PREVENTION OF ONLINE PASSWORD HACKING PROCESS WITH SECURED MULTI AUTHENTICATION SCHEME

Kunjesh Kumar Mishra (UG Scholar)[*1], Vimal Chand (UG Scholar)[2], G. Michael (Asst. Professor)[3]

[*1]Computer Science & Engineering/ Bharath University, Chennai, Tamil Nadu, India

kunjesh@live.com[*1]

[2]Computer Science & Engineering/ Bharath University, Chennai, Tamil Nadu, India

vimalchand23@gmail.com[2]

[3]Computer Science & Engineering/ Bharath University, Chennai, Tamil Nadu, India

micmgeo@yahoo.co.in[3]

Abstract-- Online Guessing attacks on Password Based Systems are inevitable and commonly observed against Web Applications. The Server Verifies User Name from the Cookie of the User's Machine, System IP, CAPTCHA, Password of the User, Number of Failure Attempts by the User and Web Browser that the User Uses for Browsing. This Process of Verification is called as Automated Turing Tests (ATT). In this paper the Authentication of User is done by asking Secret Questions which was answered during the Registration Phase.

Keywords: ATT, CAPTCHA, Multi-Authentication, SSH, Hacking, IP Address, SAN

1. Introduction

Online guessing attacks on password-based systems are inevitable and commonly observed against web applications and SSH logins. In nowadays report, SANS identified password guessing attacks on websites as a top cyber security risk. An example of SSH password-guessing attacks, one experiment Linux honey pot setup has reported that it suffered on average 2,810 SSH malicious login attempts per computer per day (see also). Interestingly, SSH servers that disallow standard password authentication may also suffer guessing attacks, e.g., through the soaking of a lesser known/used SSH server configuration called keyboard interactive authentication. However, online attacks have some implicit disadvantages compared to offline attacks: attacking machines must engage in an synergistic protocol, thus allowing easier detection; and in maximum cases, attackers can try only limited number of guesses from a single machine before being locked out, late, or challenged to answer Automated Turing Tests (ATTs, e.g., CAPTCHAs). Hence, attackers often must employ a large number of machines to avoid spotting or lock-out. On the other hand, as users generally choose common and relatively weak passwords (thus allowing effective password dictionaries), and attackers recently control large botnets (e.g., Conficker), online attacks are easier than before. One emphatic defense against automated online password guessing attacks is to restrict the number of failed trials without ATTs to a very small number (e.g., three), limiting automated programs (or bots) as used by attackers to three free password guesses for a targeted account, even if various machines from a botnet are used. However, this discommodes the legitimate user who then must answer an ATT on the next login attempt. Several other mechanisms are diffused in practice, including: permitting login attempts without ATTs from a different machine, when a few number of unsuccessful attempts occur from a given machine; allowing more attempts without ATTs after a timeout period; and time limited account locking. Many available mechanisms and proposals include ATTs, with the underlying assumption that these challenges are sufficiently difficult for bots and easy for many people. However, users increasingly dislike ATTs as these are perceived as an (unnecessary) extra step; see Yan and Ahmad [28] for usability issues related to commonly used CAPTCHAs. Due to well-turned attacks which break ATTs without human solvers, ATTs deemed to be more difficult for bots are being deployed. As a result of this arms-race, nowadays ATTs are becoming increasingly difficult for human users, fueling an expanding tension between security and usability of ATTs. Therefore, we focus on reducing user peeve by challenging users with fewer ATTs, while simultaneously subjecting bot logins to more ATTs, to drive up the economic expenditure to attackers.

Two well-known proposals for limiting online guessing attacks using ATTs are Pinakas and Sander (herein notify PS), and van Oorschot and Stubblebine (herein notify VS). The PS proposal reduces the number of ATTs sent to legal users, but at some meaningful loss of security; for instance, in an example setup (with $p = 0.05$, the fraction of incorrect login attempts requiring an ATT) PS allows attackers to eliminate 95% of the password space without answering any ATTs. The VS proposal reduces this but at a considerable cost to usability; for example, VS may need all users to answer ATTs in certain situations. The proposal in the present paper, is known as Password Guessing Resistant Protocol (PHOP), significantly improves the security-usability trade-off, and can be more generally diffused beyond browser-based authentication. PHOP builds on these 2 previous proposals. In particular, to bind attackers in under control of a large botnet (e.g., comprising hundreds of thousands of bots), PHOP enforces ATTs after a few (e.g., three) failed login attempts are made from unknown machines. On the other hand, PHOP permits a large number (e.g., 30) of failed attempts from known machines without replying any ATTs. We define known machines as those from which a successful login has occurred within a fixed term of time. These are identified by their IP addresses saved on the login server as a white-list, or cookies situated on client machines. A white-listed IP address and/or client cookie expires after a certain time. PHOP allows both graphical user interfaces (e.g., browser-based logins) and character-based interfaces (e.g., SSH logins), while the previous protocols deal exclusively with the former, searching for the use of browser cookies. PHOP allows either cookies or IP addresses, or both for tracking legal users. Tracking users through their IP addresses also allows PHOP to increase the number of ATTs for password guessing attacks and meanwhile to decrease the number of ATTs for legitimate login efforts. Although NATs and web proxies may (slightly) reduce the utility of IP address information, in rehearse, the use of IP addresses for client identification appears possible. In recent years, the trend of logging in to online accounts through multiple personal devices (e.g., PCs, laptops, smart-phones) is expanding. When used from a home environment, these devices often use a single public IP address (i.e., a simple NAT address) which makes IP-based history tracking more user-friendly with compare to cookies. For example, cookies must be stored, though transparently to the user, in every device used for login.

