PwdIP-Hash

A Lightweight Solution to Phishing and Pharming Attacks

Baber Aslam, Lei Wu and Cliff C. Zou
University of Central Florida, Orlando, FL, USA

***Abstract*—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 a 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 thus defeating Phishing/Pharming attack. The proposed solution does not increase the number of exchanged authentication messages, nor does it need hardware tokens as required by some previously proposed solutions. 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. The results show that around 95% of users think the proposed solution is easy to use and manage. Further, around 79% of users have shown willingness to use the application to protect their passwords.**

Keywords- design; web security; usability; Phishing; Pharming; password authentication

I. INTRODUCTION

Today, every user has multiple online accounts (such as email, social networking, online banking, remote working etc) to serve her different needs. All these accounts contain some 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 theft attacks, the major threats are *Phishing* and *Pharming*. Both Phishing and Pharming aim at stealing a user's sensitive information by directing her to a malicious but seemingly legitimate website. 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 website. Pharming is more dangerous since a user may be unknowingly taken to a malicious website even if she types the correct web address.

SSL/TLS is mostly being used to provide authentication and confidentiality on the Internet. 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 [2], such as a user's failure to differentiate between a HTTP and a HTTPS session (either due to lack of knowledge or due to attack sophistication) or a user's dismissal of web browsers' incorrect-certificate warnings [2]. These are the major reasons for the success of Phishing and Pharming attacks.

Solutions proposed to guard against these attacks can be classified as either active or passive. Active solutions, such as web browser add-ons [3], are not fully secure since they have false negatives and depend on users to act on the warnings, which users generally ignore [2]. Passive solutions can be password-based [4 - 6] or protocol-based [7, 8]. Protocol based solutions increase the number of messages exchanged between server and client, thus lengthening the authentication process. Further, 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 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) [4, 5] or a trusted application (that generates passwords or does authentication on user's behalf) [6]. These solutions increase the attack complexity (introducing timing constraints) but cannot eliminate man-in-the-middle (MITM) attack possibilities. Some solutions [4 - 6] incorporate server names to generate server specific passwords, thus guarding against password-reuse attack (where an attacker captures a user's password from a less secure server and uses it to access other more secure accounts) targeted at users' behavior of using same passwords for more than one account [17]. However, these solutions are still vulnerable to Pharming, MITM or replay attacks (in replay attacks a password captured from a server is used for a later access to the same server).

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 MITM attack even if users dismiss browser's security warnings. The proposed solution does not increase the number of authentication messages exchanged, nor requires hardware tokens. The solution is also safe against DoS 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 presents the proposed solution, section III gives the implementation details of our solution, section IV discusses the user study and finally section V presents the conclusion.

II. PROPOSED SOLUTION

In this paper, our focus is on attacks that target user's login credentials, i.e., username and password. A mechanism that is safe from MITM attack can withstand other attacks (such as replay attack, password reuse attack, etc); 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.

The paper does not solve attacks where a user enters 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 [9]. Further, the solution does not offer protection against malwares, spywares, key-loggers etc running on a user's computer.

A. Basic Idea

Typically (not going in SSL/TLS details), when a user/client wants to access his account (e.g., email), she 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). The user then enters her 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 (e.g., load balancing will not be conducted during the authentication process). This 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. 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 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.

C. Proposed Solution

Several notations/functions that we will use in the formal description of our solution are summarized in Table I.

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, defined by its key pair (K^+, K^-).
(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

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

- Client requests the login page (sets up a TCP connection).
- 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. The Server then discards $E_{K^-}\{N_S\}$.
- 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.

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 1. Authentication process 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 $Cert_A$ and $E_{K^-}\{N_S\}$ (received from actual server) to client, client will send P_H based on IP_A and N_S. 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 onetime server specific passwords thus guarding against password reuse attacks targeted at user's behavior of using the same password for different accounts [17]. Time stamps can also be incorporated to prevent replay attacks. The solution will require modifications on both client and server sides.

D. 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 an authentication server to maintain any state during authentication. Therefore, it is economical, light weight and immune to multi-transaction based DoS 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 a web browser.

