accounts. When an attacker steals a password from weak servers, the stolen password cannot be used to attack the same user's other accounts [5 - 7]. However, these solutions are still vulnerable to MITM or replay attack.

In this paper we present a new passive password-based solution. The proposed authentication relies on a hashed password, which is the hash value of user-typed password and the authentication server's IP-address. The solution rests on the fact that *the server connected by a client using TCP connection cannot lie about its IP address*. In case of MITM, it will be the attacker's IP address since it will be acting as authentication server to the client. Thus the hashed password tied to attacker's IP address will not be usable by the attacker on the actual authentication server. In this way, the solution not only prevents exposure of a user's real password to a malicious server, but also prevents man-in-the-middle attack even if users dismiss browser's security warnings. The proposed solution does not increase the number of authentication messages exchanged, nor requires addition hardware tokens. The solution is also safe against denial-of-service attacks since no state is maintained on server side during the authentication process.

We have prototyped our design both as a web browser plug-in and as a standalone application. We also carried out a comprehensive user study of our implementation. The study has shown that the design is easy to use and users have shown their strong willingness to use the design if a version for their favorite browser is available.

The rest of the paper is organized as follows. Section II discusses the related work, section III presents the proposed solution, section IV gives the implementation details of our solution, section V presents the user study methodology/results and finally section VI presents conclusion and future work.

II. RELATED WORK

Active or detection based solutions, such as web browser add-ons or toolbars, detect the malicious websites based on either the black lists or the web content [11, 4]. The tools that use black list are vulnerable to an attacker's malicious website until the malicious site has been added in black list. Further these can also be countered by obfuscating the URL, e.g., routing through another domain, for example using content distribution networks [12, 13]. The tools which analyze the web content normally wait for entire page to load, if the web page takes too long to load then the tools decision will also be delayed and in this time a user may navigate away using some link on the malicious page. The delay can easily be introduced through loading of an invisible image etc [13]. Further, these are not very secure as they depend on user to act on warning and these also have high false negatives [13].

A class of solutions transfers part of initial registration and login onto portable smart devices (PDA, mobile phones etc). Smart mobile device is used to initially register the website; storing some identifying information about website and also the user's credentials. Later, when user wants to login, the smart device first confirms validity of the website and then fills in the user's credentials via Bluetooth etc [14- 16]. These solutions reset on the assumption that the mobile devices will not be compromised, however this assumption is not valid since PDA/smart phones are increasingly being used for browsing the Internet and these normally do not have elaborate protection systems such as antivirus, anti-malware etc. There are also many solutions which use smart cards/tokens to store and process the shared secrets to generate login credentials [5, 6, 17, 18]. The solutions using smart cards or other hardware tokens are expensive and are usually only implemented by high risk or financial institutions. The user may have to carry one device for each of his account. These may also be lost or stolen. The home user has no incentives to use this for all of his accounts. Further, smart card readers are considered as an add-on and are not included in basic configurations of computers which means these are not available everywhere especially on public computers/kiosks (where one needs most security).

A class of solutions resets authentication on server's identification such as domain name or IP address [7-9, 19]. [7, 8, 19] generate server specific passwords from one master password. These solutions address the user's tendency to use same password across several accounts [28], the attacker, in this case, can capture a password from less secure server and use it to access other high security accounts. These solutions do not address Pharming attacks and reply attacks. In replay attacks the password captured from same server is used for a later access to same server and also Pharming attacks. Sharifi et al. have presented a multi-step challenge response authentication mechanism to guard against Phishing [9]. The solution resets on SPEKE and incorporates server's IP address to guard against Phishing [20]. However, the solution increases number of authentication steps thus increasing complexity and also inherits SPEKE's vulnerabilities [21, 22].

Our work comes closer to PwdHash by Ross et al. [7]. They have used server names to generate server specific passwords from one master password. The hashing function uses domain

name and the master password to generate server specific password. It does not defend against Pharming attacks since the hashed password is directly used by a server. We are using server's IP address to hash the password, but we do not use one master password thus if the password is compromised through other social engineering methods then only one or a few accounts (which share the same password) will be at risk. Further our solution also guards users against Pharming attacks in addition to Phishing attacks. The one time nature of passwords also protects the user against replay attacks.

Our solution differs in a number of ways from above described solutions. Our solution provides safeguard against both the Phishing and Pharming attacks without any additional devices or hardware tokens. We do not assume that user takes care of browsers warnings. Our solution do not require multiple challenge response steps and also does not require the server to maintain state, this also makes the solution immune to DoS attacks.

