Defenses against Online Password Guessing Attacks with PGRP

Nikitha H S Sowmya D M Syeda Saher Anjum Varsha V
City Engineering College, Bangalore, India.
nsubramanya72@gmail.com sowmyamanohar45@gmail.com justsweety7@yahoo.com varshav2312@gmail.com

Abstract—*Brute force and dictionary attacks on password-only remote login services are now widespread and ever increasing. Enabling convenient login for legitimate users while preventing such attacks is a difficult problem. Automated Turing Tests (ATTs) continue to be an effective, easy-to-deploy approach to identify automated malicious login attempts with reasonable cost of inconvenience to users. In this paper, we discuss the inadequacy of existing and proposed login protocols designed to address large- scale online dictionary attacks (e.g., from a Botnet of hundreds of thousands of nodes). We proposes a new Password Guessing Resistant Protocol (PGRP), derived upon revisiting prior proposals designed to restrict such attacks. While PGRP limits the total number of login attempts from unknown remote hosts to as low as a single attempt per username, legitimate users in most cases (e.g., when attempts are made from known, frequently-used machines) can make several failed login attempts before being challenged with an ATT. We analyze the performance of PGRP with two real-world data sets and find it more promising than existing proposals.*

Index Terms—*Online password guessing attacks, brute force attacks, password dictionary, ATTs.*

1 INTRODUCTION

ONLINE guessing attacks on password-based systems are inevitable and commonly observed against web applications and SSH logins. In a recent report, SANS identified password guessing attacks on websites as a top cyber security risk. As an example of SSH password- guessing attacks, one experimental Linux honeypot setup has been reported to suffer on average 2,805 SSH malicious login attempts per computer per day . Interestingly, SSH servers that disallow standard password authentication may also suffer guessing attacks, e.g., through the exploitation of a lesser known/used SSH server configuration called keyboard interactive authentication .However, online attacks have some inherent disadvantages compared to offline attacks: attacking machines must engage in an interactive protocol, thus allowing easier detection; and in most cases, attackers can try only limited number of guesses from a single machine before being locked out, delayed, or challenged to answer Automated Turing Tests . Consequently, attackers often must employ a large number of machines to avoid

detection or lock-out. On the other hand, as users generally choose common and relatively weak passwords (thus allowing effective password dictionaries and attackers currently control large botnets online attacks are much easier than before.

2 RELATED WORKS

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 (AskAT T ðÞ) 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 *AskATT( )* 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.

3 PASSWORD GUESSING RESISTANT PROTOCOL

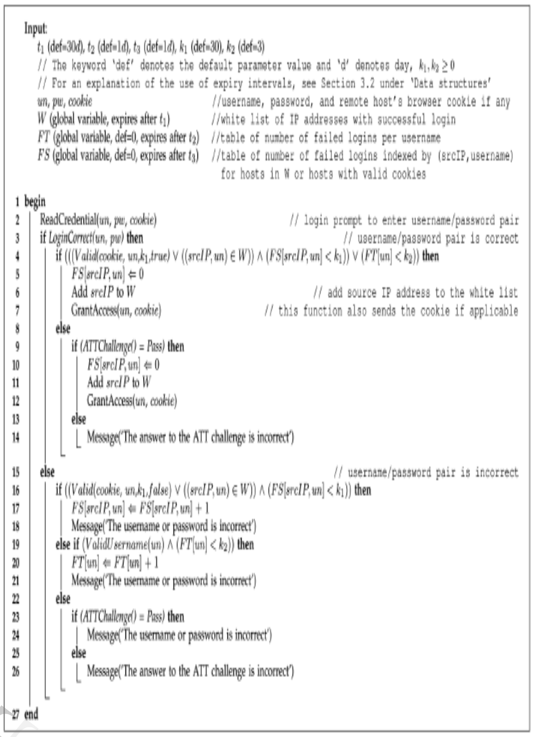
In this section, we present the PGRP protocol, including the goals and design choices.