The solution will also work in single-sign-on cases, where the authentication is done by one central server on behalf of different servers. Here, the IP address of the authentication server will be used instead of the server with which a user has an account. Further the solution will also work for clients using NAT to access the authentication server, since NAT only modifies the "from" field of the client's packets and does not modify the IP address of the authentication server.

E. Comparison with Similar Approaches

Quite a few solutions have also used server's identification (such as domain name or IP address) in authentication process [6-8, 10, 16]. [8] incorporates server's IP address to guard against Phishing, however, it inherits SPEKE's vulnerabilities [11, 12, 13]. [6, 7, 10, 16] generate server specific passwords from one master password and server's domain name by using some hash functions. This eases a user's burden to remember different passwords for all of her accounts and also addresses the password-reuse attack; however, these solutions have many other issues, which are listed below:

1) Vulnerable to Pharming and replay attacks: Use of server's domain name to generate passwords may be helpful in guarding against Phishing attacks but it does not address Pharming attacks where only the IP address can identify a malicious server and not the domain name. Further, the the solutions are also vulnerable to replay attack.

2) Incompatible generated passwords: Different servers have different security requirements on length/composition of passwords; this may make the generated passwords inacceptable. The solution [6] of using a configuration file to define each and every server's requirements is not scalable.

3) Master password - a single point of failure: If a master password is compromised due to social engineering or other methods then all associated accounts will be compromised. Whereas our solution does not use the master password so loss of password will have limited damage.

4) Difficult transition from unmanaged to managed passwords: A user needs to manually change the passwords for all the servers for which she intends using one of the previous solutions. Whereas in our solution a user needs not to change the passwords when she starts using our solution.

5) Difficult password change process: The passwords are generated from one master password therefore a password change may require changing the master password. This means all other passwords will also change so the user needs to manually update all passwords at once. Whereas in our solution the user can easily change the password of one account without affecting her other accounts.

6) Same passwords will be generated for different websites that share the same domain name/host (e.g. sites.google.com/site/abc, sites.google.com/site/xyz, etc). This is because these solutions only use the domain names and not the URLs in password generation.

III. IMPLEMENTATION

A. Design Considerations

We considered two possible options for implementing the proposed authentication scheme: as a standalone application and as a browser plug-in/add-on. The third possible option of modifying the login page was discarded due to its obvious security drawbacks [6]. For initial tests/trials we restricted ourselves to Microsoft's browser – Internet Explorer.

Visual feedbacks or cues are a very important feature of any application; they help users to make a mental model of how the application works and also to improve chances of correct operation of an application [14]. For this reason the activation button, we used, turned into a check-mark sign (over green circle) when the application was active and remained a cross sign (over red circle) otherwise (Fig. 2 & 3). 79% of users in our usability study preferred a password application that gives feedback or strong visual cues.

B. Implementation Challenges

The idea of hashing the password with server's IP address though seems simple in principle but is not simple in implementation. We faced several challenges during implementation of our proposed idea and had to refine the basic principle of hashing to overcome these challenges.

1) Server's IP Address

A challenge faced during implementation was to get the IP address of the (authentication) server to which the client is currently connected. This becomes difficult since browsers do not offer/provide a mechanism to share with other applications the IP addresses of currently connected servers. We considered several different options to solve this challenge:

Option 1: Reverse DNS lookup

One possible option was to do reverse DNS lookup on all established TCP connections (http or https). We tried this solution successfully on our prototype, but the reverse DNS lookup introduced noticeable delays.

Option 2: Additional DNS lookup

The other option was to directly query the DNS with the server's domain name to obtain its IP address. This option, however, had two potential problems:

- *Mismatched IP addresses*: If the browser's and operating system's DNS caches are incoherent then the server's domain name may map to two different IP addresses. Also, if the server has more than one IP addresses then we could also get an IP address mismatch between the browser and PwdIP-Hash. Both situations will result in computation of incorrect hashed password.
- *Exploitation by Attacker*: If the client is connected to a Phishing (attacker's) website where the attacker controls his site's DNS, then the attacker can provide the legitimate website's IP address as response to the second DNS query. In this case the attacker will be successful in capturing the usable password.

