Revisiting Defenses Against Large-Scale Online Password Guessing Attacks*

Mansour Alsaleh
School of Computer Science
Carleton University, Canada

Mohammad Mannan
Edward S. Rogers Department of
Electrical and Computer Engineering
University of Toronto

P.C. van Oorschot
School of Computer Science
Carleton University, Canada

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 propose a new protocol called Password Guessing Resistant Protocol (*PGRP*), 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 datasets and find it more promising than existing proposals.

Keywords: online password guessing attacks, brute force attacks, dictionary attacks, automated turing tests.

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 [19] 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 [17] to suffer on average 2,805 SSH malicious login attempts per computer per day (see also [7]). Interestingly, SSH servers that disallow standard password authentication may also suffer guessing attacks, as identified by a recent report [18].[1] 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 (ATTs, e.g., CAPTCHAs [24]). Consequently, attackers 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 [12]), and attackers currently control large botnets (e.g., Conficker [14]), online attacks are much easier than before.

One effective 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 different machines from a botnet are used. However, this inconveniences the legitimate user who then must answer an ATT on the next login attempt.

Several other techniques are deployed in practice, including: allowing login attempts without ATTs from a different machine, when a certain number of failed attempts occur from a given machine; allowing more attempts without ATTs after a timeout period; and time-limited account locking. Many exist-

*Version: Sept. 6, 2010. Contact author: `malsaleh@scs.carleton.ca`

[1]The attacks exploit a lesser known/used SSH server configuration called *keyboard interactive* authentication.

ing techniques and proposals involve ATTs, with the underlying assumption that these challenges are sufficiently difficult for bots and easy for most people. However, users increasingly dislike ATTs as these are perceived as an (unnecessary) extra step; see Yan and Ahmad [27] for usability issues related to commonly used CAPTCHAs. Due to many successful attacks on breaking ATTs without human solvers (e.g., [26, 22]), ATTs perceived to be more difficult for bots are being deployed. As a consequence of this arms-race, present-day ATTs are becoming increasingly difficult for human users [3]. There appears to be a growing tension between security and usability of ATTs. Therefore, we focus on reducing user annoyance by challenging them with fewer ATTs, while at the same time subjecting bot logins to more ATTs, to drive up the economic cost to attackers [10].

Two well-known proposals for limiting online guessing attacks using ATTs are Pinkas and Sander [16] (herein denoted PS), and van Oorschot and Stubblebine [23] (herein denoted VS). For convenience, a review of these protocols is given in Appendix A. The PS proposal reduces the number of ATTs sent to legitimate users, but at some meaningful loss of security; for example, in an example setup (with $p = 0.05$, the fraction of 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 significant cost to usability; for example, VS may require all users to answer ATTs in the worst case (see Appendix A). The proposal in the present paper, called Password Guessing Resistant Protocol (*PGRP*), significantly improves the security-usability trade-off, and can be more generally deployed beyond browser-based authentication.

PGRP builds on these two previous proposals. In particular, to limit attackers in control of a large botnet (e.g., comprising hundreds of thousands of bots), PGRP enforces ATTs after a few (e.g., three) failed login attempts are made from *unknown* machines. On the other hand, PGRP allows a high number (e.g., 30) of failed attempts from *known* machines. We define known machines as those from which a successful login has occurred within a fixed period of time. These are identified by their IP addresses saved on the authentication server as a white-list, or (as in PS [16]) cookies stored on client machines. A white-listed IP address and/or client cookie expire after a certain time.

PGRP is designed for 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 (requiring the use of browser cookies). PGRP uses either cookies or IP addresses, or both for tracking legitimate users. In recent years, the trend of logging in to online accounts through multiple personal devices (e.g., PCs, laptops, smart-phones) is growing. When used from a home environment, these devices often share a single public IP address which makes IP-based history tracking more user-friendly than cookies (e.g., cookies must be stored, albeit transparently to the user, in all devices used for login).

Contributions. Our contributions include:

1. Strict but User-friendly ATT-based Scheme: The proposed PGRP scheme is more restrictive against attackers than commonly used counter-measures and two earlier proposals [16, 23]. At the same time, PGRP requires answering fewer ATTs for all legitimate users, including those who occasionally require multiple attempts to recall a password.

2. First Reported Empirical Analysis of ATT-based Schemes: We compare PGRP's performance and usability (e.g., the number of ATTs triggered, ATTs sent to legitimate users) to previous such schemes, using two datasets from a university environment (SSH and web-email login data, covering more than a year).

3. Applicability to Web and Text Logins: PGRP is not limited to web-only login (unlike proposals solely relying on browser cookies), as it uses IP address and/or other methods to identify a remote machine in addition to optionally using cookies. By using text-based ATTs (e.g., [21]), SSH login can be adapted to use PGRP.

Organization. Section 2 discusses related work on prevention techniques for online dictionary attacks. Section 3 presents the PGRP login protocol. Section 4 compares PGRP with other ATT-based protocols in terms of security (Section 4.1), usability (Section 4.2), and required computational resources (Section 4.3). A summary of limitations comparing these protocols is given in Section 4.4. In Section 5, we evaluate PGRP and other ATT-based protocols on two different remote login datasets and analyze the results. Section 6 concludes. Appendix A provides a review of the PS and VS protocols.

2 Related Work

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 [16] 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 ($AskATT()$) 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 [8] 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 $AskATT()$) 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 non-deterministic keyed hash function as $AskATT()$ 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 (PGRP)

In this section, we present the PGRP protocol, based on the concept of using ATT challenges [16].

3.1 Goals, Operational Assumptions and Overview

Protocol goals and assumptions. Our objectives 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).

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.

Assumptions. We assume that adversaries can solve a small percentage of ATTs, e.g., through automated programs, brute force mechanisms, or 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 are also assumed to be minimal. Traditional password-based authentication is not suitable for any untrusted environment (e.g., a keylogger 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 [16]).

Protocol overview. The general idea behind PGRP (see Algorithm 1) is that except for the following two cases, all remote hosts must correctly answer an ATT challenge prior to being informed whether access is granted or the login attempt is unsuccessful: (i) when the number of failed login attempts for the username is very small; and (ii) when the remote host has successfully logged into the same username in the past (however, such a host must pass an ATT challenge if it generates more failed login attempts than a pre-specified threshold).

In contrast to previous protocols, PGRP uses either IP addresses, cookies, or both to identify machines from which users have been successfully authenticated. The decision to require an ATT challenge upon receiving incorrect credentials is based on the received cookie (if any) and/or the remote host's IP address. In addition, if the number of failed login attempts for a specific username is below a threshold, the user is not required to answer an ATT challenge even if the login attempt is from a new machine for the first time (whether the provided username-password pair is correct or incorrect). Section 3.2 below discusses these differences in further detail.

