On Countering Online Dictionary Attacks with Login Histories and Humans-in-the-Loop

PAUL C. VAN OORSCHOT
Carleton University
and
STUART STUBBLEBINE
Stubblebine Research Labs

Automated Turing Tests (ATTs), also known as human-in-the-loop techniques, were recently employed in a login protocol by Pinkas and Sander (2002) to protect against online password-guessing attacks. We present modifications providing a new history-based login protocol with ATTs, which uses failed-login counts. Analysis indicates that the new protocol offers opportunities for improved security and user friendliness (fewer ATTs to legitimate users) and greater flexibility (e.g., allowing protocol parameter customization for particular situations and users). We also note that the Pinkas–Sander and other protocols involving ATTs are susceptible to minor variations of well-known middle-person attacks. We discuss complementary techniques to address such attacks, and to augment the security of the original protocol.

Categories and Subject Descriptors: K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*; K.4.4 [**Electronic Commerce**]: Security

General Terms: Security, Human Factors

Additional Key Words and Phrases: Mandatory human participation schemes, online dictionary attacks, password protocols, relay attack, usable security

1. INTRODUCTION

The abuse, by automated computer programs, of Internet interfaces originally intended for humans has rekindled interest in tests designed to distinguish humans from computers [Turing 1950]. The specific goal, however, is now somewhat different: to ensure human involvement in a broad range of

A preliminary version of parts of this paper appeared in Stubblebine and van Oorschot [2004].
Version: 14 March 2006.
The first author is supported by NSERC under a Discovery Grant and as Canada Research Chair in Network and Software Security; the second is supported under NSF grant DMI-0339464.
Authors' addresses: P. C. van Oorschot, Carleton University, School of Computer Science, Ottawa, Canada; email: paulv@scs.carleton.ca; Stuart Stubblebine, Stubblebine Research Labs, New Jersey; email: stuart@stubblebine.com.

computer-based interactions. The idea, first proposed by Naor [1997], is to find simple tasks relatively easily performed by humans, but apparently difficult or infeasible for automated programs, such as, visually recognizing distorted words. Mechanisms involving such techniques have been called *mandatory human participation schemes*, *human-in-the-loop protocols*, and *Automated Turing Tests* (ATTs); see von Ahn et al. [2003, 2004].

Among others, one specific purpose for which such tests have been proposed is protecting web sites against access by automated scripts. ATTs are currently being used to protect against database queries to domain registries, to prevent sites from being indexed by search engines, and to prevent "bots" from signing up for enormous numbers of free email accounts [von Ahn et al. 2004]. They have also been proposed for preventing more creative attacks [Byers et al. 2004].

In this paper, we are primarily interested in the use of ATTs to protect web servers against online password-guessing attacks (e.g., online dictionary attacks). The idea is that automated attack programs will fail the ATT challenges. A specific instance of such a protocol was recently proposed by Pinkas and Sander [2002]. While this protocol appears to be quite simple, closer inspection reveals it to be surprisingly subtle and well-crafted. Simpler techniques preventing online dictionary attacks are not always applicable. For example, account lock-out after a small number of failed password attempts may result in unacceptable side effects, such as increased customer service costs for additional telephone support related to locked accounts and new denial of service vectors via intentional lock-out of other users [Wolverton 2002]. Another standard approach is to use successively longer delays as the number of successive invalid password attempts on a single account increases. This may lead to similarly unacceptable side effects.

1.1 Our Contributions

In this paper, we modify the protocol of Pinkas and Sander, presenting a new history-based protocol with ATTs. Our enhancements include the use of failed login counts and distinguishing between "owner" and "non-owner" modes (corresponding roughly to the use of a more trusted device, e.g., a user's regular machine versus one used temporarily in an Internet café). Our analysis indicates that the new protocol offers opportunities for improved security and user-friendliness (e.g., fewer ATTs to legitimate users) and greater flexibility (e.g., allowing protocol parameter customization for particular situations and users). We also note that many ATT-based protocols, including that of Pinkas and Sander, are vulnerable to an *ATT relay attack*: ATT challenges may be relayed to unsuspecting parties, who generate responses, which are then relayed back to the challenger. We explore this threat and mechanisms to address it, and propose additional (orthogonal) enhancements to Pinkas–Sander type protocols. We discuss complementary techniques to address such attacks and to augment the security of the original protocol.