III. Attack Models

In this paper, our focus is on the attacks that are targeted at user's login credentials i.e. username and password. An attacker can use these stolen login credentials to either masquerade the user (and steal user's personal sensitive data stored in the account) or initiate transactions on user's behalf. The paper does not solve the attacks where user enters his/her personal sensitive data (other than username and password) in form fields within emails or when visiting malicious/masquerading servers. The paper does not address Dynamic-pharming attacks in which an attacker dynamically changes the IP address returned for a particular domain name and exploits name-based same origin policy to hijack a session after authentication [10]. Further, the solution does not offer protection against malwares, spywares, keyloggers etc running on user's computer.

A. Password Reuse Attack

It is normal tendency that users share same passwords among multiple accounts. The attacker captures the password from a relatively insecure server, which may not be using https for logins e.g., [25], and then uses it to log in to user's other accounts sharing the same password.

B. Replay Attack

The attacker captures the password for a particular account and later reuses it for the same server. This type of attack may be used even in case of short-lived passwords within a small time window. This type of attack is successful against solutions that generate server specific passwords from a master password such as the PwdHash solution [7].

C. Server Masquerading Attack

The attacker masquerades a legitimate server; it can use different Phishing and Pharming techniques to trick the user into believing that it is real server. This type of attack is successful even with https and encrypted login credentials.

D. Man-in-the-middle (MITM) Attack

This type of attack is used against multi-step authentication systems. The attacker places him/herself between the user and legitimate server; acting as server to the user and user to the server. In this way, the attacker passes all the challenges/responses between user/server and gains access to user's account.

IV. Proposed Solution

A mechanism that is safe from MITM attack can withstand other attacks mentioned in Section III, therefore we assume attackers are capable of launching MITM attack. A user/client may be directed to a MITM (attacker) server via various Phishing/Pharming techniques. Further, we do not assume any vigilance on the side of users, i.e., a user may fail to differentiate between legitimate and malicious servers (e.g., between an http and an https session), or may dismiss various web browsers' security warnings.

A. Basic Idea

Typically (not going in SSL/TLS details), when a user/client wants to access his account (e.g., email), he initiates an http connection (either by entering the URL or clicking on a link) to the server (e.g., gmail.com). The URL is resolved to an IP address and a TCP connection request is sent to the server. The server responds by sending the login page and its certificate. The client's web browser authenticates the server (or generates security warnings in case of facing incorrect server's certificate). The user then enters his credentials (e.g., username and password) which are then sent to the server through SSL/TLS tunnel. The server verifies the credentials to complete the login process.

Therefore, in order to initiate and complete the login, a client must be able to know the IP address of the authentication server because of the underlying TCP connection. We can safely assume that the IP address of authentication server does not change during the authentication process. That is, any load balancing etc will not be conducted during the initiation of authentication process from the server side. That means, for a given session, we can associate a particular IP to the authentication server. We use this property to generate the secure password that is tied to the IP address of the authentication server. If the user is somehow directed to a malicious server by a Phishing or Pharming attack, the password obtained by the malicious server will be tied to the malicious server's IP address and will not be usable at the real server, and hence, the attack will be defeated.

B. Notations and Function Definitions

We define several notations/functions that we will use in the formal description of our solution (Table I). A server's

certificate is essentially represented by its public and private key pair (K^+, K^-) where K^+ is the public key and K^- is the private key. $E_K^+\{M\}$ defines an encryption function on message M using the public key K^+. Public cryptography is very resource intensive therefore data encryption is usually carried out using a randomly generated symmetric session key and only the session key is encrypted using public cryptography [26]. The encryption function $E_K^+\{M\}$ defined above employs similar techniques and we will not show the details for simplicity and compactness. $H_K(M)=H(K,M)$ defines a secure hash function, such as SHA-1 [27]; it is a one-way function such that given M, it is easy to compute $H_K(M) = M_H$, but it is computationally infeasible to compute M and K given M_H.