3.1 Goals and Operation Assumptions

3.1.1 Protocol Goals

Our objective for PGRP include the following:

1. The login protocol should make brute force and dictionary attacks ineffective even for adversaries with access to large botnets (i.e., capable of launching the attack from many remote hosts).



```
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, $k_1, k_2 \geq 0$
    // For an explanation of the use of expiry intervals, see Section 3.2 under 'Data structures'
    un, pw, cookie                              //username, password, and remote host's browser cookie if any
    W (global variable, expires after $t_1$)          //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

begin
    ReadCredential(un, pw, cookie)              // login prompt to enter username/password pair
    if LoginCorrect(un, pw) then                // username/password pair is correct
        if $(((Valid(cookie, un, k_1, true) \vee ((srcIP, un) \in W)) \wedge (FS[srcIP, un] < k_1)) \vee (FT[un] < k_2))$ then
            $FS[srcIP, un] \Leftarrow 0$
            Add $srcIP$ to $W$                        // add source IP address to the white list
            GrantAccess(un, cookie)             // this function also sends the cookie if applicable
        else
            if (ATTChallenge() = Pass) then
                $FS[srcIP, un] \Leftarrow 0$
                Add $srcIP$ to $W$
                GrantAccess(un, cookie)
            else
                Message('The answer to the ATT challenge is incorrect')
    else                                        // username/password pair is incorrect
        if $((Valid(cookie, un, k_1, false) \vee ((srcIP, un) \in W)) \wedge (FS[srcIP, un] < k_1))$ then
            $FS[srcIP, un] \Leftarrow FS[srcIP, un] + 1$
            Message('The username or password is incorrect')
        else if $(ValidUsername(un) \wedge (FT[un] < k_2))$ then
            $FT[un] \Leftarrow FT[un] + 1$
            Message('The username or password is incorrect')
        else
            if (ATTChallenge() = Pass) then
                Message('The username or password is incorrect')
            else
                Message('The answer to the ATT challenge is incorrect')
end
```

Fig 1.PGRP Password Guessing Resistant Protocol

2. The protocol should not have any significant impact on usability (user convenience). For example: for legitimate users, any additional steps besides entering login credentials should be minimal. Increasing the security of the protocol must have minimal effect in decreasing the login usability.
3. The protocol should be easy to deploy and scalable, requiring minimum computational resources in terms of memory, processing time, and disk space.

3.1.2 Assumptions

We assume that adversaries can solve a small percentage of ATTs, e.g., through automated programs, brute force mechanisms, and low paid workers (e.g., Amazon Mechanical Turk [1]). Incidents of attackers using IP addresses of known machines and cookie theft for targeted password

Guessing is also assumed to be minimal. Traditional password-based authentication is not suitable for any entrusted environment (e.g., a key logger may record all keystrokes, including passwords in a system, and forward those to a remote attacker). We do not prevent existing such attacks in untrusted environments, and thus essentially assume any machines that legitimate users use for login are trustworthy. The data integrity of cookies must be protected (e.g., by a MAC using a key known only to the login server.

3.2 Data Structure and Function Description

3.2.1 Data Structures

PGRP maintains three Data Structures

1. W. A list of {source IP address, username} pairs such that for each pair, a successful login from the source IP address has been initiated for the username previously.
2. FT. Each entry in this table represents the number of failed login attempts for a valid username, UN. A maximum of K2 failed login attempts are recorded. Accessing a nonexistent index returns 0.
3. FS. Each entry in this table represents the number of failed login attempts for each pair of (srcI P, un). Here, srcIP is the IP address for a host in W or a host with a valid cookie, and un is a valid username attempted from srcI P . A maximum of k_1 failed login attempts are recorded; crossing this threshold may mandate passing an ATT (e.g., depending on FT½un). An entry is set to 0 after a successful login attempt. Accessing a nonexistent index returns0.