1.2 Organization

The sequel is organized as follows. Section 2 discusses background context and assumptions, including a reference version of the basic ATT-based login

```
1  fix a value for system parameter p, 0 < p ≤ 1 (e.g., p = 0.10)
2  user enters userid/password
3  if (user device has cookie) then server retrieves it
4  if (entered userid/password pair correct) then
5     if (cookie present & validates & unexpired & matches userid) then
6        login passes
7     else % i.e. cookie failure
8        ask an ATT; login passes if answer correct (otherwise fails)
9     endif
10 else % i.e., incorrect userid/password pair
11    set AskAnATT to TRUE with probability p (otherwise FALSE) †
12    if (AskAnATT) then
13       ask an ATT; wait for answer; then say login fails
14    else
15       immediately say login fails
16    endif
17 endif
```

† This setting is a deterministic function of the userid/password pair [Pinkas and Sander 2002]

Fig. 1. Original ATT-based login protocol (simplified description).

protocol. Section 3 presents a new variation, with enhancements aimed toward usability, security against online dictionary attacks, and parameter flexibility. Section 4 provides analysis of security and, briefly, also usability. Section 5 pursues per-account parameter customization to improve usability, based on historical user and system statistical profiles. Section 6 presents an ATT relay attack and discusses standard techniques to both address it and to augment the security of the original protocol. Section 7 provides further background and a summary of related work. Section 8 contains concluding remarks.

2. BACKGROUND, CONSTRAINTS, ASSUMPTIONS, AND OBJECTIVES

For reference, Figure 1 provides a simplified description of the original ATT-based login protocol (for full details, see Pinkas and Sander [2002]). The system parameter p is a probability, which determines the fraction of time that an ATT is asked, in the case that an invalid userid–password pair is entered. In the case of a successful login (Figure 1, line 8), the protocol stores a cookie on the machine from which the login occurred; the cookie contains the userid (plus optionally an expiration date), and is constructed in such a way (e.g., using standard techniques involving symmetric-key encryption or a MAC) that the server can verify its authenticity.

For context, we next state a few assumptions and observations relevant to both the original and new protocols. We begin with a basic constraint.

- *Constraint: Account lock-out not tolerable.* We are interested in protocols for systems where locking-out of user accounts after some number of failed login attempts is not a viable option. (Otherwise, online login attacks are easily addressed; see Section 1.)
- *Trust model assumptions: Trusted client and ephemeral memory.* We assume that client computers, and any resident software at the time of use, are nonmalicious (e.g., free of keyboard sniffers and malicious software).

This is standard for (one-factor) password-based authentication protocols—otherwise, the password is trivially available to an attacker. For similar reasons, we assume client software leaves no residual data on user machines after a login protocol ends (e.g., memory is cleared as each user logs out). In practice, it is difficult to guarantee these assumptions are met (e.g., for borrowed machines in an Internet café); but without them, the security of almost all password protocols seems questionable.

- *Observation 1: Limited persistence by legitimate users.* A typical legitimate user will give up after some maximum (e.g., $C = 10$) of failed logins over a fixed time period, after which many will check with a system administrator, colleague or other source for help, or simply stop trying to log in. Large numbers of successive failed logins, if by a legitimate user, may signal a forgotten password or a system availability issue (here login failures are likely not formally recorded by the system); or may occur because of an attacker, as either a side effect of attempting to crack passwords, or intentionally for denial-of-service in systems susceptible to such tactics.
- *Observation 2: Users will seek convenience.* If a login protocol is necessary to access an online service, and users can find a similar alternate service with a more convenient login (though possibly less secure), then many users will switch to the alternate service. User choices are rarely driven by security; usability is usually a far greater factor, and poor usability typically leads to loss of business.

These observations lead us to our usability goal; we state it informally.

- *Usability goal: Minimal user inconvenience.* Relative to standard userid–password schemes, we wish to minimize additional inconvenience experienced by a user.
  As is often the case, the usability goal must be met in a tradeoff with security, and we have a two-part security goal. One part is protecting specific accounts (e.g., certain users may be more concerned about, or require more, protection; or a service provider may worry more about specific accounts—say those with high sales ratings or high account values). The second is protecting all accounts in aggregate (e.g., a web service provider might not want *any* user accounts misappropriated to host stolen software; a content service provider might want to protect access to content available to authorized subscribers).
- *Security goal: Control access to both specific accounts and nonspecific accounts.* Constrain information the adversary learns from trial password guesses before being "stopped" by an ATT challenge in the context of fully-automated attacks directed toward a specific account (single-account attack) and toward any account (multi-account attack).

In practice, for authentication schemes based on user-selected passwords, prevention of unauthorized access cannot be 100% guaranteed for a specific account or all accounts in aggregate, because of the nonzero probability of correctly guessing a password, and the ubiquity of poor passwords. Nonetheless, the quality of a login protocol may be analyzed independent of particular password choices and this is what we pursue. For a given password, we are interested

```
1    fix values for 0 < q ≤ 1 (e.g., q = 0.05 or 0.10) and integers b1, b2 ≥ 0
2    user enters userid/password
3    if (user device has cookie) then server retrieves it
4    if (entered userid/password pair correct) then
5      if (cookie present & validates & unexpired & matches userid) then
6        login passes
7      else % i.e., cookie failure
7.1      if OwnerMode(userid) OR (FailedLogins[userid] ≥ b1) then
7.2        ask an ATT; login passes if answer correct (otherwise fails)
7.3      else
7.4        login passes
7.5      endif
9      endif
10   else % i.e., incorrect userid/password pair
11     set AskAnATT to TRUE with probability q (otherwise FALSE) †
11.1   if (AskAnATT) OR (FailedLogins[userid] ≥ b2) then
13       ask an ATT; wait for answer; then say login fails
14     else
15       immediately say login fails
16     endif
17   endif
  † This setting is a deterministic function of the userid/password pair [Pinkas and Sander 2002]
```