TABLE I. NOTATIONS AND DESCRIPTIONS

Notations	Descriptions
C	Client/User
S	Server
IP_S	Server IP-address
N_S	Nonce generated by Server
P_H	Hash value of user-typed password P
$Cert_S$	Cerificate of server S
(K^+, K^-)	Public and private key pair of server
$E_K\{M\}$	An encryption function on message M using the key K
$H_K(M)$	A secure hash function using key K on message M

C. Assumptions

We assume that an attacker does not have access to the real server's private key or any other secret that is used to store the passwords on server machines. We also assume that the client side is free from any malware such as keyloggers etc. In addition, we do not assume that a user is able to identify a legitimate server from a fake server, or will act on browser's security warnings.

We also assume that a user has already registered with the real server and the server knows the user's login credentials. The server can employ methods to guard against stolen credentials attacks such as encrypting the credentials with its private key etc. The PwdIP-Hash will only be used for login and will not be used for initial account registration or while updating the passwords, therefore registration does not require the PwdIP-Hash.

D. Proposed Solution

The basic process of proposed authentication for a client/user (C) authenticating with a server (S) is described below (Fig. 1).

- Client requests the login page by setting up a TCP connection.
- Server sends its certificate ($Cert_S$) to client.
- Client, using a secure hash function $H_K()$ with key K, computes hashed password $P_H = H_K(P)$; where $K= IP_S$, P is the user-typed password and IP_S is server's IP address. Client then encrypts P_H with server's public key $(K^+) - E_K^+\{P_H\}$.
- Client sends encrypted P_H to server.
- Server also generates hashed password P_{HS} using its saved Client's password P and its IP address IP_S, then verifies with the received P_H.

1	C	Set up TCP connection
2	$S \rightarrow C$:	$Cert_S$
3	C	Compute $P_H = H_{IPs}(P)$; $E_K^+\{ P_H \}$
4	$C \rightarrow S$:	$E_K^+\{ P_H \}$
5	S	Compute $K= H_{K+}(IP_S)$; $P_{HS} = H_K(P)$; Verify $(P_{HS} = P_H)$

Figure 1. Basic Process of Authentication between a client and a server

If the client is connected to a MITM/malicious server (with IP address IP_A) and fails to differentiate it from the actual server, then the attacker will send $Cert_A$ to client, client will send P_H based on IP_A. In this case when the attacker relays received P_H to the actual server, the authentication will fail because the actual server has a different IP address from IP_A.

The presented authentication scheme generates server specific passwords, which means the same user-typed password will be translated into different hash passwords for different servers. Therefore, this scheme also guards against attacks password reuse attack targeted at user's behavior of using the same password for different accounts [28].

The authentication scheme can easily be modified to generate server specific one-time passwords. In this case the server also sends a nonce which is used along with IP address to generate key. The modified authentication process is described below:

- Client requests the login page.
- Server generates nonce (N_S), encrypts N_S with its private key $(K^-) - E_K^-\{N_S\}$.
- Server sends its certificate ($Cert_S$) and $E_K^-\{N_S\}$ to client.
- Client, using a secure hash function $H_K()$ with key K, computes hashed password $P_H = H_K(P)$; where $K=H_{K+}(N_S \mid IP_S)$, P is password, IP_S is server's IP address, and $(x \mid y)$ defines concatenation of x and y. Client then encrypts P_H with Server's public key $(K^+) - E_K^+\{P_H\}$.
- Client sends $E_K^+\{P_H\}$ and $E_K^-\{N_S\}$ to Server.
- Server also generates hashed password P_{HS} using its saved Client's password P, nonce N_S decrypted from the received $E_K^-\{N_S\}$, and its IP address IP_S, then verifies with the received P_H.

If the server needs to maintain the state for each authenticating client by storing N_S till completion of authentication, it may be vulnerable to denial-of-service attacks. To guard against this vulnerability the proposed solution does not maintain the state for each authenticating client and lets the client include N_S (encrypted with server's private key) with its response in Step 5 (Fig. 2).

1	C	Set up TCP connection
2	S	Generate N_S, compute $E_K^-\{N_S\}$
3	$S \rightarrow C$:	$E_K^-\{N_S\}$, $Cert_S$
4	C	Compute $K = H_{K+}(N_S \mid IP_S)$; $P_H = H_K(P)$; $E_K^+\{P_H\}$
5	$C \rightarrow S$:	$E_K^+\{P_H\}$, $E_K^-\{N_S\}$
6	S	Compute $K = H_{K+}(N_S \mid IP_S)$; $P_{HS} = H_K(P)$; Verify $(P_{HS} = P_H)$

Figure 2. Authentication process between a client and a server

The solution will require modifications on both client and server sides. For server side, a module can be added to handle necessary generation and computation steps. For client side, a web browser add-on can be installed to handle authentication. The solution will also work in case of single-sign-on cases, where the authentication is done by one central server on behalf of different servers. In this case, the IP address of authentication server will be used instead of the server with which user has an account.

