Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 2, February 2014, pg.815 – 826

RESEARCH ARTICLE

Implementation of Password Guessing Resistant Protocol (PGRP) to Prevent Online Attacks

[1]M.YUVARAJ, [2]A.R.BHARATHIDASAN, [3]N.KUMAR

[1, 2] M.E Student, [3]Assistant Professor

[1, 2, 3] Vel Tech Multi Tech Dr.Rangarajan Dr.Sakunthala Engineering College, Avadi, Tamilnadu, India

[1] yuvaraj426@gmail.com, [2] bharathi.cse09@gmail.com, [3] nkvsc.org@gmail.com

***Abstract*-The inadequacy of login protocols designed to address large scale online dictionary attacks (e.g., from a botnet of hundreds of thousands of nodes). Brute force and dictionary attacks on password-only remote login services are now widespread and emerging technique. 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 propose a protocol called Password Guessing Resistant Protocol (PGRP), derived upon revisiting recent proposals designed to avoid such attacks. In PGRP limits the total number of login attempts from unknown remote users to as low as a single attempt per username, the users in most cases (e.g., when attempts are made from known, frequently-used machines) can make multiple failed login attempts before being challenged with an ATT. We evaluate the performance of PGRP with two real-world data sets and find out more than the existing proposals.**

***Key Terms*- Online Attacks, Brute force, ATT, PGRP**

1. INTRODUCTION

1.1 Introduction

Online guessing attacks on password-based systems are inevitable and commonly observed against web applications and SSH logins. In a recent report, SANS identified password guessing attacks on websites as a top cyber security risk. As an example of SSH password guessing attacks, one experimental Linux honeypot setup has been reported [18] to suffer on average 2,805 SSH malicious login attempts per computer per day.

© 2014, IJCSMC All Rights Reserved

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 [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 (thus allowing effective password dictionaries), and attackers currently control large botnets (e.g., Conficker [15]), 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 [28] for usability issues related to commonly used CAPTCHAs. Due to successful attacks which break ATTs without human solvers (e.g., [27], [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, 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 [17] (herein denoted *PS*), and van Oorschot and Stubblebine [23] (herein denoted VS). For convenience, a review of these protocols is given in Section 6. 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 incorrect login attempts requiring an ATT) *PS* allows attackers to eliminate 95 percent of the password space without answering any ATTs.

1.2 Problem Definition

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.

1.3 Scope Of The Paper

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 user.

© 2014, IJCSMC All Rights Reserved

2. SYSTEM DESIGN

2.1 Existing System

- 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.

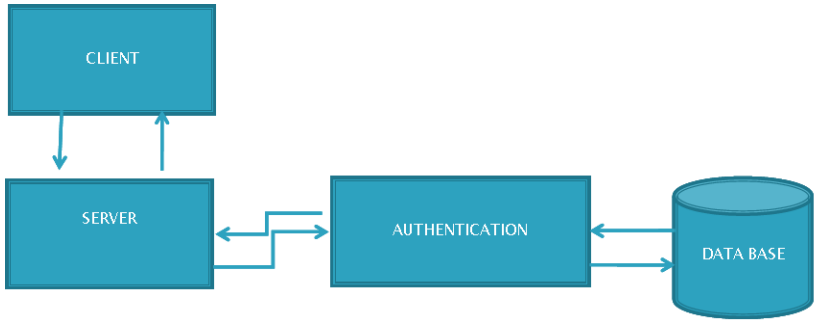
2.1.1 Limitations

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

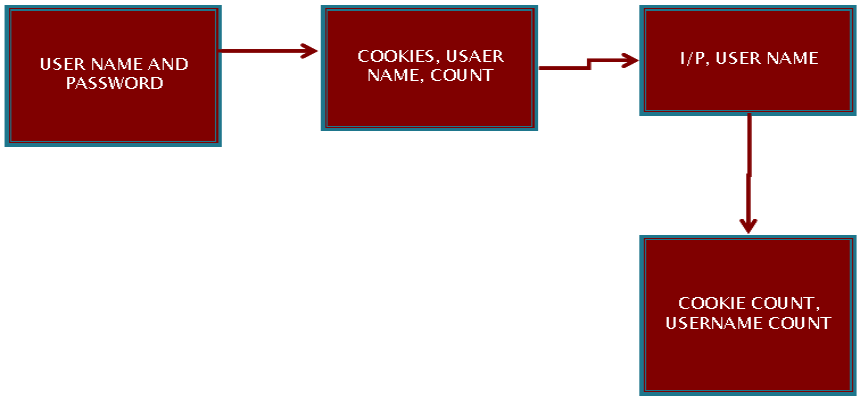
2. 2 Proposed System

- 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.

