IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013
ISSN: 2320 - 8791
www.ijreat.org

Revisiting Defenses Against Large-Scale Online Password Guessing Attacks

Soniya.J
Department of Computer Science & Engineering,
St. Peter University, Chennai.

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 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 datasets and find it more promising than existing proposals.

Keywords— *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 honey pot setup has been reported to suffer on average 2,805 SSH malicious login attempts per computer per. 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 (ATTs, e.g., CAPTCHAs). 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 (e.g., Conficker), 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 time-out period; and time-limited account locking. Many existing 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 for usability issues related to commonly used CAPTCHAs. Due to successful attacks which break ATTs without human solvers, 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, fueling a growing tension between security and usability of ATTs. Therefore, we focus on reducing user annoyance by challenging users with fewer ATTs, while at the same time subjecting bot logins to more ATTs, to drive up the economic cost to attackers.

Two well-known proposals for limiting online guessing attacks using ATTs are Pinkas and Sander (herein denoted PS), and van Oorschot and Stubblebine (herein denoted VS). 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 (the fraction of incorrect login attempts requiring an ATT) PS allows attackers to eliminate 95 percent 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 certain circumstances.

2.PASSWORD GUESSING RESISTANT PROTOCOL

Protocol goals**:** 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, 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 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 [17]).

3. Data Structure

Data structures: PGRP maintains three data structures:

1) W: A list of fsource IP address, usernameg 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 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 a valid username attempted from srcIP. A maximum of k1 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

Functions:

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

a) 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).

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

c) GrantAccess(IN: un,cookie): The function sends the cookie to the user's browser and then enables access to the specified user account.

d) Message(IN: text): Shows a text message.

e) ATTChallenge(OUT: Pass/Fail): Challenges the user with an ATT and returns "Pass" if the answer is correct; otherwise, it returns "Fail".

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

g) Valid(IN: cookie,un,k1,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 k1. The function returns true only when a valid cookie is received. If state = true (i.e., the entered user credentials are correct, as in line 4 of Algorithm 1), a new cookie is created (if cookies are supported

in the login system) including the following information: username, expiry date, and a counter of the number of failed login attempts (since the last successful login; initialized to 0).

Decision Function for Requesting ATTs

Below we discuss issues related to ATT challenges as provided by the login server in Algorithm 1. 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. For definitions of W, FT, and FS

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 V alid returns true) and the number of failed login attempts from the user machine's IP address for that username, FS[srcIP; un], is lessthan k1 over a time period determined by t3; 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 lessthan k1 over a time period determined by t3; 3) the number of failed login attempts from any machine for that username, FT[un], is below a threshold k2 over a time period determined by t2. The last case enables a user who tries to log in 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 k2

(default 3), this might indicate a guessing attack and hence the user must pass an ATT challenge.

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 V alid 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 k1 (line 16) over a time period determined by t3;

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 k1 (line 16) over a time period determined by t3;

3) the username is valid and the number of failed login attempts (from any machine) for that username, FT[un], is below a threshold k2 (line 19) over a time period determined by t2. A failed login attempt from a user with a valid cookie or in the whitelistW 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 fsource IP address, username pair will also be incremented (line 17). Once the cookie counter or the corresponding FS entry hits or exceeds the threshold k1 (default value 30), the user must correctly answer an ATT challenge.

Output messages: PGRP shows different messages in case of incorrect fusername, passwordg pair (lines 21 and 24) and incorrect answer to the given ATT challenge (lines 14 and 26). While showing a human that the entered fusername, passwordg 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.

CONCLUSION

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).

ACKNOWLEDGMENTS

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.

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