Fig. 2. New Protocol (History-based Login Protocol with ATTs). OwnerMode(userid) is a boolean (e.g., table-lookup into a bit-array), returning TRUE if userid is in owner mode; its update is not shown. *FailedLogins*[*userid*] is set to the userid's number of failed logins in a recent period T, and updated (also not shown). (see Section 3).

in how effectively a given protocol allowing online interaction prevents extraction of password-related information. Intuitively, "as little information as possible" should be leaked; while a more rigorous definition is desirable, one is not provided herein.

Requiring mandatory human participation increases the level of sophistication and resources for an attack. If ATTs are effective and ATT relay attacks are countered (e.g., by means such as embedded warnings—see Section 6.2), then constraining information leaked before being "stopped" by an ATT challenge is an important security characteristic of a password-based login protocol.

3. HISTORY-BASED LOGIN PROTOCOL WITH ATTS (NEW PROTOCOL)

Here we modify the original protocol, intending to both improve the user experience and increase security, e.g., to increase the percentage of time that an adversary is challenged with an ATT, without further inconveniencing legitimate users.[1] The modifications do not themselves prevent ATT relay attacks (Section 6.1), but are complementary to other modifications in Section 6 which do, and can thus be combined.

We assume familiarity with the original protocol (Figure 1). Linewise, the new protocol (Figure 2) differs as follows: lines 7.1–7.5 replace 8 and line 11.1

[1]One might try to improve usability by allowing a small number of trial passwords per userid without triggering an ATT. While this reduces security only in a minor way for a single-account attack (Section 4.2), the problem is greater with multi-account attacks (Section 4.3).

replaces 12. The new protocol with failed-login thresholds ($b_1 = 0, b_2 = \infty$) behaves the same as the original protocol.

In what follows, we provide detailed discussion of differences between the new and original protocols, including, cookie-handling (cookies are now stored only on "trustworthy" machines); owner and non-owner mode; per-user tracking of failed logins; and setting failed-login thresholds. The idea of dynamically changing failed-login thresholds has been previously suggested [Pinkas and Sander 2002]; we detail a concrete proposal and comparison. We emphasize that both protocols assume the use of a deterministic function of the userid–password pair for determining AskAnATT in line 11. In other words, for a fixed userid, a particular password will either always result in an ATT being asked, or never.

3.1 Handling Cookies

As noted earlier, the original protocol stores a cookie on any device after successful authentication; the new protocol does not. Optional user input controls cookie storage similar to web servers using a login page checkbox asking if users want to "remember passwords," e.g., "Is this a trustworthy device you use regularly, such as your home or office machine? YES/NO." This part of the page appears if no cookie is received by the server. Upon a YES response, a cookie is pushed to the user device only after the user successfully authenticates (requiring a successful ATT response, if challenged) at Figure 2, lines 7.2 and 7.4. This cookie approach reduces exposure to cookie theft versus the original protocol, with negligible usability downside because the question appears on the same screen as the login prompt; we recommend default answer NO, as now explained.

A default of NO reduces exposure to cookie theft; however, if users blindly accept the default, then more accounts (than otherwise) will be in non-owner mode, and there would thus be greater vulnerability to multi-account dictionary attack (see Section 4). Conversely, a default of YES, if blindly accepted by users, may increase exposure to cookie theft, but reduce the number of accounts in non-owner mode and thus decrease susceptibility to multi-account dictionary attacks, thereby improving security provided that cookie theft does not occur. Which is preferable depends on the probability (threat) of stolen cookies. If no default is given (forcing users to make their own choice), one might hope that users make the correct choice, but usability is negatively impacted.

Cookies include the following information (cf. Section 2): expiry time, userid account name, and cookie identifier (for tracking failed-login attempts with a particular cookie; see Section 3.3 below, and Section 4.4.2 under COOKIE THEFT). Cookies are associated with a server, and are integrity-protected by a MAC (message authentication code) under a symmetric key known only to server. For background on using browser cookies securely, see Fu et al. [2001].