2.3 Architecture Diagram

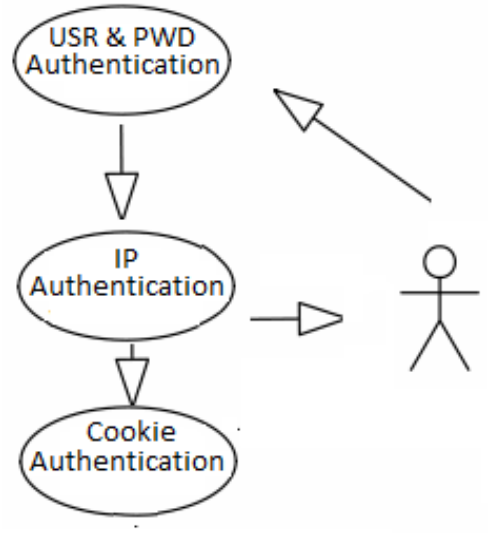


2.4 Block Diagram

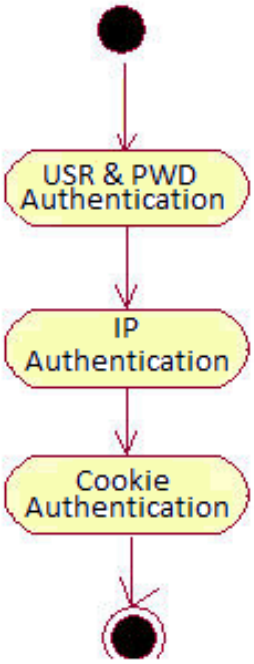


© 2014, IJCSMC All Rights Reserved

2.5 Use Case Diagram



2.6 Activity Diagram



© 2014, IJCSMC All Rights Reserved

3. SYSTEM ANALYSIS

3.1 Modular Description

The proposed system can be broadly divided into three modules, namely:

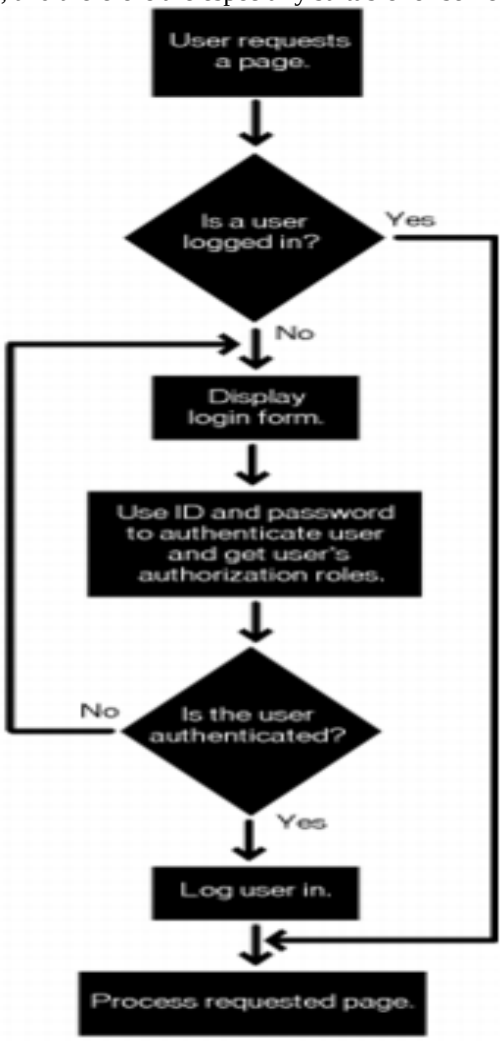
- USR & PWD Authentication
- IP Authentication
- Cookie Authentication

3.1.1 USR & PWD Authentication

- USR, PWD are two entities share a password in advance and use the password as the basis of authentication.

- Existing password authentication schemes can be categorized into two types: weak-password authentication schemes and strong-password authentication schemes.