E. Features of the proposed PwdIP-Hash theme

The solution does not require additional hardware tokens, does not increase the number of authentication steps and does not require authentication server to maintain any state during authentication. Therefore, it is economical, light weight and immune to multi-transaction based denial-of-service attacks. The solution does not require users to identify malicious activity or to act on security warnings, thus making it effective even if a user is unknowledgeable and dismisses all warnings generated by web browser. In addition, one-time property of the password guards against password reuse attacks. Time stamps can also be incorporated to prevent replay attacks, the server can send time stamp along with the nonce and compare the timestamp with current time when the hashed password is received. If it is within some predefined threshold only then the password is accepted for further verification. This prevent the attacks where the hashed password and nonce are somehow captured and are being replayed later to gain access to user's account.

V. IMPLEMENTATION

We considered two possible options for implementing the proposed authentication scheme: as a standalone application and as a browser plug-in/add-on. Another option studied by researchers is to modify the login page, but it has a security drawback that the malicious server can always send a modified login page and steal the un-hashed password [7]. Therefore we discarded the login page modification option and implemented the scheme as a browser plug-in and as a standalone application. For initial tests/trials we have restricted ourselves to Microsoft's browser – Internet Explorer.

The source codes of both implementations can be downloaded freely from our server [32]. A user must have necessary permissions to install the plug-in. On the other hand, the standalone version executable program can work without installation; this will be useful for situations where a user does not have necessary permissions/rights to install programs, e.g., when she uses a public computer in library or cafe. In these situations, the user can carry around the standalone program in a USB key. If a user cannot carry the standalone program around with her, she can connect to a download server to download the standalone application wherever she wants to use it. To guard against DNS attacks against the download server, users are advised to use the IP address instead of server name for connecting to the download server.

Our solution requires modification on both server and client side. The server side modification means that the solution will only be used by servers who have opted to implement the necessary modifications. For this reason, a user should somehow remember which of his accounts are protected by the solution and only use the standalone application /plug-in for those accounts. This could be a big challenge for a user if she has many accounts.

We considered two options to resolve this challenge; server registration and bookmarking. In the first option, all servers supporting the solution should register themselves with the company who releases the standalone program and the plug-in; the standalone program and the plug-in code contain the list of all those registered servers, and contact the company's server to periodically update the list in the similar way as current anti-virus software. The second bookmarking option can be done by each user. When a user registers with a server that supports the application, the server can prompt the user to bookmark the server with the PwdIP-Hash plug-in/standalone application. Later, the user can log in using PwdIP-Hash if a server is in the bookmark. This option has portability issues since user's bookmarks will be only present at his/her own system. This can be resolved if a user uses online bookmarking services or carries the bookmark file with him.

The solutions that do not require server side modification may seem easy to deploy, such as the PwdHash [7], but they may not be compatible with each and every server since servers have different passwords rules such as length and composition of passwords. The generated password may not meet the specification, one possible solution is to add configuration file [7], but with a large and ever increasing number of servers, each having its own password rules, it may become impossible to keep the configuration file updated.

Visual feedbacks or cues are a very important feature of any application; they help users to make a mental model of how an application works and also improve the chance of correct operation of any application [23]. For this reason we used the activation button similar to [7], this turned into a check-mark sign (over green circle) when the application was active and remained a cross sign (over red circle) otherwise (Fig. 3 & 4). We also asked the users in our user study to give there preferences on the applications visual cues etc. 79% of users preferred a password application that gives feedback or strong visual cues.

A. *Browser Plug-in/Add-on*

A user friendly implementation should automatically detect the password fields and activate the hashing process. However, if an attacker presents a login page with normal text field instead of password/protected fields then password hashing will not take place [7]. Because of this and many other security issues, covered in detail by Ross et al. [7], we decided to build our plug-in based on PwdHash model developed in [7]. In this case, the user activates the application (pressing F2 after clicking in the password field) before typing the password. The activation by a user also solves the issues of incompatible design of login pages among different websites; each website has its own design and it may become difficult to automatically detect and populate the password.