The original protocol requires that cookies be tracked by the server and ignored after exceeding a limit on failed login attempts with the particular cookie (e.g., 100 [Pinkas and Sander 2002]). We follow a similar approach. Each time a login fails (e.g., lines 7.2, 13, and 15), we increment a server-side failed login

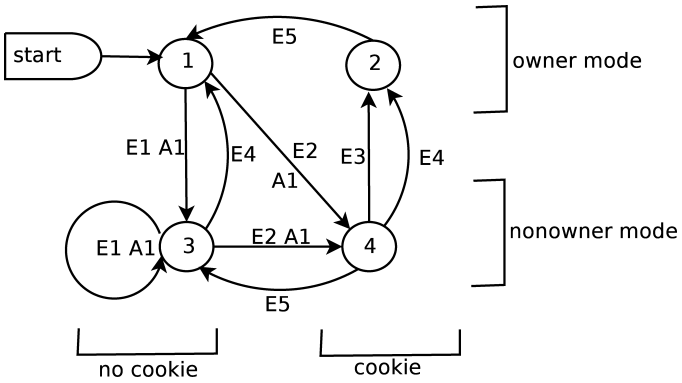

Fig. 3. Transition diagram for owner to non-owner mode. State definitions: 1, owner mode, no cookie; 2, owner mode, with cookie; 3, non-owner mode, no cookie; 4, non-owner mode, with cookie. Event definitions: E1, successful login, no cookie, NO (device is not trusted); E2, successful login, no cookie, YES (device is trusted); E3, successful login with valid cookie; E4, countdown timer reaches 0; E5, cookie expires or is lost (not saved). Action definitions: A1, reset countdown timer to W.

count associated with the cookie, if a valid cookie was received. If the cookie exceeds a failed-login threshold, a server-side flag is set to subsequently ignore this cookie (for the remainder of its normal validity period). The line 5 check that the cookie validates includes both a check of this flag and an authenticity check (e.g., cookie MAC verification). The *cookie failure threshold* is the number of failed logins allowed before a cookie is invalidated. We recommend setting this to $\min(b_1, b_2)$; see Section 4.4.

3.2 Definition of Owner and Non-owner Mode

A user is more likely to login from "nonowned" devices when traveling (e.g., borrowing an Internet access device in a library, guest office, conference room, Internet café). We also, expect that a user submitting a login request without valid cookie is more often using a nonowned device (and less often, logging in from an owned device, e.g., initially or after cookie expiry). As a consequence of how cookies are handled, we can assume (and be correct more often than not) that a user is on a nonowned device if their most recent successful login is cookieless. We initially set a user account to be in what we call *owner* mode and expect an account to primarily be in owner mode if most of the time the user uses their regular device (e.g., one of the devices they own). An account transitions to *non-owner* mode when a login is successfully authenticated without the server receiving a valid cookie (Figure 2, line 7.4), and returns to owner mode after a specified time-out period W (e.g., 24 hours) or a successful login with cookie present. The time-out period is restarted, and the account remains in non-owner mode, if another cookieless successful login occurs. The time-out period reduces the number of accounts in non-owner mode, which lowers the security risk; accounts in non-owner mode are more susceptible to multi-account dictionary attacks (see Section 4.3). A state transition diagram for owner to non-owner mode is given in Figure 3. On event E2, the server attempts to download a cookie to the client.

3.3 Tracking Failed Logins

We define *FailedLogins*[*userid*] to be the number of failed-login attempts for a specific userid within a recent period T (e.g., 30 days). Here *failed-login attempts* includes: nonresponses to ATT challenges, incorrect responses, failed userid–password pairs, and outstanding authentication attempts (e.g., the adversary may simultaneously issue multiple login attempts; one strategy might be to issue a very large number, and respond to only a subset of resulting ATT challenges, perhaps being able to exploit some "weak subclass" of ATTs for which computer-generated responses are feasible).

3.4 Setting the Failed-Login Thresholds (Bounds b_1, b_2)

Low values for b_1, b_2 maximize security at the expense of usability (e.g., for users who frequently enter incorrect passwords). A reasonable bound may be $b_1, b_2 \leq 10$ (perhaps larger for large T). In the simplest variation, the protocol bounds b_1, b_2 are fixed-system variables; in a more elaborate design (cf. Section 5), they are dynamic and/or set on a per-user basis (varying for a particular userid, based on a history or profile and possibly subject to system-wide constraints, e.g., maximum bound on b_2). For example, certain users who regularly enter a password incorrectly might be given a higher failed-login threshold (to increase usability) compared to users who almost always enter correct passwords. If it is expected or known from a historical profile that a user will log in L times over a period T, and that say 5% of legitimate login attempts fail, then b_2 might be set somewhat larger than $(0.05) * L$ (e.g., $T = 30$ days, $L = 100$, $b_2 = 5$). Over time, per-user rates of legitimate failed logins (e.g., mistyped or forgotten/mixed up passwords, perhaps more frequent on unfamiliar machines) can be used to establish reasonable thresholds. To simplify presentation, updating of per-user table entries *FailedLogins*[*userid*] is not shown in Figure 2. Note that while per-user values require server-side storage when these values cannot be user-stored via cookies, a small amount of per-user server-side storage is already required in both the original and new protocol to ameliorate cookie theft (see above).