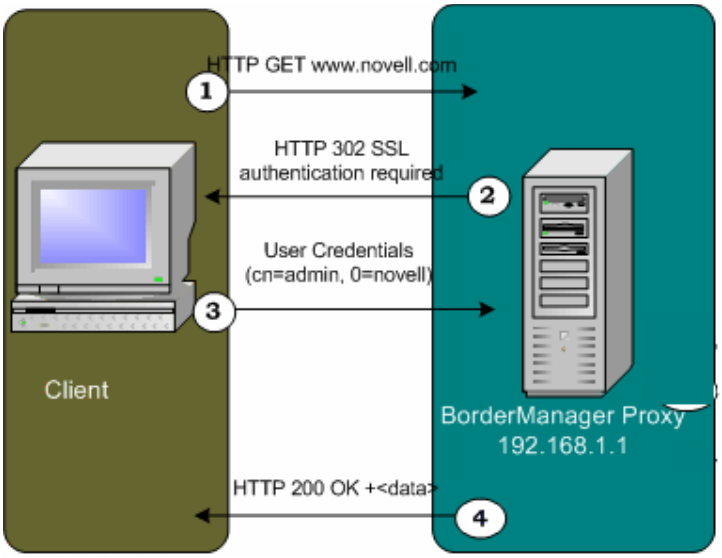
- In general, strong-password authentication protocols have the advantages over the weak-password authentication schemes in that their computational overhead are lighter, designs are simpler, and implementation are easier, and therefore are especially suitable for some constrained environments.



© 2014, IJCSMC All Rights Reserved

3.1.2 IP Authentication

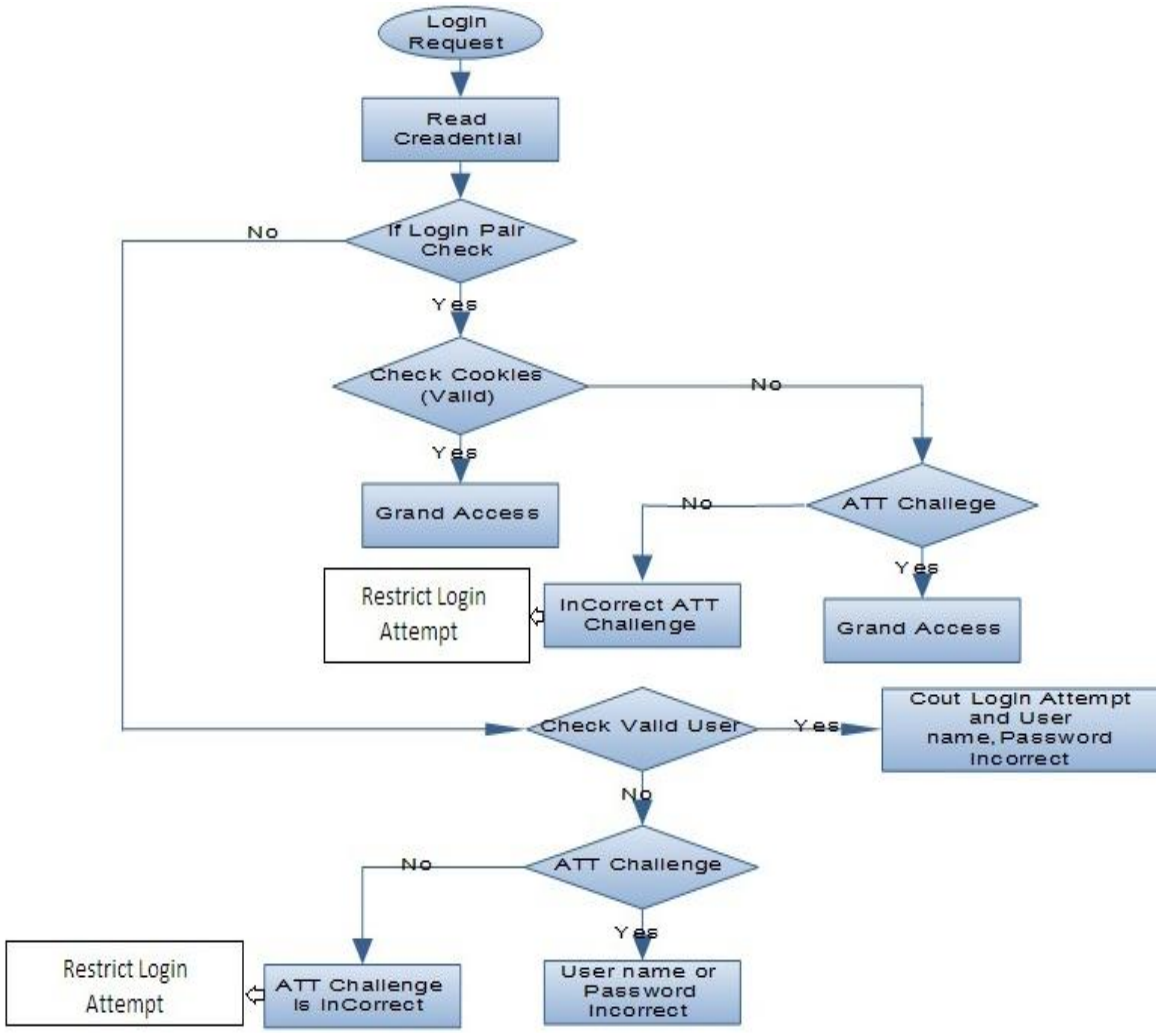
- Internet Protocol Authentication (IPAuth) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session.
- IPAuth also includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session.
- IPAuth is an end-to-end security scheme operating in the Internet Layer of the Internet Protocol Suite.
- It can be used in protecting data flows between a pair of hosts (host-to-host), between a pair of security gateways (network-to-network), or between a security gateway and a host (network-to-host).