We reused the basic key-hook framework of [7] and replaced some functions according to our own needs. We implemented our hash class, which accepts password and IP address (gets it from *gethostbyname* function) as parameters and generates the hashed password. For convenience, here we used MD5 as the hash function. In the real world application, of course, we might apply other hash functions. We also replace the icons in toolbar to make them more noticeable (Fig. 3, Fig. 4).

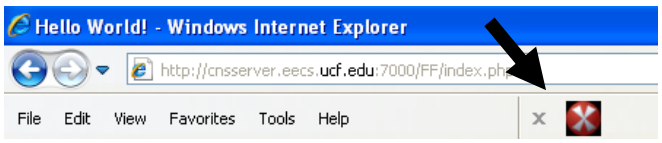

Figure 3. Inactive status

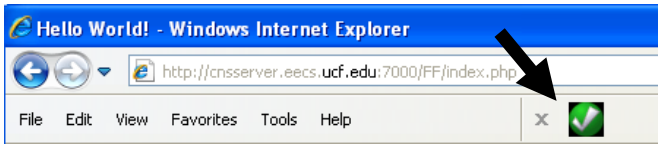

Figure 4. Active status

B. *Standalone Application*

The standalone application is illustrated in Fig. 5. It has two inputs; the domain name of the authentication server and the password. The user can first load the authentication server's login page either by typing the URL or using bookmarks or clicking a hyperlink. Next the user activates the standalone application, which will present URLs of all currently loaded web pages in a dropdown list (as Fig. 6 shows). The user selects the URL of the desired login server, enters the password, and then clicks the "Generate Password and Copy to ClipBoard" button. The standalone application will generate the hashed password and copy it to clipboard (see Fig. 7). After that the user can manually paste the hashed password to the relevant password field in the login page and log in.

The standalone application is a dialog-based program. Its automatic URL detection feature is currently compatible with IE only, for other browsers, users need to type-in the domain/URL themselves.

Figure 5. PwdIP-Hash standalone application

Figure 6. PwdIP-Hash detects current loaded IE pages and presents the corresponding URLs in its drop-down list.

Figure 7. PwdIPHash copies hashed password to clipboard

Compared with the plug-in code, the standalone application has the advantage of browser independent. Currently the number of web browsers is continuously increasing and each of them is also frequently updated with new versions. Therefore, it becomes increasingly difficult for the plug-in program to support all the browsers and new releases may make the plug-in incompatible. In addition, the browsers for handheld devices generally do not support plug-ins. Another advantage is that the standalone application can be carried around by a user to be conveniently used on public computers.

C. Fallback Mechanism

We also considered the options for users to log in from a computer where neither the plug-in nor the standalone application can be used. Obviously, one possible solution could be to use server components [29] to detect the plug-in and if it is not present then ask user to install it. This type of solution can easily be defeated if the login page is modified by attacker/malicious-server. The second option could be to always prompt the user, before login, to confirm that the plug-in or standalone application has been installed and if not install it. This solution relies on user's vigilance and may fail if a user fails to detect the absence of plug-in/standalone application or if the user does not have sufficient privileges to install the plug-in such as on kiosks/public access computers at airports etc. Further, it also has the same vulnerability that exists in the first option. The third option is to have an online password hashing server that behaves equivalent to the standalone application; a user can access the server to generate the hashed password [7]. This solution is though easy to implement but may become a single point of failure especially if an attacker launches a fake online password hashing server. Further, if a login server uses multiple IP addresses, the online password hashing server may use an IP address different from the one to which the user is currently connected to, and hence, cannot provide a correct hashed password.

The other two options are: deny a user from logging in or let the user log in without the added security offered by our application. In the second case we can modify the server to first check the hashed password and if that does not match then proceed to standard authentication procedure. The security of this will not be worse than the existing authentication schemes. We added a question in user study to ascertain users' preferences as to whether they would like to be logged-in without the additional security or would like to be denied login if the plug-in/standalone application cannot be used. 73.5% users preferred to be able to log in even if the added security is not available to them. This highlights a very important preference of users: a security product, no matter how good and secure it is, must be user-friendly in order to be widely accepted and used by the general population. Security providers should keep this in mind while designing security solutions.

VI. Usability Study

A comprehensive user study was carried out to check the usability of the proposed solutions. For this a total of 34 users were recruited, this number is 1.7 times of the number required for a decent usability study as Faulkner has shown that twenty users can find more than 98% of usability problems [24]. To help readers to understand our user study, we have posted our user study questionnaires on our server [32].