As a further option, setting the ATT challenge probability q on a per-user basis also allows flexibility for tuning usability and security on a per-account basis.

As with any security-critical application, care should be taken to implement the new protocol in a manner resilient to denial of service attacks, e.g., by an adversary attempting to exhaust any dynamically allocated per-userid state memory.

4. COMPARATIVE ANALYSIS—SECURITY AND USABILITY

In this section we provide a comparative analysis of the new and original protocols. We begin with a security analysis against single-account attacks and then consider multi-account and other attacks, and usability.

4.1 Assumptions

We generally follow the original assumptions [Pinkas and Sander 2002], including the following. Passwords are from a fixed set (dictionary) of cardinality N,

Table I. Summary of Comparative Security Analysis (Single-Account Attack)

		New Protocol	
		Account Mode	
Question	Original Protocol	Owner	Non-owner
Q1	$(1-p)N$	$(1-q)b_2$	$\max(b_1,(1-q)b_2)$
Q2	$\frac{1}{2}pN$	$\frac{1}{2}(N-(1-q)b_2) \approx N/2$	$\frac{1}{2}(N-\max(b_1,(1-q)b_2))$
Q3a ($c=0$)	0	0	b_1/N
Q3b ($c=1$)	$1/pN$	$(1-(1-q)^{b_2+1})/qN$	$(b_1+1)/N$
Q3c ($c \geq 2$)	c/pN	$\leq \min(\frac{c}{q}, b_2+c)/N$	$(b_1+c)/N$

and for analysis purposes, are equiprobable (a more precise analysis, with differing password probabilities, follows a similar approach, but password probability distributions are generally unknown). For a more realistic model, one could restrict the password space (and N) to that subset of the full space that contains passwords of probability above some bound, providing a more representative "effective password space." The delay from the time of password entry to the time an ATT challenge is received should not depend on whether the entered password is correct. Login is assumed to be via a browser, with cookies enabled (the actual base requirement is simply that the server have a reliable way of authenticating computing devices [Pinkas and Sander 2002]). Probabilities p and q are as defined in the protocols herein. ATTs are asked for a fraction (p or q) of incorrect, as well as the single correct, password. We make an additional simplifying assumption: a particular account remains in either owner or non-owner mode.

Below, we first assume no cookie theft (an attacker knows a userid but has no corresponding cookie, so that a correct password guess puts the attacker into the cookie failure block at line 7 in the new protocol) but consider the implications of theft in Section 4.4.

4.2 Security Analysis for Single-Account Attacks

To drive our comparative security analysis, we ask the following questions for the original and new protocols, for an attack on a single account.

Q1: What is the expected number of passwords that an attacker can eliminate from the password space, without answering any ATTs?

Q2: What is the expected number of ATTs an attacker must answer to correctly guess a password?

Q3: What is the probability of a confirmed correct guess for an attacker willing to answer c ATTs?

Table I summarizes the answers based on the best-attack strategies known to the authors (see next paragraph), and the assumptions noted above. Also, to the benefit of the attacker, we assume that failed login counts b_1 and b_2 are 0 at the start of an attack. In the table, q is used for p in the new protocol to emphasize that $q \neq p$ is allowed.

For Q1, in a first pass, an attacker of the original protocol may simply try all passwords in an attack dictionary, and quit on each guess triggering an ATT (in total, a p-fraction of the dictionary); all others, i.e., $(1-p)N$ passwords,

result in a protocol response confirming password incorrectness without the cost of answering an ATT, and thus can be eliminated at a cost of zero ATTs. In the new protocol, the number of "free" (i.e., without ATT) guesses in owner mode is certainly limited by the failed-login threshold b_2, after which an ATT is required for each guess; but for a q-fraction of these b_2 guesses, one expects an ATT triggers at line 13, which to the attacker is indistinguishable from a line 7.2 ATT—thus these passwords cannot be safely discarded as incorrect. Thus the expected number of eliminatable passwords here is actually $(1 - q)b_2$, as noted in the table. Similar reasoning yields the table entries for Q1 and Q2 in non-owner mode.

For Q2, an attacker would, on average, be expected to guess the correct password after going through one-half the remaining (noneliminated) passwords and each of these guesses has a cost of one ATT. Thus, in the new protocol for an owner mode account, after eliminating $(1 - q)b_2$ passwords, an attacker is expected to try one-half the remaining $N - (1 - q)b_2$ passwords, at a cost of $(N - (1 - q)b_2)/2$ ATTs, before hitting the correct password. Similar reasoning gives the Q2 table entry for the original protocol.