Option 3: Additional DNS lookup- Revised Hashing Method

The first problem discussed in Option 2 can be solved by letting the authentication server to check the password against all of its assigned IP addresses for verification.

The second problem in Option 2 can be solved by using the domain name of the server along with its IP address to compute the password; the attacker's (Phishing) website will have a different domain name and hence the generated password will not be usable on the legitimate server. This can be implemented by re-defining key K as $K=H_{K+}(Domain\text{-}name \mid N_S \mid IP_S)$, in steps *4 & 6* of our solution shown earlier in Fig. 1.

2) Different Passwords for Websites sharing Host

Another challenge was to generate different passwords for websites that are hosted on the same server. Since in this case the domain name would resolve to the same IP address and a password captured from one website can be used for authentication with the other websites on the same hosting server. Previous modification of using domain name in addition to IP addresses can also solve this issue. But the websites that share domain name/host (e.g. sites.google.com/site/abc, sites.google.com/site/xyz, etc) will still be prone to this attack. A possible solution to this is to use URL instead of shared domain name to ensure unique passwords among these sites.

3) Clients using Proxy Servers

If a user is accessing the Internet via a proxy server then the user's browser will be connected to proxy server instead of authentication server. In this case the browser will see the IP address of proxy server and not the authentication server. In this case our previous modification of getting server's IP address by separate DNS query will get the IP address of server.

C. 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. Because of this and other security issues identified by Ross et al., we decided to use the plug-in activation model given in [6]. In this case, the user activates the application (pressing F2 after clicking in the password field) before typing her password. This also solves the issue of heterogeneous design of login pages among different websites.

We reused the basic key-hook framework of [6] and replaced some functions according to our own needs. We implemented our hash class, which accepts password and IP address as parameters and generates the hashed password. For convenience, here we used MD5 as the hash function.

D. Standalone Application

The standalone application is illustrated in Fig. 4. 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 (automatic URL detection feature is currently compatible with IE only) as shown in Fig. 5. 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. 6). After that the user can manually paste the hashed password to the relevant password field in the login page and log in.

E. Deployment

1) Installation - Client

The source codes of both plug-in and standalone prototypes can be downloaded freely from our server [20]. A user must have necessary permissions to install the plug-in prototype. 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. The user may carry around the standalone program in a USB key or download it from the download-sever (IP address instead of server name may be used to guard from DNS attacks targeted against download-server).

2) Installation - Server

Our solution requires modification on both server and client side. For server side, a module can be added to handle necessary generation and computation steps.

3) User Transition

When a user starts using the solution she does not need to update the passwords on her accounts but she needs to somehow remember which of her 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. To resolve this challenge, we propose two options:

Server registration: In this option, a list of all servers supporting PwdIP-Hash can be maintained; the standalone program/plug-in can periodically update their lists just like anti-virus software.

Bookmarking: In this option, whenever a user registers with a server (that supports PwdIP-Hash), the server prompts the user to bookmark the server with the PwdIP-Hash plug-in/standalone application. Later, the user can log in using these bookmarks. This option has portability issues since user's bookmarks will only be present on his/her computer. This can be resolved by using online bookmarking services or carrying the bookmark file with oneself.

4) Server Transition

It is not safe to assume that all the clients of a particular server will be using PwdIP-Hash from the moment the server implements the solution and starts accepting modified password for authentication. There will always be a transition period (and it may be long) during which some of the clients would still be using old authentication methods. We suggest that the server first checks the authentication using PwdIP-Hash and if it fails then it checks using the old authentication method for smooth transition. In case a user is authenticating using old method then additional security checks, such as asking security questions (as used by financial servers to authenticate users), should be added to strengthen authentication.

F. 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/installed. One 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 [6]. The solution though easy to implement but may become a single point of failure, especially if an attacker launches a fake online password hashing server.

The other 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 can be improved by incorporating additional security questions as previously suggested. 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 and should keep this in mind while designing security solutions.

IV. 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 [15]. To help readers to understand our user study, we have posted our user study questionnaires on our server [20].

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.

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.