A. Study Design Considerations and Settings

Users were briefed at the beginning to ensure that all users get the same information. The briefing covered basic purpose of our application, the components of the user study and how to use the application. Each user was also given a brief manual which contained the stages of study and usage of the application as a ready reference.

All tests were conducted in a single location on the same computer; this was especially done to control the computer performance and the environment variation. Further, all users were asked to perform the tests on our own developed web server. This ensured that all users were presented with the same interface. Care was taken, so that the login page does not resemble any of the famous login pages such as email or social networking sites, since this similarity may produce bias in results between users who are familiar with the websites and those who are not familiar.

B. Stages

The user study was divided into four stages: pre-trial questionnaire, short Internet/computer security tutorial, application trial and post-trial questionnaire.

1) Pre-trial questionnaire: After initial briefing a user was given a pre-trial questionnaire which besides demographic information also collected some data regarding the user's familiarity with Internet/security etc.

2) Internet/computer security tutorial: Next a user was asked to go through a brief tutorial on Phishing and Pharming. This was incorporated to educate the user on these topics since the user may not be aware of the threat for which we have designed the solution. The tutorial was based on the material from [30, 31].

3) Application trial: Our prototype is used only for login and not during signup or password update/change operations, therefore in order to check user's response on the difference between password entering mechanisms the trial besides login also included the signup and password change operations. The trial consisted of four steps.

a) Step 1: Create a user account on the server, users were free to write their usernames and passwords on provided sheet since a new username and password may be difficult to remember and users were encouraged to use some new usernames other than their normal ones to ensure privacy. Users were not required to use the application in the account signup stage.

b) Step 2: Log in to the account. Users were required to use the plug-in for filling up the password field.

c) Step 3: Change the password and log out. Changing password does not require the activation of application.

d) Step 4: Again log in the server this time using new password. Users were asked to use the standalone version, this time, for login.

4) Post-trial questionnaire: The final stage was a post-trial survey which asked for users' experience and recommendations.

C. Participant Recruitment and Demographics

The study was advertised via flyers which were posted in different departments of our university. The participants were required to be familiar with computer/Internet and login based accounts such as web emails etc. Interested participants were given the consent form, and those who agreed were recruited for the study. To facilitate the recruitment, each participant was given a small amount of compensation money. Overall we recruited 34 users for this user study. The 34 participants ranged in age from 18 to 37 (Mean=23.6). In gender distribution 56% were male and 44% were female.

In terms of educational level, 41% had a high school diploma, 21% had Associate, 10% had a Bachelors degree and 28% had a Masters degree. In terms of their majors, 52% were from non-technical disciplines (such as Accounting, Psychology, Business, Art, Film, Elementary education, Writing, Teaching, Music, etc) and the rest 48% were from technical disciplines (such as Computer Science, Mechanical/Electrical/Computer/Civil Engineering, Physics, Biology/Microbiology, etc).

D. Participants' awareness to computer/Internet/security

In terms of familiarity with computer/Internet, on average each user spent 6 ~ 7 hrs on Internet daily and 94% of users reported that they have used Internet for online banking, bill pay or purchases. On average each user had 10 ~ 11 online accounts (min=3, max=25) and was using 4 ~ 5 different passwords for these accounts (min=1, max=10). Average length of the longest password among users was 11 ~ 12 characters (min=8, max=21) and that of shortest password was 6 ~ 7 characters (min=3, max=10). The password shared by most of a user's accounts had an average length of 8 ~ 9 characters and was shared among 5 ~ 6 different accounts (min=1, max=20).

Participants were also asked to report their familiarity with terms such as Phishing, Pharming, https, digital certificates, etc. 26% reported that they were not familiar with at least half of the terms. 32% were not familiar with Phishing, 79% were not familiar with Pharming. Only 18% were familiar with both Phishing and Pharming. These statistics show that a large portion of people, even among college students, are not familiar with the potential threat introduced by Phishing and Pharming.

The large number of online accounts per user, password reuse habits and lack of awareness to security further highlights the threat which people are facing from Phishing and Pharming attacks.

E. Discussion

All 34 participants successfully completed the user registration step (step 1) of the trial; a few took more than one attempt. During the login phase using plug-in (step 2) some users forgot to activate the application and thus encountered the login failure error. Most of the users recovered from the error by consulting the user's guide and repeating the login again successfully after activating the application. Password change step (step 3) was also successfully completed by most of the users. Most of the users successfully completed login using standalone application in first attempt, though some users indicated the inconvenience of additional steps; but these additional steps also helped users to successfully log in. The detailed results are discussed in succeeding paragraphs.

