The International Journal Of Engineering And Science (IJES)
||Volume||2 ||Issue|| 6 ||Pages|| 35-38||2013||
ISSN(e): 2319 – 1813 ISSN(p): 2319 – 1805

Improving Efficiency of Password Security Against Large Scale Online Attacks

Harshitha.H.K[1] And Sreedevi.N[2]

[1] *4th sem M.Tech, Department of CSE, MVJ College of Engineering, Bangalore-560067, Visveswariah Technological university, Belgaum, Karnataka, India*

[2] *Hod,Assistant Professor, Department of Computer Science and Engineering, MVJ College of Engineering, Bangalore-560067, Karnataka, India*

----------ABSTRACT----------

The use of passwords is a major point of vulnerability in computer security, as passwords are often easy to guess by automated programs running dictionary attacks. From a user's perspective user friendliness is a key requirement. 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 online attacks is a major concern in security systems. 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 a novel authentication scheme that preserves the advantages of conventional password authentication is proposed, while simultaneously raising the costs of online dictionary attacks by orders of magnitude. The proposed scheme is easy to implement and overcomes some of the difficulties of previously suggested methods for improving the security of user authentication schemes. The key idea is to efficiently combine traditional password authentication with a challenge that is very easy to answer by human users, but is infeasible for automated programs attempting to run dictionary attacks. This is done without affecting the usability of the system. A new Password Guessing Resistant Protocol (PGRP) is proposed, 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.

Date of Submission: 20 June 2013, Date of Publication: 5.july 2013

I. INTRODUCTION

Online attacks on password based systems are inevitable and commonly observed against web applications. Online guessing attacks on password-based 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 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 and attackers currently control large botnets, 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 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. 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, the focus is 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.

II. METHODOLOGY

The protocol proposed in this work here uses the ATTs-Automated turing tests (also commonly known as CAPTCHAS) to operate against online attackers. The Automated Turing test (ATT) is a standard security technique for addressing the threat of undesirable or malicious bot programs. The basic idea of CAPTCHA is to force there to be a human in the loop –it works as a simple two-round authentication protocol as follows.

S(ervice) → C(lient): a CAPTCHA challenge
C → S: response

A CAPTCHA challenge is a test that most humans can pass but current computer programs cannot pass. Such a test is often based on a hard, open problem in artificial intelligence, e.g. automatic recognition of distorted text, or of human speech against a noisy background. Usually, CAPTCHA challenges are automatically generated and graded by a computer. Since only humans are able to return a sensible response, an automated Turing test embedded in the above protocol can verify whether an attack is automated or there is a human behind the challenged computer.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 without answering any ATTs. 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 login server as a white-list, or cookies stored on client machines. A white-listed IP address and/or client cookie expires after a certain time. PGRP accommodates both graphical user interfaces (e.g., browser-based logins) and character-based interfaces (e.g., SSH logins

). PGRP uses either cookies or IP addresses, or both for tracking legitimate users. Tracking users through their IP addresses also allows PGRP to increase the number of ATTs for password guessing attacks and meanwhile to decrease the number of ATTs for legitimate login attempts.

III. SYSTEM DESIGN

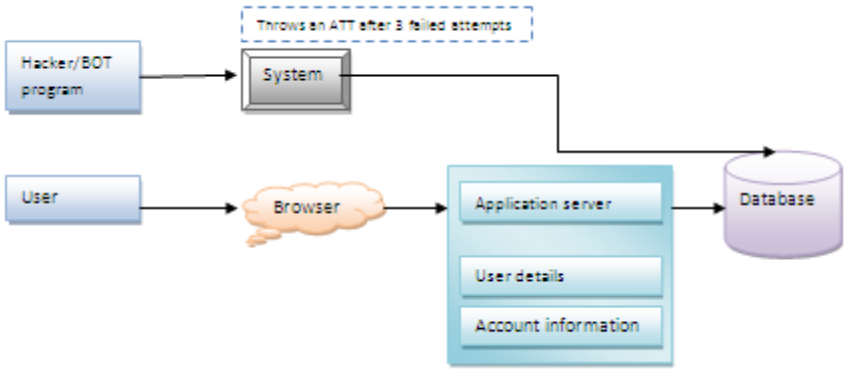


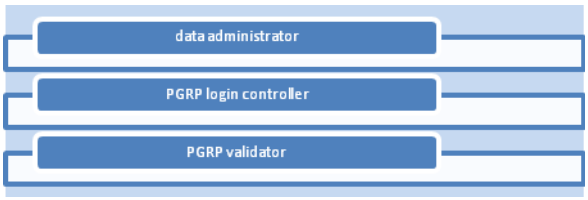
Fig 1:Architectural diagram

3.1.Function modules:

Data administrator

The administrator's major objective is to control the data and usage of the legitimate user. The details of the user are got in a form for the creation of account, and the user's credentials are checked manually then using those details the account is created if the credentials are correct. After the successful login the legitimate user can do all the required tasks. The login program invokes the user shell and enables command execution. Login is the very first step for the user to use their account. The usage of login form is to give security to the user. If the user doesn't have the account then the user should create the account. A new account is created by the administrator for the legitimate user. Data structures/tables created: 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 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.