2. Related Work:

2.1 Module:

Although online password guessing attacks have been known since the early days of the Internet, there is little academic literature on prevention techniques. Account locking is a customary mechanism to prevent an adversary from attempting multiple passwords for a particular username. Although locking is generally temporary, the adversary can mount a DoS attack by making enough failed login attempts to lock a particular account. Delaying server response after receiving user credentials, whether the password is correct or incorrect, prevents the adversary from attempting a large number of passwords in a reasonable amount of time for a particular username. However, for adversaries with access to a large number of machines (e.g., a botnet), this mechanism is ineffective. Similarly, prevention techniques that rely on requesting the user machine to perform extra nontrivial computation prior to replying to the entered credentials are not effective with such adversaries. As discussed in Section 1, ATT challenges are used in some login protocols to prevent automated programs from brute force and dictionary attacks. Pinkas and Sander [17] presented a login protocol (PS protocol) based on ATTs to protect against online password guessing attacks. It reduces the number of ATTs that legitimate users must correctly answer so that a user with a valid browser cookie (indicating that the user has previously logged in successfully) will rarely be prompted to answer an ATT. A deterministic function (Ask ATT) of the entered user credentials is used to decide whether to ask the user an ATT. To improve the security of the PS protocol, van Oorschot and Stubblebine [23] suggested a modified protocol in which ATTs are always required once the number of failed login attempts for a particular username exceeds a threshold; other modifications were introduced to reduce the effects of cookie theft. For both PS and VS protocols, the decision function Ask ATT requires careful design. He and Han [9] pointed out that a poor design of this function may make the login protocol vulnerable to attacks such as the “known function attack” (e.g., if a simple cryptographic hash function of the username and the password is used as Ask ATT) and “changed password attack” (i.e., an adversary mounts a dictionary attack before and after a password change event initiated by a valid user). The authors proposed a secure nondeterministic keyed hash function as Ask ATT so that each username is associated with one key that should be changed whenever the corresponding password is changed. The proposed function requires extra server-side storage per username and at least one cryptographic hash operation per login attempt.

2.2 Module Description

2.2.1 USER REGISTRATION:

In the online accessing system we have to register the user with certain details for his future retrieval process. Without registering, a user can’t access the further details. For registering, the user should give the User name, CAPTCHA, and password. Once a user registered his details he can access the online facilities further. Each user will be identified by a unique username, password, System IP, Web browser which is stored in cookies.

2.2.2 SERVER:

A server is a computer program running to serve the requests of other programs, the "clients". Thus, the "server" performs some computational task on behalf of "clients". The clients either run on the same computer or connect through the network. Here the Server acts as the main resource for the client. Server is responsible for maintaining all the client information and further used for the Authentication process when the user reenters the online process.

2.2.3 COOKIES VERIFICATION:

The details got from the user during registration are stored in cookies and then sent to the server. When the user reenters the online process, for e.g.: banking process the cookie will verify for the details such as Username, System IP, CAPTCHA, Password, Web browser and the number of attempts of the user and then allows only the authorized user.

2.2.4 SYSTEM IP & CAPTCHA:

The System IP is nothing but the IP address of the System which was used by the user during online registration process. CAPTCHA is the verification text displayed inside the box. These two verifications will help the online processing to be more secure.

2.2.5 PASSWORD & WEB BROWSER:

The user should enter the password and the type of web browser during registration itself. So that it will be stored and it is used with other details for the authentication process. If the user needs to change the web browser or the System IP then he need to answer the secret Question.