Data structures. PGRP maintains three data structures:

1. W: A list of IP addresses from which a successful login has occurred in the last t_1 units of time.

2. FT: Each entry in this table represents the number of failed login attempts for a username, un (only valid usernames). A maximum of k_2 failed login attempts are recorded. Accessing a non-existing index returns 0.

3. FS: Each entry in this table represents the number of failed login attempts for each pair of ($srcIP$, un). Here, $srcIP$ is the IP address for a host in W or a host with a valid cookie, and un is the username attempted from $srcIP$ (un could be only valid usernames). A maximum of k_1 failed login attempts are recorded; crossing this threshold may mandate passing an ATT (e.g., depending on $FT[un]$). An entry is set to 0 after a successful login attempt. Accessing a non-existing index returns 0.

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., the hashbelt data structure [13]). 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 (i.e., the entry timestamp is the most recent modification time). 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 Cookies vs. Source IP Addresses

Similar to the previous protocols, PGRP keeps track of user machines from which successful logins have been initiated previously. Browser cookies seem a good choice for this purpose if the login server offers a web-based interface. Typically, if no cookie is sent by the user browser to the login server, the server sends a cookie to the browser after a successful login to identify the user on the next login attempt. However, if the user uses multiple browsers or more than one OS on the same machine, the login server will be unable to identify the user in all cases. Cookies may also be deleted by users, or automatically as enabled by the private browsing mode of most modern browsers. Moreover, cookie theft (e.g., through session hijacking) might enable an adversary to impersonate a user who has been successfully authenticated in the past [6]. In addition, using cookies requires a browser interface (which, e.g., is not applicable to SSH).

Alternatively, a user machine can be identified by the source IP address. Relying on source IP addresses to trace users may result in inaccurate identification for various reasons, including: (i) the same machine might be assigned different IP addresses over time (e.g., through the network DHCP server and dial-up Internet); and (ii) a group of local machines might be represented by a smaller number or even a single Internet-addressable IP address if a NAT mechanism is in place (techniques to identify machines behind a NAT exist, e.g., [2, 9]).

Drawbacks of identifying a user by means of either a browser cookie or a source IP address include: (i) failing to identify a machine from which the user has authenticated successfully in the past; and (ii) wrongly identifying a machine the user has not authenticated before. Case (i) decreases usability since the user might be asked to answer an ATT challenge for both correct and incorrect login credentials. On the other hand, case (ii) affects security since some

```
Input:
    $t_1$ (def=30$d$), $t_2$ (def=1$d$), $t_3$ (def=1$d$), $k_1$ (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($un$, $pw$)                       // login prompt to enter username/password pair
3      if $LoginCorrect(un, pw)$ then                   // username/password pair is correct
4          if $((Valid(cookie,un,k_1,true) \vee (srcIP, un) \in W) \wedge FS[srcIP, un] < k_1) \vee FT[un] < k_2$ then
5              $FS[srcIP, un] \Leftarrow 0$
6              Add $srcIP$ to $W$                       // add source IP address to the white list
7              GrantAccess()                            // this function also adds/updates the cookie if applicable
8          else
9              if $(ATTChallenge() = Pass)$ then
10                 $FS[srcIP, un] \Leftarrow 0$
11                 Add $srcIP$ to $W$
12                 GrantAccess()
13             else
14                 Message(‘The answer to the ATT challenge is incorrect’)
15     else                                             // username/password pair is incorrect
16         if $((Valid(cookie,un,k_1,false) \vee (srcIP, un) \in W) \wedge FS[srcIP, un] < k_1)$ then
17             $FS[srcIP, un] \Leftarrow FS[srcIP, un] + 1$
18             Message(‘The username or password is incorrect’)
19         else if $(ValidUsername(un)) \wedge (FT[un] < k_2)$ then
20             $FT[un] \Leftarrow FT[un] + 1$
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 answer to the ATT challenge is incorrect’)
27 end
```

Algorithm 1: PGRP: Password Guessing Resistant Protocol

users/attackers may not be asked to answer an ATT challenge even though they have not logged in successfully from those machines in the past. However, the probability of launching a dictionary or brute force attack from these machines appears to be low. First, for identification through cookies, a directed attack to steal users’ cookies is required by an adversary. Second, for identification through IP addresses, the adversary must have access to a machine in the same user NAT-based subnet to share the same IP address.

Consequently, we choose to use both browser cookies and source IP address (or only one of them if the other is not applicable) in PGRP to minimize user inconvenience during the login process. Also, by using IP addresses only, PGRP can be used in character-based login interfaces such as SSH. An SSH server can be adapted to use PGRP using text-based ATTs (e.g., [21]). For example, a prototype of a text-based CAPTCHA for SSH is available as a source code patch for OpenSSH [11].

The security implications of mistakenly treating a machine as one that a user has previously successfully logged in from is limited by a threshold such that after a specific number of failed login attempts (k_1 in Algorithm 1), an ATT challenge is imposed. For identification through a source IP address, the condition $FS[srcIP, un] < k_1$ in line 4 (for correct credentials) and in line 16 (for incorrect credentials) limits the number of failed login attempts an iden-

[1]For an explanation of the use of expiry intervals, see Section 3.1 under “Data structures”.

tified user can make without answering ATTs (see Algorithm 1). Also, as explained further below, the function $Valid(cookie, un, k_1, true)$ in line 4 updates a counter in the received cookie in which the cookie is considered invalid once this counter hits or exceeds k_1. Similarly, the same function is also called in line 16 to check this counter in case of a failed login attempt.

3.3 Decision Function for Requesting ATTs

For PGRP, the decision to challenge the user with an ATT depends on two factors: (i) whether the user has authenticated successfully from the same machine previously; and (ii) the total number of failed login attempts for a specific user account. Below we discuss issues related to ATT challenges as provided by the login server in Algorithm 1. For definitions of W, FT, and FS, see "Data structures" in Section 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, $FS[srcIP, un]$, 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, $FS[srcIP, un]$, is less than k_1 over a time period determined by t_3; or
3. the number of failed login attempts (from any machine) for that username, $FT[un]$, is below a threshold k_2 over a time period determined by t_2.

The last case enables a user who tries to log in from a new machine/IP address for the first time before k_2 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.

If no cookie is received from the user's browser and the entered user credentials are correct, the function $Valid$ in line 4 sends a new cookie to the user's browser (assuming cookies are applicable) that includes the following information: username, expiry date, and a counter of the number of failed login attempts made (since last successful login; initial value is set to 0) using this cookie. The function returns false in this case.