Each entry in W, FT, and FS has a “write-expiry” interval such that the entry is deleted when the given period of time (t_1 , t_2 , or t_3) has lapsed since the last time the entry was inserted or modified. There are different ways to implement write-expiry intervals (e.g., hashbelt [14]). A simple approach is to store a timestamp of the insertion time with each entry such that the timestamp is updated whenever the entry is modified. At anytime the entry is accessed, if the delta between the access time and the entry timestamp is greater than the data structure write-expiry interval (i.e., t_1 , t_2 , or t_3), the entry is deleted.

3.2.2 Functions

PGRP uses the following functions (IN denotes input and OUT denotes output):

1.ReadCredential(OUT:un,pw,cookie). Shows a login prompt to the user and returns the entered username and password, and the cookie received from the user’s browser (if any).

2.LoginCorrect(IN:un,pw;OUT:true/false). If the provided username-password pair is valid, the function returns true; otherwise, it returns false.

3.GrantAccess (IN: UN, cookie). The function sends the cookie to the user’s browser and then enables access to the specified user account.

4.M essage(IN: text). Shows a text message.

5.ATT Challenge (OUT: Pass/Fail). Challenges the user with an ATT and returns “Pass” if the answer is correct; otherwise, it returns “Fail.”

6.ValidUsername(IN:un;OUT:true/false). If the provided username exists in the login system, the function returns true; otherwise, it returns false.

7.Valid(IN:cookie,un,k_1,state;OUT:Cookie ,true/false).

First, the function checks the validity of the cookie (if any) where it is considered invalid in the following cases: 1) the login username does not match the cookie username; 2) the cookie is expired; or 3) the cookie counter is equal to or greater than k_1. The function returns true only when a valid cookie is received. If state ¼ true (i.e., the entered user credentials are correct, as in line 4 of Fig. 1), a new cookie is created (if cookies are supported in the login system) including the following information: user- name, expiry date, and a counter of the number of failed login attempts (since the last successful login; initialized to 0). Notice that if state ¼ true, the function does not send the created cookie to the user's browser. Rather, the cookie is sent later by the GrantAccessðÞ function. If state ¼ false (i.e., the entered user credentials are incorrect, as in line 16 of Fig. 1) And a valid cookie is received, the cookie counter is incremented by one and the cookie is sent back to the user's browser. No action is performed for all the other cases.

3.3 Decision Function for requesting ATTs

Below we discuss issues related to ATT challenges as provided by the login server in Fig.1.The decision to challenge the user with an ATT depends on two factors:

1) whether the user has authenticated successfully from the same machine previously; and
2) The total number of failed login attempts for a specific user account.

3.3.1 Username –Password pair is valid

As in the condition in line 4, upon entering a correct username-password pair, the user will not be asked to answer an ATT challenge in the following cases:

1. A valid cookie is received from the user machine (i.e., the function Valid returns true) and the number of failed login attempts from the user machine's IP address for that username, FS[½srcIP, un], is less than k_1 over a time period determined by t_3 ;
2. The user machine's IP address is in the whitelist W and the number of failed login attempts from this IP address for that username, FS½srcIP ; un , is less than k_1 over a time period determined by t_3 ;
3. The number of failed login attempts from any machine for that username, FT ½un , is below a threshold k_2 over a time period determined by t_2 .

The last case enables a user who tries to login from a new machine/IP address for the first time before K2 is reached to proceed without an ATT. However, if the number of failed login attempts for the username exceeds the threshold k_2 (default 3), this might indicate a guessing attack and hence the user must pass an ATT challenge.

3.3.2 Username-Password Pair is Invalid

Upon entering an incorrect username-password pair, the user will not be asked to answer an ATT challenge in the following cases:

1. A valid cookie is received from the user machine (i.e., the function Valid returns true) and the number of failed login attempts from the user machine's IP address for that username, FS½srcIP ; un , is less than k_1 (line 16) over a time period determined by t_3 ;
2. The user machine's IP address is in the whitelist W and the number of failed login attempts from this IP address for that username, FS½srcIP ; un , is less than k_1 (line 16) over a time period determined by t_3 ;
3. The username is valid and the number of failed login attempts (from any machine) for that username, FT ½un, is below a threshold K2 (line 19) over a time period determined by t_2.