For Q2 and Q3 against both the original protocol and owner-mode accounts in the new protocol, for an attacker who measures cost in terms of the number of ATTs that must be answered, there appears to be no penalty in simply answering the first c ATTs asked (without eliminating bad passwords in a preliminary "zero ATT cost" pass). The reason is as follows. In the original protocol, ATTs are asked on only a p-fraction of the password space, which includes the correct password; upon being asked an ATT, there is a fixed probability that the password candidate is correct. For the new protocol, the same holds, but, in addition, each nonanswered ATT increases the failed-login counter, which shortens the number of remaining guesses available (if any) before triggering an ATT on every single guess. In essence, an attacker trying a candidate password D gains no benefit from failing to pursue D upon getting an ATT (unless he has no intention of answering *any* ATTs, which is a valid multi-account attack strategy against non-owner mode accounts—see Section 4.3); rather, the ATT should be answered, since D may, indeed, be the one correct password and subsequent trials of the same password W will also trigger an ATT.

The answer to Q3 for the original protocol is c/pN. This follows from being able to narrow the password space down to pN candidates, anc c ATTs allowing the fraction c/pN of this space to be covered. For $c = 0$ in owner mode of the new protocol, the probability is 0 of correctly guessing a password without answering an ATT, since an ATT is always asked if the correct password is tried without a valid cookie. For $c = 0$ in non-owner mode, an attacker has b_1 "free" guesses before the failed-login threshold kicks in at line 7.1 and any ATT asked during the first b_1 trials necessarily results from line 13, confirming an incorrect password; furthermore any password guess after trial b_1 cannot be confirmed as correct, since line 7.1 would trigger an ATT, which the attacker is not willing to answer. This gives a Q3a probability of b_1/N with 0 ATTs and similarly for c ATTs a probability of $b_1 + c/N$. The remaining entries in the table, namely Q3b ($c = 1$) and Q3c ($c \geq 2$) for owner mode of the new protocol, require further explanation, as given by the following Lemmas.

LEMMA 1. *Let R be the probability of a confirmed correct guess for an attacker willing to answer $c = 1$ ATT in owner mode of the new protocol. Then $R = (1 - (1 - q)^{b_2+1})/qN$.*

PROOF. Arguing as above with an attacker simply answering the first ATT asked, the number of trial password guesses is limited to $b_2 + 1$, since testing bound b_2 at line 11.1 guarantees a second ATT if the game ever proceeds beyond b_2+1 trial guesses, with each such trial thus having zero probability of success. Thus, R is the sum of the probabilities of a confirmed correct guess over the first $b_2 + 1$ trials. Let c_i be the probability of a correct guess upon reaching trial i, and let e_i be the probability the event trial i occurs. Then $R = \sum_{i=1}^{b_2+1} c_i \cdot e_i$. Now for all i, it follows from simple probability tree (or other) arguments that $c_i = 1/N$. Regarding e_i, for trial i to occur, line 13 must have been avoided on all previous iterations (otherwise the single ATT would have been consumed); this has probability $(1-q)^{i-1}$. Thus $R = (1/N)\sum_{i=1}^{b_2+1}(1-q)^{i-1} = (1-(1-q)^{b_2+1})/qN$, as claimed. □

Note 1. From Lemma 1, if $b_2 = 0$ then $R = 1/N$; this yields a smaller (better) probability than the corresponding $1/pN$ in the original protocol. Moreover, the bound b_2 check in line 11.1 ensures, at most, $b_2 + 1$ trials can pass before consuming the $c = 1$ available ATT, so for any value b_2, $R \leq (b_2 + 1)/N$ (at most, $b_2 + 1$ of N passwords can be guessed, using in total one ATT). Finally, as $b_2 \to \infty$, $(1 - q)^{b_2+1} \to 0$ for $q > 0$, and thus $R \to 1/qN$, as in the original protocol; this is as expected, since $b_2 \to \infty$ means the bound b_2 becomes unused. Combining these we have $R \leq \min(1/qN, (b_2+1)/N)$, with the first term $1/qN$ taking precedence when $q \geq 1/(b_2 + 1)$, in which case larger probabilities q lower R.

LEMMA 2. *Let R be the probability of a confirmed correct guess for an attacker willing to answer $c \geq 2$ ATTs in owner mode of the new protocol, using an attack strategy of answering the first c ATTs asked. Then $R \leq min(\frac{c}{q}, b_2+c)/N$.*

PROOF. Similar to reasoning in the proof of Lemma 1, checking (only) the threshold b_2 in line 11.1 implies that, at most, $b_2 + c$ passwords can be guessed before c ATTs are consumed. (In fact one would expect fewer trials, because of the probability q in line 11 also contributing to the consumption.) This yields the $(b_2 + c)/N$ term. The c/qN term follows from the fact that the probability q (alone) at line 11 would cause line 13 to be expected to occur every $1/q$ trials, and, thus, in the absence of a b_2 bound, one would expect line 13 to consume c ATTs after c/q trials, allowing the attacker to guess c/q of the N password candidates, giving an expected success probability of c/qN. The interaction between these two possible outcomes, each of which consumes ATTs independently, gives the claimed upper bound. □