If a cookie is received from the user's browser, the function $Valid$ in lines 4 and 16 checks the validity of the cookie, and returns false (i.e., the cookie is considered invalid) only in the following cases:

1. the login attempt username (un as in Algorithm 1) does not match the cookie username;
2. the cookie is expired;
3. the cookie counter is equal to or greater than k_1.

This function also updates the cookie such that the counter of the number of failed login attempts is set to 0 or incremented according to the function's last input parameter which indicates whether the entered user credentials are correct or not (the cookie expiry date is also updated).

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, $FS[srcIP, un]$, 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, $FS[srcIP, un]$, is less than k_1 (line 16) over a time period determined by t_3; or
3. the username is valid and the number of failed login attempts (from any machine) for that username, $FT[un]$, is below a threshold k_2 (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 as explained

	Basic Protocol [16]	**PS [16]**	**VS [23]**		**PGRP Protocol**
			Owner	**Non-owner**	
Q1	0	$N - pN$	$(1-p)b_2$	$max(b_1, (1-p)b_2)$	k_2
Q2	$\frac{1}{2}N$	$\frac{1}{2}pN$	$\frac{1}{2}(N-(1-p)b_2) \approx \frac{1}{2}N$	$\frac{1}{2}(N - max(b_1, (1-p)b_2))$	$\frac{1}{2}(N-k_2)$
Q3	0	0	0	b_1/N	k_2/N
Q4	c/N	c/pN	$\leq min(\frac{c}{p}, b_2 + c)/N$	$(b_1 + c)/N$	$(k_2 + c)/N$

Table 1: Comparative security analysis for single-account attacks. (Consider $k_2 = 3$, $p = 0.05$, $b_1 = 5$, and $b_2 = 5$, for concreteness; see Appendix A for a review of the PS and VS algorithms)

above. 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.

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 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.

Why not to black-list offending IP addresses. We choose not to create a blacklist for IP addresses making many failed login attempts for the following reasons: (i) this list may consume considerable memory; (ii) legitimate users from blacklisted IP addresses could be blocked (e.g., using compromised machines); and (iii) hosts using dynamic IP addresses seem more attractive targets for adversaries to launch their attacks from (e.g., spammers [25]).

If the cookie mechanism is not available for the login server, PGRP can operate by using only source IP addresses to keep track of user machines. Security and usability implications in this case are discussed in Section 4.

4 Comparison with other ATT-based Protocols

In this section, we analyze the security, usability, and required system resources of PGRP as compared to a basic protocol and the PS and VS protocols (see Algorithm 2, 3, and 4 in Appendix A for a review of these protocols). This section also provides a comparative summary of major limitations in each protocol.

4.1 Security Analysis

Following the previous analysis of PS [16], assume a fixed password space of cardinality N, assume passwords are equi-probable, and that the delay between when the {username, password} pair is entered and the ATT challenge is presented to the user is identical whether or not the credentials are correct. Also assume that cookie theft, and adversaries using legitimate users' IP addresses[1] occur rarely.

4.1.1 Single-Account Attacks

In a single account attack, a specific user account is targeted. Following the security analysis of VS [23] in this case, we consider the following questions:

- Q1: What is the expected number of passwords that an adversary can eliminate from the password space without answering any ATT challenge?
- Q2: What is the expected number of ATT challenges an adversary must answer to correctly guess a password?
- Q3: What is the probability of a confirmed correct guess for an adversary unwilling to answer any ATT?
- Q4: What is the probability of a confirmed correct guess for an adversary willing to answer c ATTs?

Table 1 compares PGRP with the PS and VS protocols. For simplicity, we use only the case $c \geq 2$ in Q4 for the VS protocol. The answer to Q1 depends on the threshold k_2. The adversary can eliminate only k_2 passwords without answering ATTs. Likewise, for

[1]For example, in case of dynamic IP addresses, an attacker machine may be assigned an IP address previously used by a targeted user's machine.

	Basic Protocol [16]	**PS [16]**	**VS [23]**		**PGRP Protocol**
			Owner	**Non-owner**	
Q1	0	0	0	mb_1/N	mk_2/N
Q2	c/N	c/pN	$\leq min(\frac{c}{p}, b_2 + c)/N$	$(mb_1 + c)/N$	$(mk_2 + c)/N$

Table 2: Comparative security analysis for multi-account attacks

Q2, the expected number of ATTs the adversary must answer to correctly guess a password is one-half of the remaining passwords of the password space after subtracting the number of login attempts that do not require ATTs. Using a small value for k_2 yields $\frac{1}{2}(N - k_2) \approx \frac{1}{2}N$. For Q3, given that k_2 is intended to be small (e.g., 3), the probability of guessing a password for a single-account attack without answering any ATT is very small (e.g, for 8-char case-sensitive alphabetical passwords chosen randomly, $N = 52^8$ and p(guessing the right password) $= 2/52^8$). For Q4, the adversary has only k_2 free attempts after which ATTs must be answered. Therefore, he can guess a total of $k_2 + c$ passwords with a probability of $(k_2 + c)/N$ to find a correct password.

This analysis shows that PGRP provides improved security over PS and VS with respect to all four questions, and identical security compared to the basic protocol for $k_2 = 0$.

4.1.2 Multi-Account Attacks

In contrast to a directed attack on a single account, an adversary could attempt to break into multiple accounts at the same time. In fact, this is the current trend of brute force and dictionary attacks [19]. In this case, the adversary usually has access to a large number of machines (e.g., compromised machines in a botnet) and initiates the attack from many sources at the same time. This typically gives the adversary a greater chance of compromising user accounts than targeting a single account.

We compare previous protocols and PGRP by answering the following questions in Table 2:

- Q1: What is the probability that an adversary knowing m usernames can correctly guess a password without answering any ATT challenge?
- Q2: What is the probability of a confirmed correct guess for an adversary knowing m usernames and willing to answer c ATTs?

Considering Q1 in Table 2, PS appears more secure for multi-account attacks than VS and PGRP. However, it may be unrealistic to assume that an adversary with access to a large botnet is unable to break a small percentage of ATTs [26, 22] (which leads us to Q2).