A failed login attempt from a user with a valid cookie or in the whitelist W will not increase the total number of failed login attempts in the FT table since it is expected that legitimate users may potentially forget or mistype their password (line 16-18). Nevertheless, if the user machine is identified by a cookie, a corresponding counter of the failed login attempts in the cookie will be updated. In addition, the FS entry indexed by the {source IP address, username} pair will also be incremented (line 17). Once the cookie counter or the corresponding FS entry hits or exceeds the threshold k_1 (default value 30), the user must correctly answer an ATT challenge.

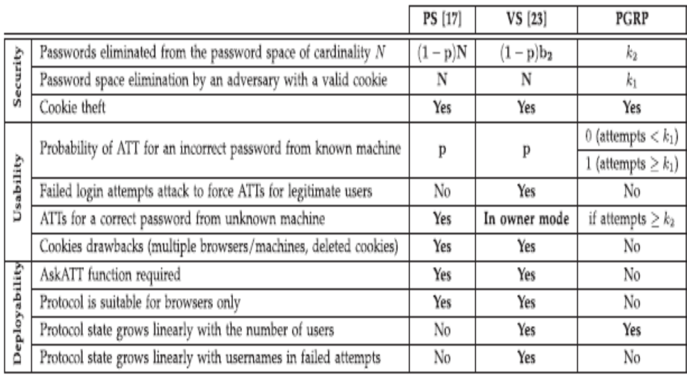
3.3.3 Output Messages

PGRP shows different messages in case of incorrect {username, password} pair (lines 21 and 24) and incorrect answer to the given ATT challenge (lines 14 and 26). While showing a human that the entered pair is incorrect, an automated program unwilling to answer the ATT challenge cannot confirm whether it is the pair or the ATT that was incorrect. However, while this is more convenient for legitimate users, it gives more information to the attacker about the answered ATTs. PGRP can be modified to display only one message in lines 14, 21, 24, and 26 (e.g., "login fails" as in the PS and VS protocols) to prevent such information leakage.

4 SYSTEM RESOURCES

In PGRP, three tables must be maintained. First the whitelist, W is expected to grow linearly with the number of users. At any given time, W contains a list of {source IP address, username} pairs that have been successfully authenticated in the last t_1 units of time. Second, the number of entries in FT increases by one whenever a remote host makes a failed login attempt using a valid username, if the username is not already in FT, and the remote host's IP address is not in W (or has no valid cookie). Therefore, unlike the VS protocol, the total number of valid usernames in the login server puts an upper bound on the number of entries in FT since a failed login attempt for a nonexistent username does not affect this table.

A new entry is added to FS only when a valid{username, password} pair is provided from an IP address not used before for this username. Therefore, the number of entries in FS is proportional to the number of IP addresses legitimate users successfully authenticated from. Increasing t_3 increases the number of entries in FS since the table entries last longer. The number of entries in FS is expected to be close to the number of active users within the last t_3 units of time.



		PS [17]	VS [23]	PGRP
Security	Passwords eliminated from the password space of cardinality N	$(1-p)N$	$(1-p)b_2$	k_2
	Password space elimination by an adversary with a valid cookie	N	N	k_1
	Cookie theft	**Yes**	**Yes**	**Yes**
Usability	Probability of ATT for an incorrect password from known machine	p	p	0 (attempts $< k_1$)
				1 (attempts $\geq k_1$)
	Failed login attempts attack to force ATTs for legitimate users	No	**Yes**	No
	ATTs for a correct password from unknown machine	**Yes**	**In owner mode**	if attempts $\geq k_2$
	Cookies drawbacks (multiple browsers/machines, deleted cookies)	**Yes**	**Yes**	No
Deployability	AskATT function required	**Yes**	**Yes**	No
	Protocol is suitable for browsers only	**Yes**	**Yes**	No
	Protocol state grows linearly with the number of users	No	**Yes**	**Yes**
	Protocol state grows linearly with usernames in failed attempts	No	**Yes**	No