The tutorial was aimed at increasing the participants' awareness to Internet security especially Phishing and Pharming. 91% of users agreed/strongly-agreed that they have learned something new from the tutorial, this also highlights the strong need of user education/awareness to Internet security, even for the young generation. 76% of participants agreed/strongly-agreed to consider improving their password habits so that their passwords are strong and distinct.

In response to the usability of PwdIP-Hash, 94% agreed/strongly-agreed that the task was easy, 97% agreed/strongly-agreed that the task was manageable, 85% showed their satisfaction with the user interface and functionality. 79% considered using the application if a version was available for their favorite browser. These statistics demonstrate that our solution is user-friendly and practical. In addition, 56% preferred plug-in version over standalone version.

All participants successfully completed the entire application trial, though some had to repeat certain steps more than once to complete the task. Fig. 8 gives detailed account of application trial attempts. 74% were able to complete all four application trial steps in first attempt, 27% failed to complete the step 2 (login using plug-in) in first attempt whereas 17% failed to complete step 4 (login using standalone) in first attempt. In case of login using plug-in the users forgot to activate the plug-in (pressing F2 after clicking in the password field) and in case of standalone users pressed enter-key after entering the password (which closed the application) instead of clicking the "Generate Password and Copy to ClipBoard" button. We have modified the standalone application so that pressing enter-key acts same as clicking the "Generate Password and Copy to ClipBoard" button.

For fallback mechanism, in case the application is not installed and cannot be installed at a public or a friend's computer, we asked the user whether they prefer not being able to log in to their accounts or allowed to log in to their accounts without the added protection of PwdIP-hash. Only 26.5% of users preferred to not being allowed to login if added protection of PwdIP-hash in not available to them.

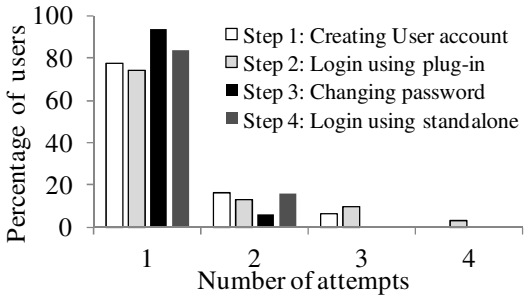

Figure 8. Number of attempts made by users to complete each step of application trial

VII. CONCLUSION AND FUTURE WORK

In this paper we have presented a lightweight solution that can effectively defend against both attacks. Our solution does not require any hardware tokens and does not assume that a user is able to differentiate between a fake and a legitimate website. We have prototyped the solution as a web browser plug-in and as a standalone application. The usability trials have shown that our prototypes are easy to use and most of the users have shown their willingness to use the solution if made available as a standalone (44%) or as a plug-in (56%) for their favorite browser.

PwdIP-Hash may generate incorrect password if the browser's and operating system's DNS caches are incoherent where the authentication server's domain name maps to two different IP addresses. One possible solution for PwdIP-Hash to obtain the same authentication server's IP address as the browser does is to do reverse DNS lookup on all established TCP connections (on http or https ports). We have tried this solution successfully on our prototype, but presently the reverse DNS lookup will introduce noticeable delay to our prototype. In future work, we intend to find a compatible and fast solution to resolve this issue.

We also intend to develop PwdIP-Hash for other famous web browsers such as Firefox, Chrome, Safari, etc and to compare their performance. Furthermore, a user study involving different solutions and involving more general participants than college students can give us more insight in how users see security and what are their preferences.

REFERNCES

[1] Internet crime report, 2008, Internet Crime Complaint Center.

[2] T. Dierks and E. Rescorla, "The TLS protocol version 1.2", RFC 5246, Aug. 2008.

[3] C. K. Karlof, "Human factors in web authuntication", PhD Thesis, University of California at Berkeley, Feb 2009.

[4] N. Chou, R. Ledesma, Y. Teraguchi, and J. Mitchell, "Client-side defense against web-based identity theft", In Proceedings of the 11th Annual Network and Distributed Systems Security (NDSS), 2004.

[5] Z.-C. Chai, Z.-F. Cao, and R.-X. Lu, "Effcient password-based authentication and key exchange scheme preserving user privacy", Lecture Notes in Computer Science, 4138:467-477, 2006.