For Q2, the probability in PGRP depends on the number of usernames the adversary knows and k_2. While PGRP is comparable to the VS protocol in multi-account attacks, PS seems slightly better than PGRP but only for login systems with a large number of users as in equation (1).

$$\begin{aligned} \frac{m \cdot k_2 + c}{N} &> \frac{c}{p \cdot N} \\ m &> \frac{c}{k_2}(\frac{1}{p} - 1) \end{aligned} \quad (1)$$

To consider a concrete example, for any password length, $k_2 = 3$, $p = 0.05$, and an adversary willing to answer $c = 2^{20}$ ATTs: $m > 1/3$ $(2^{20}/0.05 - 2^{20})$; i.e., when $m > 2^{22}$ users, PS is better than PGRP in Q2.

In the PGRP protocol, an adversary may be able to guess a subset of the valid usernames which is undesirable in certain cases [5]. In line 19 of Algorithm 1, the FT list is not updated if the username is invalid, thus an ATT will be requested for each login attempt with an invalid username. Therefore, the adversary could generate a list of valid usernames as follows: if an attempted username requires an ATT for the first login attempt, the username is considered invalid; otherwise, the username is valid. However, the adversary will overlook valid usernames that have at least k_2 failed attempts. While the condition $ValidUsername(un)$ in line 19 can be omitted to overcome this drawback, the number of entries in the list FT will be now proportional to the number of all attempted usernames (whether valid or invalid) by users/attackers within a time period determined by t_2 (see Section 3.1 under "Data structures"). We choose to keep the condition $ValidUsername(un)$ in line 19 to restrict the maximum size of FT to the number of valid usernames, even when guessing attacks involving a large number of usernames (both valid and invalid) are launched.

4.2 Usability Comments (Number of ATT Challenges)

Our main security goal is to restrict an attacker who is in control of a large botnet from launching online single-account or multi-account password dictionary attacks. In terms of usability, we want to reduce the number of ATTs sent to legitimate users as much as possible. A user receives ATTs when the total number of failed attempts exceeds threshold k_2, and the login attempt is initiated from (i) an unknown machine (i.e., no valid cookies or whitelisted IP addresses), or (ii) a known machine from which the user has already failed k_1 times. This happens for both cases of correct and incorrect username-password pairs, assuming the provided username is valid. Below we discuss different login scenarios and the extra effort as required from users by PGRP. The analysis below indicates that only limited usability impact may be expected from our proposal; the same can also be inferred from our real-world data analysis, e.g., the number of ATTs sent to legitimate users (see Section 5). However, we have not yet carried out any formal user testing. For notation and parameters as used in the following, see Algorithm 1. For definitions of W, FT, and FS, see "Data structures" in Section 3.1.

First time login from an unknown machine. If a valid username-password pair is provided from an unknown machine (i.e., one from which no successful login has occurred within a designated period), no ATTs are required if the total fail count from unknown machines is below k_2 (within a time period determined by t_2). This threshold may be exceeded as follows: (i) the user may provide incorrect passwords from that machine k_2 times; (ii) attackers may have attempted k_2 failed passwords (from unknown machines); or (iii) a combination of (i) and (ii). Once a user successfully logs in, the machine's IP address is added to the known list (W).

Subsequent login from a known machine. ATTs are sent to a known machine (i.e., one from which a successful login has occurred within a designated period) only when k_1 is hit or crossed (see line 4 in Algorithm 1) for that machine and the user account is possibly under attack (i.e., k_2 failed attempts also occurred on the account's username from unknown machines). By setting k_1 to be relatively large (e.g., $k_1 = 30$), legitimate users may make a reasonable number of password mistakes without experiencing any ATTs.

Valid password is provided. Users may be understandably annoyed if they provide a valid password, and yet are asked to answer an ATT. When a valid password is provided by the user, no ATT challenges are sent if the attempt comes from a known machine which has not been used for more than $k_1 - 1$ failed login attempts within a time period determined by t_3. If the user hits or crosses the threshold k_1, still no ATTs are sent if the number of failed login attempts from unknown machines remains below k_2. Thus, users must pass ATT challenges only when they attempt login from unknown machines and the number of failed attempts from unknown machines has hit or crossed k_2 (possibly due to an ongoing attack). We believe this is not a common occurrence, as was apparent from our collected data.

Invalid password. This may be a common occurrence for several reasons: (i) if users need multiple attempts to recall the correct password; (ii) if users cycle-through multiple passwords due to multi-password interference [4]; and (iii) typing errors including activating the caps-lock key, sometimes aggravated by on-screen masking of password characters (see e.g., Nielsen's blog [15]). From each known machine, a user is allowed up to k_1 attempts, before challenged with ATTs; i.e., if the user has logged in from n known machines (within a time period determined by t_3), then in total $n \cdot k_1 + k_2$ attempts are allowed without ATTs. While high values of k_1 (30 by default) provide convenient login for legitimate users in common use-cases, we do not recommend very high values (e.g., $k_1 = 10000$) as that may aid guessing attacks when a cookie is stolen or a dynamic whitelisted IP address is assigned to an attacker's machine (i.e., a bot). Note that in VS [23], an adversary can make b_2 failed connection attempts for all (or as many as possible) users of system, with the result that any failed login attempt from a legitimate user will face an ATT challenge. In PGRP, user convenience is unaffected by an attacker's actions, as long as there are not more than $k_1 - 1$ unsuccessful login attempts from known machines.

Invalid username. When a user tries login with an invalid (i.e., non-existent) username (e.g., typing errors), an ATT challenge is given. Irrespective of the password or ATT answer, the login fails. This feature restricts attackers from learning valid usernames (except the usernames obtained via brute force attacks as explained in Section 4.1.2), and improves protocol performance in terms of memory usage (i.e., no entries in protocol data structures W, FT, or FS).

		PS [16]	VS [23]	PGRP
Security	Passwords eliminated from the password space of cardinality N	$\mathbf{(1-p)N}$	$\mathbf{(1-p)b_2}$	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 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 3: Comparison of protocol limitations (limitations are in bold face)

However, from a usability point of view, this is not ideal. We expect that this type of error would be limited in practice (in part because usernames, in contrast to passwords, are echoed on a display).

4.3 System Resources

No lists are maintained in the PS protocol (see Algorithm 3), thus no extra memory overhead is imposed on the login server. In the VS protocol (see Algorithm 4), only FT is maintained. The number of entries in this list grows linearly with unique usernames (both valid and invalid) used in failed login attempts. An attacker may try to exhaust a login server's memory by failed login attempts for many usernames. For any cookie-based login protocol, the login server may also need to store information regarding each generated cookie to ameliorate cookie theft attacks [23]. Note that neither the PS nor VS protocol uses IP addresses. The most expensive server operation in PS, VS, and PGRP is generating an ATT.

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 non-existing 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 (as also shown in the analysis of two real-world datasets in Section 5).