Internet/computer security tutorial: Next a user was asked to go through a brief tutorial on Phishing and Pharming (based on the material from [18, 19]). 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.

Application trial: The trial consisted of four steps.

- *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.
- *Step 2:* Log in to the account, users were required to use the plug-in for filling up the password field.
- *Step 3:* Change the password and log out, changing password does not require the activation of application.
- *Step 4:* Again log in the server this time using new password. Users were asked to use the standalone version, this time, for login.

Post-trial questionnaire: The post-trial survey 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, Teaching, Music, etc) and the rest 48% were from technical disciplines (such as Computer Science, Engineering, Physics, Biology/Microbiology, etc).

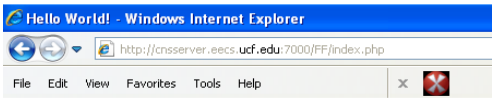

Figure 2. Inactive status of PwdIP-Hash browser plug-in

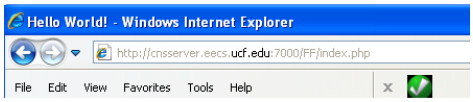

Figure 3. Active status of PwdIP-Hash browser plug-in

Figure 4. PwdIP-Hash standalone application

Figure 5. PwdIP-Hash detects current loaded IE pages and presents the corresponding URLs in its drop-down list

Figure 6. PwdIPHash copies hashed password to clipboard

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 having 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 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 users, 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 security awareness highlights the threat which people are facing from Phishing and Pharming attacks.

E. *Results*

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. Fig. 7 gives detailed account of application trial attempts.

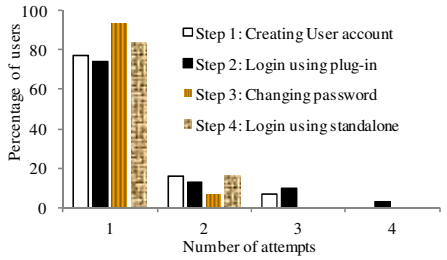

Figure 7. Number of attempts made by users to complete each step of application trial

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 show that our solution is user-friendly and practical. In addition, 56% preferred plug-in version over standalone version.

V. CONCLUSION

In this paper we have presented a lightweight solution that can effectively defend against both Phishing and Pharming 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.

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.

REFERENCES

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

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

[3] N. Chou, R. Ledesma, Y. Teraguchi, and J. Mitchell, "Client-side defense against web-based identity theft", In NDSS 2004.

[4] Z.-C. Chai, Z.-F. Cao, and R.-X. Lu, "Effcient password-based authentication and key exchange scheme preserving user privacy", In LNCS, 4138:467-477, 2006.

[5] 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.

[6] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, "Stronger password authentication using browser extensions", Usenix Security 05.

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

[8] 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 ICISS 2007.

[9] C. Karlof, U. Shankar, J.D. Tygar, D. Wagner, "Dynamic pharming attacks and locked same-origin policies for web browsers", In CCS07.

[10] J. A. Halderman, B.Waters, and E. Felten, "A convenient method for securely managing passwords", In WWW 2005.

[11] D. Jablon, "Strong password-only authenticated key exchange", ACM CCR, ACM SIGCOMM, vol. 26, no. 5, pp. 5-26, Oct. 1996.

[12] Q. Tang and C. J. Mitchell, "On the security of some password-based key agreement schemes", In CIS 2005.

[13] M. Zhang, "Analysis of the SPEKE password- authenticated key exchange protocol", IEEE Comm. Ltrs. v. 8, no.1, pp. 63-65, Jan. 2004.

[14] S. Chiasson and P.C. V. Oorschot, "A Usability Study and Critique of Two Password Managers", In USENIX Security 2006.

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

[16] E. Gabber, P. B. Gibbons, Y. Matias, and A. J. Mayer, "How to make personalized web browsing simple, secure, and anonymous", In Financial Cryptography, pages 17-32, 1997.

[17] D. Florencio, C. Herley, "A large-scale study of web password habits", In WWW 2007.

[18] http://phishguru.org/designs/all_phishguru_designs.pdf.

[19] http://www.SecurityCartoon.com.

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