[6] I-E. Liao, C.C. Lee, and M.S. Hwang, "A password authentication scheme over insecure networks," Journal of Computer and System Sciences, 72 (4) (2006), pp. 727–740.

[7] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, "Stronger password authentication using browser extensions", In 14th Usenix Security, 2005.

[8] M. G. Gouda, A. X. Liu, L. M. Leung, M. A. Alam, "SPP: An anti-phishing single password protocol", Computer Networks 51 (2007) 3715-3726.

[9] M. Sharifi, A. Saberi, M. Vahidi, and M. Zorufi, "A zero knowledge password proof mutual authentication technique against real-time phishing attacks. Information systems security", In 3rd International conference, ICISS, December 2007.

[10] C. Karlof, U. Shankar, J.D. Tygar, D. Wagner, "Dynamic pharming attacks and locked same-origin policies for web browsers", In Proceedings of the 14th ACM conference on Computer and communications security, October 28-31, 2007.

[11] PhishGuard. http://www.phishguard.com.

[12] The Coral Content Distribution Network. http://www.coralcdn.org/.

[13] Y. Zhang, S. Egelman, L.F. Cranor, and J. Hong, "Phinding phish: Evaluating anti-phishing tools", In Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS 2007), February 2007.

[14] W. Han, Y. Wang, Y. Cao, J. Zhou, L. Wang, "Anti-Phishing by Smart Mobile Device", IFIP International Conference on Network and Parallel Computing, 2007.

[15] B. Parno, C. Kuo, and A. Perrig. "Phoolproof phishing prevention", In Proceedings of Financial Cryptography (FC'06), February 2006.

[16] M. Mannan and P.C.V. Oorschot, "Using a Personal Device to Strengthen Password Authentication from an Untrusted Computer", FC 2007 and USEC 2007, LNCS 4886, pp. 88–103, 2007.

[17] M. Lei, Y. Xiao, S. V. Vrbsky, C.-C. Li, and L. Liu, "A Virtual Password Scheme to Protoct Passwords," In Proc. of the IEEE International Conference on Communications, IEEE ICC 2008.

[18] R. Oppliger, R. Hauser, and D. Basin, "SSL/TLS session-aware user authentication", IEEE Computer, 41(3):78–84, March 2008.

[19] J. A. Halderman, B.Waters, and E. Felten, "A convenient method for securely managing passwords", In Proceedings of the 14th International World Wide Web Conference (WWW 2005), May 2005.

[20] D. Jablon, "Strong password-only authenticated key exchange", ACM Computer Commununication Review., ACM SIG- COMM, vol. 26, no. 5, pp. 5-26, Oct. 1996.

[21] Q. Tang and C. J. Mitchell, "On the security of some password-based key agreement schemes", In CIS 2005, Part II, LNAI 3802, pp. 149-154, 2005.

[22] M. Zhang, "Analysis of the SPEKE password- authenticated key exchange protocol", IEEE Com munications Letters, vol. 8, no.1, pp. 63-65, Jan. 2004.

[23] S. Chiasson and P.C. V. Oorschot, "A Usability Study and Critique of Two Password Managers", In Proceedings of the 15th conference on USENIX Security Symposium, Security '06, August 2006.

[24] L. Faulker, "Beyond the five – user assumption: Benefits of increased sample sizes in usability testing", Behavior Research Methods, Instrumets, & Computers, 35(3):379-383, 2003.

[25] EDAS Conference Services, http://edas.info/.

[26] Schneier, Bruce, "Applied Cryptography", 2nd, New York: John Wiley & Sons, 1996.

[27] FIPS 180-1 "Secure Hash Standard", Federal Information Processing Standard (FIPS), Publication 180-1, National Institute of Standards and Technology, US Dept. of Commerce, Washington D.C., April 1995.

[28] D. Florencio, C. Herley, "A large-scale study of web password habits", In Proceedings of the 16th intl. conf. on World Wide Web, May 2007.

[29] BrowserObject™ .NET, http://www.browserobject.com/.

[30] Protect yourself from phishing scams, The PhishGuru, Carnegie Mellon, http://phishguru.org/designs/all_phishguru_designs.pdf.

[31] S. Srikwan and M. Jakobsson, Security Cartoons, http://www.SecurityCartoon.com.

[32] PwdIP-Hash. http://www.cs.ucf.edu/~czou/PwdIP-Hash/.