4.4 Limitations

Table 3 summarizes major shortcomings in the PS, VS, and PGRP protocols. Under each protocol, the text is highlighted in bold face if the corresponding entry is a limitation. The first limitation in the security row represents Q1 as discussed in Section 4.1.1. The second limitation is about password space elimination for an adversary with a valid cookie or who can use an IP address from which a username has been successfully authenticated in the past. Neither the PS nor VS protocol restricts the number of failed login attempts for such adversaries. Cookie theft is a possible attack that can be mounted against all these protocols, but can be mitigated by updating a counter in the cookie for the maximum number of failed login attempts [16]. The VS protocol stores cookies only on trustworthy machines (as discussed in Appendix A), to reduce exposure to cookie theft.

As outlined in Table 3, the first limitation in the usability row is about subsequent login from a known machine, as discussed in Section 4.2. Only PGRP allows legitimate users to try a relatively large number of wrong passwords ($k_1 = 30$ default) without passing ATTs. In the second usability limitation, only the VS protocol allows an adversary to make enough failed login attempts for the valid usernames so that legitimate users must then pass ATTs first. The third usability limitation is about ATT challenges for a user who successfully logs in from an unknown machine for the first time (as discussed in Section 4.2). Usability drawbacks of cookies are discussed in Section 3.2. By using either IP addresses or both cookies and IP addresses for tracking legitimate users, PGRP is the only protocol that avoids usability drawbacks of using cookies.

As discussed in Section 2, the design of the deterministic function AskATT() in both PS and VS protocols could have security and deployability drawbacks. Given that the design of both PS and VS protocols considers only cookies to identify machines, only PGRP is designed for both login systems that are web-based and those that are not web-based (e.g., SSH and FTP). The last two limitations in the deployability row are as discussed in Section 4.3.

5 Empirical Evaluation

In this section we provide the details of our test setup, empirical results, and analysis of PGRP on two different datasets. PGRP results are also compared to those obtained from testing the PS and VS protocols on the same datasets.

5.1 Datasets

We used two datasets from an operational university network environment. Each dataset logs events of a particular remote login service, over a one-year period each.

SSH Server Log. The first dataset was a log file for an SSH server serving about 44 user accounts. The SSH server recorded details of each authentication event, including: date, time, authentication status (success, failed, or invalid username), username, source IP address, and source port. Log files were for the period of January 4, 2009 to January 22, 2010 (thus, slightly over one year). Table 4 shows that the majority of the login events (95%) are for invalid usernames suggesting that most login attempts are due to SSH guessing attacks. Note that attack login attempts involving valid usernames are not distinguishable from incorrect logins by legitimate users since there is no indication whether the source is malicious or benign. However, there were only few failed login attempts for valid usernames either over short bursts or over the whole log capture period. The number of invalid usernames that appear to be mis-typed valid usernames represents less than 1%.

Email Server Log (Web Interface). The second dataset consisted of log files of a Horde IMP email client[1] for the period of January 15, 2009 to January 25, 2010. The Horde email platform is connected to an IMAP email server in a university environment. For each authentication event, a log entry contained: date, time, authentication status (success, failed, or invalid username), username, and source IP address. Although the number of registered user accounts in this server is 1758, only 147 accounts were accessed. Compared to the SSH log, Table 4 shows that malicious login attempts are far less prevalent, at only about 1%. Login attempts with valid usernames generated by guessing attacks are, as above, not distinguishable. We were unable to determine the percentage of misspelled valid usernames since the log file data including the usernames was anonymized.

Number of:	SSH log	Email log
a) Login events	90,190	48,375
i) with valid usernames	5%	99%
ii) with invalid usernames	95%	1%
b) Valid usernames entered	26	147
c) Invalid usernames entered	13,654	130

Table 4: Login events from SSH (Jan. 4, 2009 to Jan. 22, 2010) and Horde email servers (Jan. 15, 2009 to Jan. 25, 2010)

5.2 Simulation Method and Assumptions

We performed a series of experiments with a Python-based implementation of PGRP with different settings of the configuration variables (k_1, k_2, t_1, t_2, and t_3). The login events in each dataset are or-

[1]Horde IMP is an open source PHP-based Webmail client for IMAP; see `http://www.horde.org/imp/`.

Exp.	Protocol Settings					Number of ATTs		Entries in W		Entries in FT		Entries in FS	
no.	k_1	k_2	t_1	t_2	t_3	SSH	Email	SSH	Email	SSH	Email	SSH	Email
1	30	0	30	1	1	86,118	6,232	70	524	0	0	12	56
2	30	1	30	1	1	**85,669**	**1,002**	70	524	**1**	**1**	12	56
3	30	2	30	1	1	**85,592**	**728**	70	524	**6**	**9**	12	56
4	30	3	30	1	1	**85,552**	**646**	70	524	6	9	12	56
5	30	4	30	1	1	**85,540**	**617**	70	524	6	9	12	56
6	10	3	30	1	1	**85,552**	**668**	70	524	6	9	12	56
7	30	3	30	2	2	**85,554**	**656**	70	524	6	9	**16**	**79**
8	30	3	10	1	1	**85,552**	**678**	**41**	**219**	6	9	**12**	**56**

Table 5: Number of ATTs triggered and number of entries in W, FT, and FS for PGRP (non-default parameters are shaded; for each experiment, changes in results from the previous experiment are in bold face)

dered according to date (older entries first). Each event is processed by PGRP as if it runs in real-time, with protocol tables updated according to the events. Since entries in the tables W, FT, and FS have write-expiry intervals,[1] they get updated at each login event according to the date/time of the current event (i.e., the current time of the protocol is the time of the login event being processed).

We assume that users always answer ATT challenges correctly. While some users will fail in answering some ATTs in practice (see, e.g., [3]), predicting the percentage of failed ATTs depends on the mechanism used to generate the ATTs, the chosen challenge degree of difficulty (if configurable), and the type of the service and its users. The number of generated ATTs by the server can be updated accordingly; for example, if the probability of answering an ATT correctly is p, then the total number of generated ATTs must be multiplied by a factor of $1/p$. Since no browser cookie mechanism is implemented in either services of the datasets, the function $Valid(cookie, un, k_1, status)$ always returns false.

For a comparative analysis, we also implemented the PS and VS protocols under the same assumptions. The cookie mechanism in these protocols is replaced by IP address tracking of user machines since cookies are not used in either datasets. The probability p of the deterministic function (see Appendix A) is set to 0.05 (suggested by Pinkas and Sander [16]), 0.30, and 0.60 in each experiment. For VS, b_1 and b_2 (see Appendix A) are both set to 5 (van Oorschot and Stubblebine [23] suggested 10 as an upper bound for both b_1 and b_2).