3.1.3 Cookie Authentication

- Cookie based authentication scheme by delivering a session cookie with the username of the person.
- All the scripts there after check for this cookie, and they also pull its value (in this case, username) to run the required queries to the database.
- The only thing successfully retrieving a cookie should imply, from a security standpoint, is that, at one time, someone using that particular browser (session, if you're using session cookies) was successfully authenticated period.

© 2014, IJCSMC All Rights Reserved



3.2 Feasibility Analysis

A system is a feasible system only if it is feasible within limited resource and time. The different feasibilities that are to be analyzed are:

3.2.1 Technical Feasibility

According to Roger S Pressman, "Technical Feasibility is the assessment of the technicalities of the system". The system needs IBM compatible machines with VGA monitors. The system is developed for Windows 2000 environment. J2SDK1.4 is used to develop the system. The technical feasibility has been carried out and the system proves to be technically feasible for development.

3.2.2 Operational Feasibility

Operational Feasibility deals with the study of prospects of system. This system is operationally feasible as most of the entities are available as simulated model. The security algorithms can be integrated with these simulated models. The system ensures safe transfer and storage of data.

3.2.3 Economic Feasibility

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based system paper. As this system proposes the general PC system as Data Unit, Switch and Servers, it reduces the extra cost incurred on procuring these devices. Thus this system proves to be economically feasible.

© 2014, IJCSMC All Rights Reserved

4. BACKGROUND & RELATED WORKS

4.1 Usability Comments on 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 1) an unknown machine (i.e., no valid cookies or white-listed IP addresses), or 2) 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. However, we have not yet carried out any formal user testing.

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 k1, 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 an uncommon occurrence, as was apparent from our collected data.
This may be a common occurrence for several reasons: 1) if users need multiple attempts to recall the correct password; 2) if users cycle-through multiple passwords due to multi password interference [5]; and 3) typing errors including activating the caps lock key, sometimes aggravated by onscreen masking of password characters (see, e.g., Nielsen's blog [16]). 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=10,000) as that may aid guessing attacks when a cookie is stolen or a dynamic white-listed IP address is assigned to an attacker's machine (i.e., a bot). Note that in *VS* [23], an adversary can make a certain number of failed connection attempts (the threshold b_2 in Fig. 4) 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.

4.2 System Resources

No lists are maintained in the *PS* protocol thus no extra memory overhead is imposed on the login server. In the VS protocol, 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. 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 white list, 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 t1 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 nonexisting 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 t3 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.

© 2014, IJCSMC All Rights Reserved

4.3 Background On Previous ATT-Based protocols

Pinkas and Sander [17] introduced the topic with a straw man origin protocol that requires answering an ATT challenge first before entering the {*username, password*} pair. Failing to answer the ATT correctly prevents the user from proceeding further. This protocol requires the adversary to pass an ATT challenge for each password guessing attempt, in order to gain information about correctness of the guess.