2.2.6 VERIFICATION:

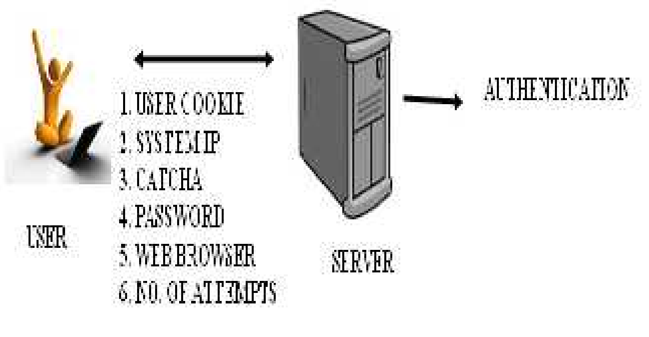
In this project, Verification is done by means of Cookies. Using cookies, the data already entered by the user is compared with the currently given data by the user. If the Username, System IP, CAPTCHA, Password and Web browser matches, then he will be allowed for the further processing or else access is denied. Among the above details if the user needs to change the System IP or Web browser then he needs to answer the secret question and CAPTCHA. After authentication, updating will be done.

3. Algorithm

3.1 Algorithm

```
        Input:
        $t_1$ (def=30d), $t_2$ (def=1d), $t_3$ (def=1d), $k_1$ (def=30), $k_2$ (def=3)
        // The keyword 'def' denotes the default parameter value and'd' denotes day, $k_1, k_2 \geq 0$
        Un, pw,cookie //username, password, and remote host's browser cookie if any
        W (global variable, expires after $t_1$)^1           //white list of IP addresses with successful login
        FT (global variable, def=0, expires after $t_2$)   //table of number of failed logins per username
        FS (global variable, def=0, expires after $t_3$)   //table of number of failed logins indexed by (srcIP, username) for hosts
                                                             in W or hosts with valid cookies
1  begin
2       ReadCredential(un, pw, cookie)      // login prompt to enter username/password pair
3       if LoginCorrect(un, pw) then        //username/ password pair is correct
4           if $(((\text{Valid}(cookie, un, k_1, \text{true}) \vee ((srcI\ P, un) \in W)) \wedge (FS[srcIP,un] < k_1)) \vee (FT[un] < k_2))$ then
5               $FS[srcIP,un] <= 0$
6               Add $scrIP$ to $W$               // add source IP address to the white list
7               GrantAccess($un, cookie$)   //  this function also sends the cookie if applicable
8           else
9               if ($ATTChallenge() = Pass$) then
10                  $FS[secIP,un] <= 0$
11                  Add $srcIP$ to $W$
12                  GrantAccess($un, cookie$)
13              else
14                  Message ('The answer to the ATT challenge is incorrect')
15      else                                          //username/password pair is applicable
16          if $((Valid(cookie, un, k_1, false) \vee ((srcIP,un) \in W)) \wedge (FS[srcIP,un] < k_1))$ then
17              $FS[srcIP,un] <= FS[srcIP,un] + 1$
18              Message('The username or password is incorrect')
19          else if $(ValidUsername(un) \wedge (FT[un] < k_2))$ then
20              $FT[un] <= FT[un] + 1$
21              Message('The username or password is incorrect')
22          else
23              if($ATTChallenge() = Pass$) then
24                  Message('The username or password is incorrect')
25              else
26                  Message('The to the ATT challenge is incorrect')
27 end
```

Algorithm. PHOP: Password Hacking Obstructive Protocol



3.1. Architectural Diagram:

The reports here generated vividly and the employee salary details & his transaction details can be seen through the reports.

The implementation of the proposed system is as follows:

1. Comparison of user name from the user with the user name in cookies of the user's machine.
2. User's system IP of login is verified with the system IP of the same user when registered.
3. User's CAPTCHA is also verified.
4. Password of the user is also verified.
5. Number of failure attempts by the user.
6. Web browser that the user uses for browsing.

4. Implementation and Experimental Setup

4.1. Implementation.

Implementation generally refers to the results and information that are generated by the system for many end-users; output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application. In the project, the admin details and employee details once are given. It stores in to the data base added.

4.2. Experimental Setup:

The database design is a must for any application developed especially more for the data store projects. Since the chatting method involves storing the message in the table and produced to the sender and receiver, proper handling of the table is a must. The salary and attendance table is common for all staff details. The different users view the data in different format according to the privileges given.