[1]For an explanation of the use of expiry intervals, see Section 3.1 under "Data structures".

5.3 Analysis of Results

In Table 5 we list the protocol parameter settings of 8 experiments. For both SSH and email datasets, the total number of ATTs that would be served over the log period, and the maximum number of entries in the W, FT, and FS tables are reported.

In the first five experiments, we change the parameter k_2 from 0 to 4. k_2 bounds the number of failed login attempts after which an ATT challenge will be triggered for the following login attempt. Note that the total number of ATTs served over the log period decreases slightly with a larger k_2 for both datasets. Other parameters have minor effects on the number of ATTs served.

The number of entries in W in the email dataset is larger than the SSH dataset since there are more email users. Note that although the number of failed login attempts is larger in the SSH dataset, the number of entries in FT is smaller than the email dataset because the number of usernames is less in the SSH dataset with very few common usernames (e.g., common first or last names that can be used in brute force attacks). Given that the protocol requires an ATT for each failed login attempt from a source not in W (and with no valid cookie) when k_2 is set to 0, the FT table is empty in the first experiment for both datasets (as the second condition in line 19 in Algorithm 1 is always false). Increasing t_3 increases the number of entries in FS since the table entries last longer as in the seventh experiment.

Tables 6 and 7 show the results of the PS, VS, and PGRP protocols for the SSH and email datasets respectively. Configuration variables not listed in the settings columns for PGRP are the default values (as in Algorithm 1). Test results are analyzed from different perspectives below.

Protocol	Settings	Successful Login number of:				Failed Login number of:					
		a) attempts		b) unique usernames		c) attempts using valid usernames		d) unique valid usernames		e) attempts using invalid usernames	
		w/ ATT	w/o ATT	w/ ATT	w/o ATT	w/ ATT	w/o ATT	w/ ATT	w/o ATT	w/ ATT	w/o ATT
	$p = 0.05$					0	563	0	20	3,930	81,528
PS [16]	$p = 0.30$	346	3,823	24	23	146	417	4	16	19,015	66,443
	$p = 0.60$					557	6	18	2	79,408	6,050
	$p = 0.05$					418	145	12	20	50,806	34,652
VS [23]	$p = 0.30$	346	3,823	24	23	444	119	14	16	59,543	25,915
	$p = 0.60$					557	6	18	2	82,160	3,298
	$k_2 = 0$	412	3,757	24	23	248	315	16	14		
	$k_2 = 1$	50	4,119	13	24	161	402	13	20		
	$k_2 = 2$	20	4,149	7	24	114	449	11	20		
PGRP	$k_2 = 3$	3	4,166	3	24	91	472	5	20	85,458	0
	$k_2 = 4$	1	4,168	1	24	81	482	3	20		
	$k_1 = 10$	3	4,166	3	24	91	472	5	20		
	$t_2 = 2, t_3 = 2$	5	4,164	3	24	91	472	5	20		
	$t_1 = 10$	3	4,166	3	24	91	472	5	20		

Table 6: Experimental results for the SSH dataset (best results are shaded)

a) The number of successful login attempts. The larger the ratio of successful login attempts without answering ATTs to total successful login attempts, the more convenient the login experience for the user. For the default parameters of PGRP (i.e., $k_2 = 3$ in Tables 6 and 7 and other parameters as given in Algorithm 1), the ratio is $4,166/(4,166+3) = 0.999$ for the SSH dataset and $46,201/(46,201+26) = 0.999$ for the email dataset. The ratio decreases slightly as k_2 is decreased in both datasets. No other parameters significantly affect this ratio. All the experiments have a ratio over 99% except when k_2 is 0 for the email dataset (89%). Both PS and VS protocols have a ratio of $3823/(3823 + 346) = 91\%$ for the SSH dataset and 90% for the email dataset.

b) The number of unique usernames in successful logins. For PGRP default parameters, the number of unique usernames in successful logins that involved answering ATTs (in the SSH dataset) is 3. Thus, the majority of valid users were not challenged with any ATT. For the other dataset, 11 valid usernames (out of 147) faced an ATT challenge. Almost all usernames were used in successful logins without answering ATTs in both datasets. k_2 and t_2 are the only parameters that affected the results. For both datasets, most SSH users were asked to answer ATTs in both the PS and VS protocols; therefore, PGRP offers a more convenient login for legitimate users.

c) The number of failed login attempts with valid usernames. Failed login attempts with valid usernames could be from either malicious or benign sources. In the first experiment on PGRP ($k_2 = 0$), there are 315 failed attempts not involving ATTs in the SSH dataset and 1,199 in the email dataset. Given that the source IP addresses of all these attempts are in W, these failed attempts are considered benign. In general, the lower the number of attempts with ATTs the better for user convenience. For PGRP default parameter settings, 16% $(91/(472+91))$ of the failed attempts (with valid usernames) involved ATT challenges in the SSH dataset and 3% $(46/(1,528+46))$ in the email dataset. Even if we assume that all failed attempts (with ATTs) are made by legitimate users, PGRP results are better compared to 74% $(418/(418 + 145))$ for the SSH dataset and 61% for the email dataset in the VS best case (for $p = 0.05$). PS offers slightly better results, however, this is only when $p = 0.05$ which also reduces the number of required ATTs for password guessing attempts (i.e., with invalid usernames as in the last column in Tables 6 and 7).

d) The number of unique valid usernames in failed login attempts. In both datasets, setting $k_2 \geq 1$ in PGRP causes a significant decrease in the number of unique valid usernames that face ATT challenges in failed login attempts. Other parameters have no significant effect in this manner. For $k_2 = 3$ (default value), in both datasets the number of affected usernames (i.e., the number of legitimate users that are asked to answer ATTs for failed login attempts) is comparable to PS results but less than VS; therefore, PGRP offers a more convenient login for legitimate users.