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 then made their actual proposal, a login protocol that reduces the number of ATTs legitimate users are required to pass; see pseudocode *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 Ask ATT username; password, an ATT challenge might be required before informing the user that the {*username, password*} pair is incorrect. Ask ATT 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 Ask ATT (*un,pw*)in Fig. 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 × *N* of the password space for a given username require ATTs).

With this protocol, legitimate users must pass ATTs in the following cases: 1) when the user logs in from a machine for the first time; and 2) when the user's {username, password} pair is incorrect and Ask ATT() 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 Ask ATT() 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 confirm that (1-*p*) (*N*-1) ≈*N- pN* of the passwords in the password space are incorrect without answering any ATT challenge. The expected number of ATTs an adversary must correctly answer to guess a password correctly is 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 Ask ATT() should request ATTs for the majority of the possible passwords in the overall password space (e.g., 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 strawman protocol, except for successful logins with valid cookies where no ATT is required. Van Oorschot and Stubblebine proposed modifications to the previous protocol which track failed logins per username to impose ATT challenges after exceeding a configurable threshold of failures (threshold b_1 for correct {*username, password*} pair and threshold b_2 for incorrect pair; see Fig. 4). Hence, for an incorrect {*username, password*} pair, the decision to request an ATT not only depends on the function Ask ATT() but also on the number of failed login attempts for the username. 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 no 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

© 2014, IJCSMC All Rights Reserved

a password correctly as opposed to 1/2 *pN* in the *PS* protocol. While this VS protocol addresses the security drawback of the *PS* [17] 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 subsequent login attempts.

5. IMPLEMENTATION

5.1 Data Sets

We used two data sets from an operational university network environment. Each data set logs events of a particular remote login service, over a one-year period each. SSH Server Log. The first data set 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 percent) 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 mistyped valid usernames represents less than one percent. Email server log (web interface). The second data set consisted of log files of a Horde IMP email client2 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 1,758, only 147 accounts were accessed. Compared to the SSH log, Table 4 shows that malicious login attempts are far less prevalent, at only about one percent. 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.

5.1.1 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 data set are ordered 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,[3] 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, 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 actor of 1=p. Since no browser cookie mechanism was implemented in our tests, in either services of the data sets, he function valid cookie; un; k1; status always returns else. In the absence of a browser cookie mechanism, a machine from which a user has previously logged in successfully would not be identified by the login system if he machine uses a different IP address that is not in W. Such legitimate users will e challenged with ATTs in this case. or 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 racking of user machines since cookies are not used in either data sets. The probability p of the deterministic function.

6. CONCLUSIONS

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. Our empirical experiments on two data sets (of one-year duration) gathered from

© 2014, IJCSMC All Rights Reserved

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). the number of users in a system. PGRP can also be used with remote login services where cookies are not applicable.