Note 2. From Lemma 2, the second term $(b_2 + c)/N$ takes precedence provided $q \leq c/(b_2 + c)$, which we would expect for many practical parameter selections (when $c \neq 0$). For a bound tighter than that given by Lemma 2 and better understanding of the protocol, we consider a game between the algorithm and the attacker, with the attacker willing to "spend" c ATTs, answering each ATT

when asked. The question is to find g, the expected number of trial password guesses by the attacker, up to and including the one triggering the last (cth) ATT; it then follows that the probability of a successful password guess is g/N. We find it helpful to consider two cases. Case 1: the cth ATT is consumed before the point at which the bound b_2 triggers an ATT at every single iteration of line 11.1—this will happen for $c/q \leq b_2$, probably an uncommon parameter choice (e.g., $q = 0.1$, $c = 2$, $b_2 = 25$). Case 2: the cth ATT is consumed after trial b_2, but within the first $b_2 + c$ trials.[2]

Determining g, and hence the answer to Q3c (new protocol, owner mode), thus motivates the question: what is the number x of trials (password guesses) expected before consuming the cth ATT? Under our assumptions discussed earlier, an ATT results from line 11 of the new protocol with independent probabilility q on each trial, assuming an attacker randomly selects (different) password candidates. This is a standard negative binomial distribution question, with the random variable X being the number of "failures" (trials where no ATT is asked), the experiment continued until a fixed number c "successes" occur (an ATT is asked), the probability of success and failure, respectively, being q and $1 - q$ and our interest being in the expected value of $g = x + c$, for the probability function $f(x) = C(x+c-1, x) \cdot q^c(1-q)^x$ for $x = 0, 1, \ldots$. From this a precise expression for R in Lemma 2 can be given, but a general closed form is not evident; thus, we find the bound given by Lemma 2, and the discussion above, suffices.

Note 3. Rows Q1 and Q2 of Table I indicate that the number of passwords that an attacker can eliminate for free (i.e., without any ATTs) is substantially greater in the original protocol.[3] A second observation favoring the new protocol follows from rows Q3b and Q3c: the probability of a successful attacker guess in the new protocol (on the order of $1/N$) is generally significantly smaller than in the original (on the order of $1/pN$), except that when b_2 is relatively large the new protocol effectively behaves as in the original, with probability c/qN matching the table entry c/pN for the original protocol; when b_2 is small, $b_2 + c$ is less than c/q, so the probability in the new protocol is better, i.e., less than in the original.

Note 4. Row Q3a indicates that for an attacker unwilling to answer any ATTs, both protocols have the same security, except that we relax security in the new protocol (i.e., to b_1/N) for those accounts in non-owner mode (presumably a relatively small number), to improve usability as indicated in Table II (bottom row). Especially given the latter, we view this as an acceptable risk, in general, for a single-account attack.

---

[2]After trial number $b_2 + c$, a correct password guess cannot be confirmed, as all c ATTs have been consumed. As per Note 2, b_2 plays a role in limiting the number of password guesses iff $c/q \geq b_2 + c$.

[3]For the new protocol, these figures are per time period T. However, for a sophisticated multi-period attack, the new protocol remains better (fewer passwords are eliminatable), assuming $p = q$, unless at least N/b_2 time periods are used (e.g., about 1600 years for $T = 1$ month, $N = 100\,000$ and $b_2 = 5$).

Table II. Fraction of the Time a Legitimate User Must Answer an ATT (1.0 = 100%)

		New Protocol	
		Account Mode	
	Original Protocol	Owner	Non-owner
Incorrect password	p	$q^{\dagger}$	$q^{\dagger}$
Correct password—valid cookie	0	0	0
Correct password—no valid cookie	1.0	1.0	$0^{\dagger}$

See Note 8.

Note 5. For both $c = 1$ and $c \geq 2$, for the new protocol, Table I gives a probability (i.e., an expectation over a large number of runs) bounding the worst case, and Note 2 discusses which term therein takes precedence. For reasonable values b_2, this probability is substantially better (lower) than the corresponding expected value in the original protocol.

4.2.1 *Summary of Analysis.* Table I summarizes this analysis, with further text explanations in Notes 3, 4, and 5. The new protocol has better security characteristics against single-acount attacks, except in non-owner mode for an attacker unwilling to answer any ATTs, where security is decreased somewhat, but still quite acceptable, as per Note 4. Case Q3c (see Notes 2 and 3) is likely of greatest interest to practitioners.

4.3 Security Analysis for Multi-account Attacks

In multi-account (or system-wide) attacks, an attacker seeks to break into any one of many accounts, not necessarily a specific account. They are generally of greater concern to serviceprovider than individual users and usually more difficult to protect against (than single-account) attacks, since they tend to find "weakest links" across a system or user space.

In the analysis below, we use assumptions from the beginning of Section 4 and here, in addition, assume that an attacker knows m valid userids of the L total user accounts; setting $m = L$ gives the attacker the greatest advantage and, in this sense, is the most general (worst) case—albeit overestimating insecurity in many environments.