3.2. PGRP Login Controller

The main objective of this module is to control the access to the user accounts. The controller displays the login form and upon receiving the login credentials, sends these data to login validator. The login controller then, based upon the response from login validator decides whether to grant access or deny access to the user. It also keeps track of user status information with the help of user cookies.

3.3.PGRP Validator

The main function of this module is to validate the data entered by the user in the login form by comparing it against the data stored by the administrator corresponding to that user .It checks for valid email address, passwords, no. of failed login attempts etc: . Update and deletion functions are also carried out by this module. It decides whether a user is valid or not by following his/her login behavior and checking his/her failed login attempts against threshold values assigned to them. If a user exceeds the failed threshold value and fails to answer an ATT function, user is asserted as invalid and denied access. PGRP uses the following functions:

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. *V alidUsername(IN: un; OUT: true/false):* If the provided username exists in the login system, the function returns true; otherwise, it returns false.
g. *V alid(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.

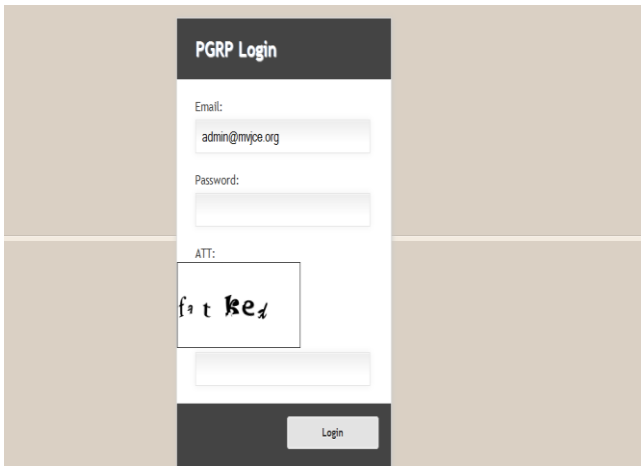
IV. ANALYSIS OF TEST CASES

Table 6.4 Summary of test cases

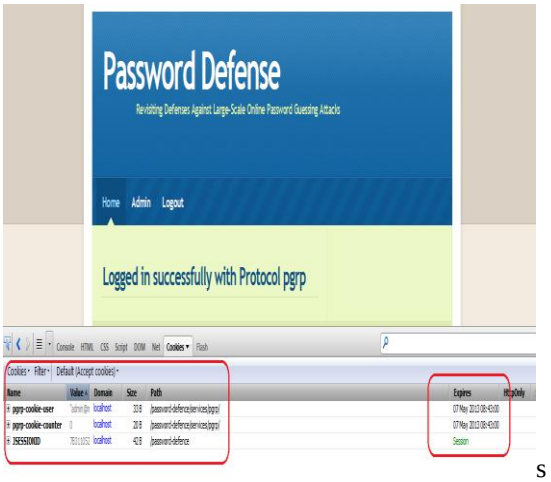
Test ID	Test Name	Input	Actual result	Expected result
1	Known User login (valid cookie)	Valid username and password	Successful login message displayed	Successful login Message displayed
2	Known user login (expired cookie)	Valid username and password	Requested to answer an ATT challenge	Requested to answer an ATT challenge
3	User login from unknown machine	Valid username and password	After exceeding known user failure attempt threshold(10)	After exceeding known user failure attempt threshold ,challenged

			,challenged with an ATT	with an ATT
4	User login from known machine after few failed attempts	Valid username and password ,correct reply to ATT challenge	ATT answered successfully and access granted	ATT answered successfully and access granted
5	Hacker/bot login	Valid Username and incorrect password	3 login attempts allowed ,later challenged with ATT	3 login attempts allowed ,later challenged with ATT

All the above test cases have been tested and expected results have been observed.



napshot: login with wrong password; limit exceeded



s

snapshot: successful login showing creation of user cookie

V. CONCLUSION

Online password guessing attacks on password-only systems have been observed for decades. Present-day attackers targeting such systems are empowered by having control of thousand to million node botnets. In 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. The work shows 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, 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.

REFERENCES

[1] B. Pinkas and T. Sander. -Securing passwords against dictionary attacks in ACM conference on Computer and communications security

[2] E. Bursztein, S. Bethard, J. C. Mitchell, D. Jurafsky, and C. Fabry "How good a humans at solving CAPTCHAs? "

[3] P. C. van Oorschot and S. Stubblebine- On countering online dictionary attacks with login histories and humans-in-the-loop.