		Successful Login number of:				**Failed Login** number of:					
		a) attempts		b) unique usernames		c) attempts using valid usernames		d) unique valid usernames		e) attempts using invalid usernames	
Protocol	**Settings**	w/ ATT	w/o ATT	w/ ATT	w/o ATT	w/ ATT	w/o ATT	w/ ATT	w/o ATT	w/ ATT	w/o ATT
	$p = 0.05$					166	1,408	7	101	24	550
PS [16]	$p = 0.30$	4,609	4,1618	134	102	442	1,132	29	79	85	489
	$p = 0.60$					1,524	50	98	10	543	31
	$p = 0.05$					961	613	41	101	291	283
VS [23]	$p = 0.30$	4,609	41,618	134	102	1,103	471	55	79	350	224
	$p = 0.60$					1,528	46	100	10	545	29
	$k_2 = 0$	5,283	40,944	134	100	375	1,199	88	71		
	$k_2 = 1$	279	45,948	69	127	149	1,425	47	108		
	$k_2 = 2$	81	46,146	32	132	73	1,501	18	108		
PGRP	$k_2 = 3$	26	46,201	11	134	46	1,528	11	108	574	0
	$k_2 = 4$	9	46,218	5	134	34	1,540	6	108		
	$k_1 = 10$	29	46,198	13	134	65	1,509	13	108		
	$t_2 = 2, t_3 = 2$	36	46,191	11	134	46	1,528	11	108		
	$t_1 = 10$	34	46,193	13	134	70	1,504	13	108		

Table 7: Experimental results for the email dataset (best results are shaded)

e) The number of failed login attempts with invalid usernames. Any login attempt with invalid username triggers an ATT in PGRP (i.e., no failed login attempt with invalid usernames avoids an ATT). Indeed, all attempts with invalid usernames trigger ATTs in both datasets. In contrast, for the SSH dataset, only 0.046% in PS and 0.59% in VS trigger ATTs for $p = 0.05$ (0.04% and 0.51% in the email dataset).

Summary of comparison. The trade-off between user convenience (item (c) above) and login security with respect to password guessing (item (e)) in both PS and VS protocols is evident from the above discussion; i.e., increasing the number of ATTs to limit password guessing attempts also increases the number of ATTs legitimate users must answer. Such a trade-off is significantly limited with PGRP. Moreover, the number of legitimate login attempts that trigger ATTs (and the number of affected users) is significantly lower in PGRP than both PS and VS. On the other hand, in PGRP, more ATTs must be answered in password guessing attacks; if g is the number of password guessing attempts for m usernames, PGRP requires answering ATT challenges for at least $g - k_2 m$ password guessing attempts. Our datasets represent two very different scenarios: the SSH server received almost 95% invalid login attempts, and the email server received only 1% of such attempts (see Table 4). Yet, as the above analysis indicates, PGRP is significantly better (for both security and usability) than previous ATT-based protocols in both cases, and it can be deployed without affecting the login experience of legitimate users.

6 Concluding Remarks

Online password guessing attacks on password-only systems have been observed for decades (see e.g., [20]). 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 datasets (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).

Acknowledgments

The second author is supported by an NSERC post-doctoral fellowship and by NSERC ISSNet (`http://issnet.ca`). 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 ISS-Net is also acknowledged.

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 measurment*, pages 267–272, New York, NY, 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] 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.

[5] D. Florêncio, C. Herley, and B. Coskun. Do strong web passwords accomplish anything? In *USENIX workshop on Hot topics in security (HotSec'07)*, pages 1–6, Berkeley, CA, USA, 2007.

[6] K. Fu, E. Sit, K. Smith, and N. Feamster. Dos and don'ts of client authentication on the web. In *USENIX Security Symposium*, pages 251–268, Washington, DC, USA, 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`. Accessed: Feb. 2010.

[8] Y. He and Z. Han. User authentication with provable security against online dictionary attacks. *Journal of Networks (JNW)*, 4(3):200–207, May 2009.

[9] T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. In *IEEE Symposium on Security and Privacy*, pages 211–225, Washington, DC, USA, 2005. IEEE Computer Society.

[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. In *USENIX Security Symposium*, Washingtion, DC, USA, Aug. 2010.

[11] C. Namprempre and M. N. Dailey. Mitigating dictionary attacks with text-graphics character CAPTCHAs. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E90-A(1):179–186, 2007.

[12] A. Narayanan and V. Shmatikov. Fast dictionary attacks on human-memorable passwords using time-space tradeoff. In *ACM Computer and Communications Security (CCS'05)*, pages 364–372, Alexandria, VA, USA, Nov. 2005.

[13] National Institute of Standards and Technology (NIST). hashbelt. Accessed: Sept. 2010. `http://www.itl.nist.gov/div897/sqg/dads/HTML/hashbelt.html`.

[14] NetworkWorld.com. The biggest cloud on the planet is owned by ... the crooks. News article (Mar. 22, 2010). `http://www.networkworld.com/community/node/58829`.

[15] J. Nielsen. Stop password masking. Online article (June 23, 2009). `http://www.useit.com/alertbox/passwords.html`.

[16] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *ACM conference on Computer and communications security (CCS'02)*, pages 161–170, Washington, DC, USA, Nov. 2002.

[17] D. Ramsbrock, R. Berthier, and M. Cukier. Profiling attacker behavior following SSH compromises. In *IEEE/IFIP Dependable Systems and Networks (DSN'07)*, pages 119–124, Edinburgh, UK, June 2007.

[18] SANS.org. Important information: Distributed SSH brute force attacks. SANS Internet Storm Center handler's diary (June 18, 2010). `http://isc.sans.edu/diary.html?storyid=9034`.

[19] SANS.org. The top cyber security risks. Online article (Sept. 2009). `http://www.sans.org/top-cyber-security-risks/`.

[20] C. Stoll. *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage.* Doubleday, 1989.

[21] textCAPTCHA. Accessible Text CAPTCHA Logic Questions. Accessed: Mar. 2010. `http://textcaptcha.com`.

[22] TheRegister.co.uk. Botnet pierces Microsoft Live through audio captchas. News article (Mar. 22, 2010). `http://www.theregister.co.uk/2010/03/22/microsoft_live_captcha_bypass/`.

[23] P. C. van Oorschot and S. Stubblebine. On countering online dictionary attacks with login histories and humans-in-the-loop. *ACM Transactions on Information and System Security (TISSEC)*, 9(3):235–258, 2006.

[24] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Eurocrypt*, pages 294–311, Warsaw, Poland, May 2003.

[25] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How dynamic are IP addresses? *SIGCOMM Comput. Commun. Rev.*, 37(4):301–312, 2007.

[26] J. Yan and A. S. E. Ahmad. A low-cost attack on a Microsoft CAPTCHA. In *ACM Computer and Communications Security (CCS'08)*, pages 543–554, Alexandria, VA, USA, Oct. 2008.

[27] J. Yan and A. S. E. Ahmad. Usability of CAPTCHAs or usability issues in CAPTCHA design. In *Symposium on Usable Privacy and Security (SOUPS'08)*, pages 44–52, Pittsburgh, PA, USA, July 2008.