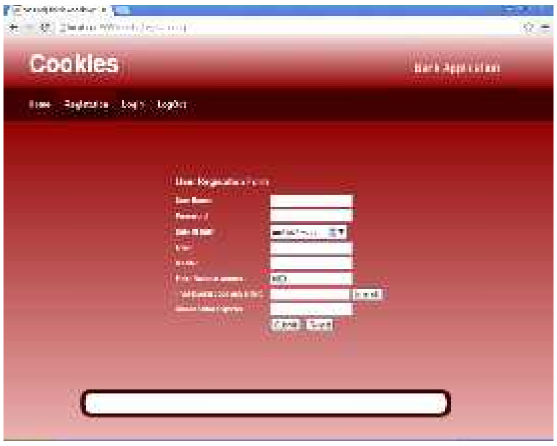


Figure 1. Registration Phase:
It shows the user's registration form. The user registers with the user's name password, date of birth, email, mobile, balance amount and the CAPTCHA generated in the flowing code.

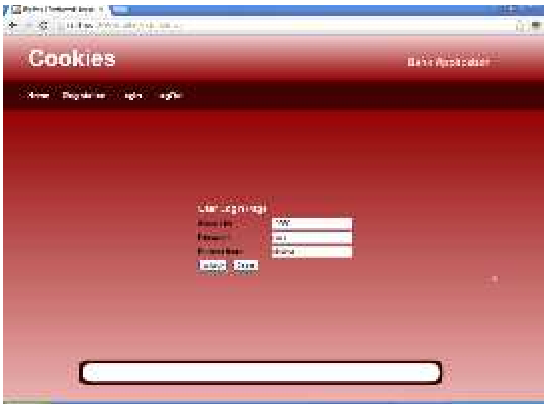


Figure 2. User's Login Phase:
It shows the user login after registering. After registering of the user the system generates the user account number. The user uses his/her account number and password which he/she created during the login phase. After clicking on submit button the user enter into its newly created account.

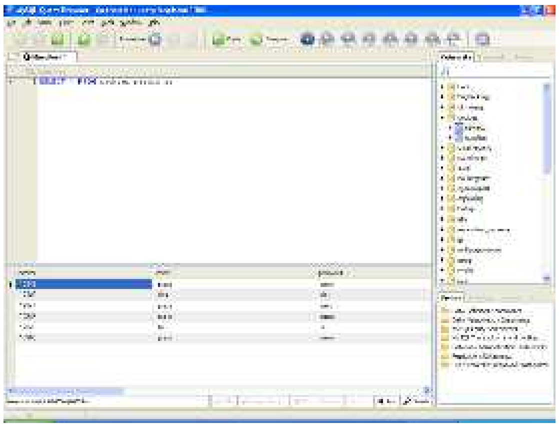


Figure 3. Database Connection:
It shows the database profile saved in the database connection. We can easily see the relevant information such as user name, password, account number, system IP, web browser used and answer of the security questions.

5. CONCLUSION

Online password guessing attacks on password-only systems have been observed for decades. Present-day attackers targeting such systems are empowered by having control of thousand to million node botnets. In former ATT-based login protocols, there exists a security-useful trade-off with respect to the number of free failed login attempts (i.e., with no ATTs) versus user login convenience (e.g., less ATTs and other requirements). In particular, PHOP is more restrictive against brute force and dictionary attacks while safely allowing a large number of free failed attempts for legitimate users. Our empirical experiments on two datasets (of one-year duration) gathered from operational network environments show that while PHOP is apparently more effective in preventing password guessing attacks (without answering ATT challenges), it also offers more appropriate login experience, e.g., a few number of ATT challenges for legitimate users even if no cookies are available. However, we again say that no user-testing of PHOP has been conducted so far.

REFERENCES

[1] Amazon Mechanical Turk. Accessed: June 2010. https://www.mturk.com/mturk/.

[2] S. M. Bellovin. A technique for counting natted hosts. In ACM SIGCOMM Workshop on Internet measurement, pages 267–272, New York, USA, 2002 ACM.

[3] E. Bursztein,, S. Bethard, J. C. Mitchell, D. Jurafsky, and C. Fabry. How good are humans at solving CAPTCHAs: A large scale evaluation? In IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 2010.

[4] M. Casado and M. J. Freedman. Peering through the shroud: The effect of edge opacity on ip-based client identification. In 4th USENIX Symposium on Networked Systems Design and Implementation (NDSS'07), 2007.

[5] S. Chiasson, P. C. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In USENIX Security Symposium, pages 1–16, Vancouver, B.C., Canada, 2006.

[6] D. Florˆencio, C. Herley, and B. Coskun. Do strong web passwords endow anything? In USENIX workshop on Hot topics in securities (Hot Sec'07), page 1–6, Berkeley, CA, United State, 2007.

[7] K. Fu, E. Sit, K. Smith, and N. Feamster. Dos and dont's of client authentication on the web. In USENIX Security Symposium, pages 251–268, Washington, DC, USA, 2001.