TABLE 1
Comparison of Protocol limitations

5 CONCLUDING REMARKS

Online password guessing attacks on password-only systems have been observed for decades (see, e.g., [21]). Present day attackers targeting such systems are empowered by having control of thousand to million-node botnets. In previous ATT-based login protocols, there exists a security- usability 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 contrast, PGRP 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 data sets (of one-year duration) gathered from operational network environments show that while PGRP is apparently more effective in preventing password guessing attacks (without answering ATT challenges), it also offers more convenient login experience, e.g., fewer ATT challenges for legitimate users even if no cookies are available. However, we reiterate that no user testing of PGRP has been conducted so far.

PGRP appears suitable for organizations of both small and large number of user accounts. The required system resources (e.g., memory space) are linearly proportional to the number of users in a system. PGRP can also be used with remote login services where cookies are not applicable (e.g., SSH and FTP).

ACKNOWLEDGEMENTS

The second author is supported by an NSERC postdoctoral fellowship and by NSERC ISSNet. The third author is Canada Research Chair in Authentication and Computer Security and acknowledges NSERC for funding the chair, and a Discovery Grant. Partial funding from NSERC ISSNet is also acknowledged.

REFERNCES

[1] Amazon Mechanical Turk. https://www.mturk.com/mturk/, June 2010.
[2] S.M. Bellovin, "A Technique for Counting Natted Hosts," p ACM SIGCOMM Workshop Internet Measurement, pp. 267-272,2002
[3] M. Casado and M.J. Freedman, "Peering through the Shroud: The Effect of Edge Opacity on Ip-Based Client Identification," Proc. Fourth USENIX Symp. Networked Systems Design and Implementation (NDSS '07), 2007.
[4] S. Chiasson, P.C. van Oorschot, and R. Biddle, "A Usability Study and Critique of Two Password Managers," Proc. USENIX Security Symp., pp. 1-16, 2006.
[5] D. Florencio, C. Herley, and B. Coskun, "Do Strong Web Passwords Accomplish Anything?," Proc. USENIX Workshop Hot Topics in Security (HotSec '07), pp. 1-6, 2007.
[6] K. Fu, E. Sit, K. Smith, and N. Feamster, "Dos and Don'ts of Client Authentication on the Web," Proc. USENIX Security Symp., pp. 251-268, 2001.
[7] P. Hansteen, "Rickrolled? Get Ready for the Hail Mary Cloud!," http://bsdly.blogspot.com/2009/11/rickrolled-get-ready-for-hail-mary.html, Feb. 2010.
[8] Y. He and Z. Han, "User Authentication with Provable Security against Online Dictionary Attacks," J. Networks, vol. 4, no. 3, pp. 200-207, May 2009.
[9] T. Kohno, A. Broido, and K.C. Claffy, "Remote Physical Device Fingerprinting," Proc. IEEE Symp. Security and Privacy, pp. 211-225,2005.
[10] M. Motoyama, K. Levchenko, C. Kanich, D. Mccoy, G.M. Voelker, and S. Savage, "Re: CAPTCHAs Understanding CAPTCHA-Solving Services in an Economic Context," Proc. USENIX Security Symp., Aug. 2010.
[11] C. Namprempre and M.N. Dailey, "Mitigating Dictionary Attacks with Text-Graphics Character Captchas," IEICE Trans. Fundamentals of Electronics, Comm. and Computer Sciences, vol. E90-A, no. 1, pp. 179-186, 2007.
[12] Nat'l Inst. of Standards and Technology (NIST), Hashbelt.

http://www.itl.nist.gov/div897/sqg/dads/HTML/hashbelt.html,