A Background on Previous ATT-based Protocols

Pinkas and Sander [16] introduced the topic with a basic login protocol (see pseudo-code in Algorithm 2), that requires answering an ATT challenge first before entering the {username, password} pair. Failing to answer the ATT correctly prevents the user from starting the authentication process. This protocol requires the adversary to pass an ATT challenge for each password guessing attempt.

While this simple protocol is effective against online dictionary attacks assuming that the used ATTs are secure, legitimate users must also pass an ATT challenge for every login attempt. Therefore, this protocol affects user convenience substantially, and requires the login server to generate an ATT challenge for every login attempt.

Pinkas and Sander [16] then made their actual proposal, a login protocol that reduces the number of ATTs legitimate users are required to pass; see pseudo-code in Algorithm 3 (PS protocol). The protocol stores a browser cookie on the machine of users who had previously logged in successfully. The cookie is tied to the username of the last successful login attempt. Once the user requests the login server URL, the user's browser sends the cookie (if any) back to the server. The protocol then requests the user to enter a {username, password} pair. If the pair is correct and a valid cookie (i.e., an unexpired cookie indicating that a successful login for the username was made from the same browser) is received from the browser then the user is granted access. If the pair is correct but no valid cookie is received, then an ATT challenge must be answered before account access is granted. Otherwise, if the {username, password} pair is incorrect then according to a function $AskATT(username, password)$, an ATT challenge might be required before informing the user that the {username, password} pair is incorrect.

$AskATT(username, password)$ must be a deterministic function of the entered {username, password} pair such that for a specific pair, an ATT challenge is either always requested, or never (this function is denoted $AskATT(un, pw)$ in Algorithm 3). That is, for a password space of size N, pN of the possible passwords require ATTs (e.g., if $p = 0.05$, $0.05 \times N$ of the password space for a given username require ATTs).

With this protocol, legitimate users must pass ATT

```
begin
    if ATTChallenge() = Pass then
        ReadCredential(un, pw)                 // login prompt to enter username/password pair
        if LoginCorrect(un, pw) then           // username/password pair is correct
            Access is granted to the server
        else
            Message('The username or password is incorrect')
    else
        Message('ATT answer is incorrect')
end
```

Algorithm 2: Secure but inconvenient login protocol [16]

```
begin
    ReadCredential(un, pw)                     // login prompt to enter username/password pair
    if LoginCorrect(un, pw) then               // username/password pair is correct
        if Valid(un, cookie) then              // cookie unexpired and matches username
            GrantAccess()                      // server access is granted
        else                                   // no cookie or the cookie is invalid
            if ATTChallenge() = Pass then GrantAccess()   // server access is granted
            else Message('login fails')
    else                                       // username/password pair is incorrect
        if AskATT(un, pw) = True then
            if ATTChallenge() = Pass then Message('login fails')
            else Message('login fails')
        else
            Message('login fails')
end
```

Algorithm 3: PS protocol, adapted from Pinkas and Sander [16]

challenges in the following cases: (i) when the user logs in from a machine for the first time; and (ii) when the user's {username, password} pair is incorrect and $AskATT()$ triggers an ATT. On the other hand, an automated program needs to correctly answer an ATT for each password guessing attempt except one case: when the {username, password} pair is incorrect and a deterministic function $AskATT()$ did not request an ATT.

In addition to the correct password, this protocol requires ATTs for a fraction p of the incorrect passwords. Therefore, an adversary can find out that $(1-p)(N-1) \approx N-pN$ of the passwords in the password space N are incorrect without answering any ATT challenge. The expected number of ATTs an adversary must correctly answer to guess a password correctly is $\frac{1}{2}pN$. Thus, if the adversary is willing to answer c ATTs, the probability of finding a correct password is c/pN. For better defence against online dictionary attacks, the function $AskATT()$ should request ATTs for the majority of the possible passwords in the overall password space (i.e., $p > 0.75$). However, the probability that a legitimate user is given an ATT challenge upon entering an incorrect password will also increase, creating a trade-off between password security and user convenience. In fact, setting $p = 1$ makes this protocol similar to the basic protocol above, except for successful logins with valid cookies where no ATT is required.

Van Oorschot and Stubblebine [23] proposed modifications to the previous protocol (see Algorithm 4; VS protocol) which track failed logins per username to impose ATT challenges after exceeding a configurable threshold of failures (b_1 threshold for correct {username, password} pair and b_2 threshold for incorrect pair; see Algorithm 4). Hence, for an incorrect {username, password} pair, the decision to request an ATT not only depends on the function $AskATT()$

```
Input:
    $FT$ (global variable, def=0, expires after $t_2$) //table of number of failed logins per username

1  begin
2      ReadCredential(un, pw)                                   // login prompt to enter username/password pair
3      if LoginCorrect(un, pw) then                             // username/password pair is correct
4          if Valid(un, cookie) then                            // cookie unexpired and matches username
5              Access is granted to the server
6          else                                                 // no cookie or the cookie is invalid
7              if (OwnerMode(un)) OR ($FT[un] \geq b_1$) then
8                  if ATTChallenge() = Pass then GrantAccess()  // server access is granted
9                  else Message('login fails')
10             else
11                 GrantAccess()                                // server access is granted
12     else                                                     // username/password pair is incorrect
13         if (AskATT(un, pw) = True) OR ($FT[un] \geq b_2$) then
14             if ATTChallenge() = Pass then Message('login fails')
15             else Message('login fails')
16         else
17             Message('login fails')
18 end
```

Algorithm 4: VS protocol, adapted from van Oorschot and Stubblebine [23]

but also on the number of failed login attempts for the username (line 13 in Algorithm 4). In addition, upon entering correct credentials in the absence of a valid cookie, the user is asked whether the machine in use is trustworthy and if the user uses it regularly. The cookie is stored in the user's machine only if the user responds yes to the question. This approach aims to reduce the possibility of cookie theft since a negative answer is expected if the user logs in from a public machine. The user account is set to be in *non-owner mode* for a specified time window when a login is successful without receiving a valid cookie from the user machine; otherwise the account is set to *owner mode.*

The number of incorrect passwords that an adversary can eliminate without passing any ATT challenge is decreased to about $(1 - p)b_2$. Moreover, the adversary is expected to need to correctly answer about $N/2$ ATTs in order to guess a password correctly as opposed to $\frac{1}{2}pN$ in the PS protocol. While this VS protocol addresses the security drawback of the PS [16] algorithm, the legitimate user always faces an ATT challenge once the threshold b_2 is exceeded. This feature enables adversaries to affect user login convenience, by initiating $\geq b_2$ failed login attempts for each targeted username, forcing ATT challenges for the following login attempts.