REFERENCES

[1] Amazon Mechanical Turk. https://www.mturk.com/mturk/,June 2010.
[2] S.M. Bellovin, "A Technique for Counting Natted Hosts," Proc.ACM SIGCOMM Workshop Internet Measurement, pp. 267-272,2002.
[3] E. Bursztein, S. Bethard, J.C. Mitchell, D. Jurafsky, and C.Fabry, "How Good Are Humans at Solving CAPTCHAs? A Large Scale Evaluation," Proc. IEEE Symp. Security and Privacy,May 2010.
[4] M. Casado and M.J. Freedman, "Peering through the Shroud: The Effect of Edge Opacity on Ip-Based Client Identification," Proc.Fourth USENIX Symp. Networked Systems Design and Implementation (NDSS '07), 2007.
[5] S. Chiasson, P.C. van Oorschot, and R. Biddle, "A Usability Study and Critique of Two Password Managers," Proc. USENIX Security Symp., pp. 1-16, 2006.
[6] D. Florencio, C. Herley, and B. Coskun, "Do Strong Web Passwords Accomplish Anything?," Proc. USENIX Workshop Hot Topics in Security (HotSec '07), pp. 1-6, 2007.
[7] K. Fu, E. Sit, K. Smith, and N. Feamster, "Dos and Don'ts of Client Authentication on the Web," Proc. USENIX Security Symp.pp. 251-268, 2001.
[8] P. Hansteen, "Rickrolled? Get Ready for the Hail Mary Cloud!,"http://bsdly.blogspot.com/2009/11/rickrolled-get-ready-forhail-mary.html, Feb. 2010.
[9] Y. He and Z. Han, "User Authentication with Provable Security against Online Dictionary Attacks," J. Networks, vol. 4, no. 3,pp. 200-207, May 2009.
[10] T. Kohno, A. Broido, and K.C. Claffy, "Remote Physical Device Fingerprinting," Proc. IEEE Symp. Security and Privacy, pp. 211-225,2005.
[11] M. Motoyama, K. Levchenko, C. Kanich, D. Mccoy, G.M. Voelker, and S. Savage, "Re: CAPTCHAs Understanding CAPTCHA Solving Services in an Economic Context," Proc. USENIX Security Symp., Aug. 2010.
[12] C. Namprempre and M.N. Dailey, "Mitigating Dictionary Attacks with Text-Graphics Character Captchas," IEICE Trans. Fundamentals of Electronics, Comm. and Computer Sciences, vol. E90-A, no. 1,pp. 179-186, 2007.
[13] A. Narayanan and V. Shmatikov, "Fast Dictionary Attacks on Human-Memorable Passwords Using Time-Space Tradeoff," Proc.ACM Computer and Comm. Security (CCS '05), pp. 364-372, Nov.2005.
[14] Nat'l Inst. of Standards and Technology (NIST), http://www.itl.nist.gov/div897/sqg/dads/HTML/hashbelt.html, Hashbelt., Sept.2010.
[15] "The Biggest Cloud on the Planet Is Owned by ... the Crooks," NetworkWorld.com., http://www.networkworld.com/community/node/58829, Mar. 2010.
[16] J. Nielsen, "Stop Password Masking," http://www.useit.com/alertbox/passwords.html, June 2009.
[17] B. Pinkas and T. Sander, "Securing Passwords against Dictionary Attacks," Proc. ACM Conf. Computer and Comm. Security (CCS '02),pp. 161-170, Nov. 2002.
[18] D. Ramsbrock, R. Berthier, and M. Cukier, "Profiling Attacker Behavior following SSH Compromises," Proc. 37th Ann. IEEE/IFIP Int'l Conf. Dependable Systems and Networks (DSN '07), pp. 119-124,June 2007.
[19] SANS.org, "Important Information: Distributed SSH Brute Force Attacks," SANS Internet Storm Center Handler's Diary, http://isc.sans.edu/diary.html?storyid=9034, June 2010.
[20] "The Top Cyber Security Risks," SANS.org, http://www.sans.org/top-cyber-security-risks/, Sept. 2009.
[21] C. Stoll, The Cuckoo's Egg: Tracking a Spy through the Maze of Computer Espionage. Doubleday, 1989.
[22] "Botnet Pierces Microsoft Live through Audio Captchas,"TheRegister.co.uk, http://www.theregister.co.uk/2010/03/22/microsoft_live_captcha_by pass/, Mar. 2010.
[23] P.C. van Oorschot and S. Stubblebine, "On Countering Online Dictionary Attacks with Login Histories and Humans-in-the-Loop," ACM Trans. Information and System Security, vol. 9, no. 3,pp. 235-258, 2006.

© 2014, IJCSMC All Rights Reserved

[24] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA:Using Hard AI Problems for Security," Proc. Eurocrypt, pp. 294-311, May 2003.
[25] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords," Proc. 17th ACM Conf. Computer and Comm.Security, pp. 162-175, 2010.
[26] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber,"How Dynamic Are IP Addresses?," SIGCOMM Computer Comm.Rev., vol. 37, no. 4, pp. 301-312, 2007.
[27] J. Yan and A.S.E. Ahmad, "A Low-Cost Attack on a Microsoft CAPTCHA," Proc. ACM Computer and Comm. Security (CCS '08),pp. 543-554, Oct. 2008.
[28] J. Yan and A.S.E. Ahmad, "Usability of CAPTCHAs or Usability Issues in CAPTCHA Design," Proc. Symp. Usable Privacy and Security (SOUPS '08), pp. 44-52, July 2008.

© 2014, IJCSMC All Rights Reserved