1. *Case* $c = 0$ (zero ATTs answered) For the new protocol, consider an attack strategy where the attacker (assumed to have no valid cookie) acquires a list of m valid userids for accounts in non-owner mode (or exhaustively tries userid–password combinations hoping to meet this condition—see Note 6) and makes password guesses. Either a guess is correct ("lucky guess" resulting in line 7.4, Figure 2), is incorrect with immediate failure notice (line 15), or an ATT challenge results (lines 7.2 or 13). The attacker quits all login sessions returning an ATT, moving onto a new password (on the same userid, at most, b_1 times, or another userid). The goal is to reach line 7.4, which requires an account in non-owner mode, below the failed-login bound b_1. For accounts in owner mode, the new protocol performs similarly (from a security perspective) against multi-account attacks as the original protocol.

   For non-owner mode accounts, the new protocol permits an attacker up to b_1 free guesses (per account) before challenging with an ATT on any subsequent

correct password guess (cf. Note 4). Thus, the probability of successful attack may approach (worst case) $m \cdot b_1/N$. In contrast, in the original protocol an ATT challenge occurs every single time a password guess results in line 8 and thus the probability of successful attack (for $c = 0$) is zero. Thus, as emphasized in Section 4.2, in non-owner mode, security has been decreased in a trade-off for improved usability. This security loss can be made arbitrarily small by reducing b_1, e.g., at the extreme setting $b_1 = 0$; this has the effect of artificially putting the account into owner mode, yielding behavior matching that of the original protocol at line 8 in Figure 1.

Note that accounts found, or forced to adhere to a strong password selection policy (e.g., through password checking programs that check the quality of passwords and/or password-generation programs [FIPS PUB 112, 1995; FIPS PUB 181, 1993]) could be assigned a setting $b_1 \geq 1$ or somewhat higher, with $b_1 = 0$ (or very low) for other accounts. Such tuning of b_1 on a per-user basis (see related discussion in Section 5) effectively results in users who chooose "weak" passwords forfeiting usability benefits in non-owner mode. Alternatively, users of accounts, which are often in non-owner mode, could be selectively encouraged to use stronger passwords.

2. *Case* $c \geq 1$ (one or more ATTs answered)[4] For the new protocol with an attacker willing to answer one or more ATTs, the best strategy would again seem to involve making trial guesses on a particular non-owner mode account until reaching the failed-login bound b_1 for that account, and then moving on to another non-owner mode account. This attack does not succeed against accounts in owner mode, since for those, an ATT is demanded (guaranteed) upon reaching line 7.1.

   As discussed earlier (Table I, row Q3c), the new protocol has significantly better security than the original against *single-account* attackers willing to answer $c \geq 1$ ATTs. This security advantage carries over for multi-account attacks (including in situations where the adversary arranges that ATTs be answered, e.g., by relaying them to a "sweatshop"), but only for accounts in owner mode. For non-owner mode accounts, it is thus important to take special precautions as suggested under Case $c = 0$. This leads to Note 6.

Note 6. It is of significant advantage to an attacker of the new protocol to find ways to distinguish owner from non-owner mode accounts and, correspondingly, for the system to prevent this information from being easily available. An attacker unable to do so will face far more ATTs. Related to this, the system ratio of owner to non-owner accounts affects security from the viewpoint of system resistance to multi-account attacks.

To discern modes, an adversary might track login histories of usernames in systems where those usernames are publicly available (e.g., shown as leading bidder in an online auction) and are the same login identifier used as part of authentication. The adversary notes that the timeout period W (see Section 3) implies that usernames with a recent login have a much greater chance of being in non-owner mode than others. A countermeasure is to use publicly available

[4]We expect this case to be of primary interest to practitioners.

identifiers that do not expose actual login identifiers. If it is impossible to hide information that helps an adversary to discern modes, then parameter b_1 can be adjusted lower to force a line 7.1 ATT challenge on a correct user name and password. To guard against an adversary targeting only certain distinguished accounts or account types, b_1 can be adjusted on a per-account (type) basis; this helps balance security and usability.

4.4 Other Attacks

Here we briefly consider several other types of attack.

4.4.1 *Parallel Login Attacks.* An attacker may try to launch a parallel-login attack, simultaneously attempting a login to one userid a large number of times (e.g., 1000) on different servers. (See related definition of failed login attempts in Section 3.) This attack attempts to take advantage of architectures, which involve large number of servers to load-balance activities, in the case of large user spaces and the difficulty of centrally updating failed-login counts (or other authentication state information) across different servers. It can be countered by deterministically routing authentication requests by userid, to a particular server or server farm preassigned to that userid. With such a system design, each such server is able to stay current on updates without excessive login delays.

4.4.2 *Cookie Theft.* The above analysis assumes no cookie theft occurs. Here we make a few observations in case it does.

4.4.2.1 *New Protocol.* If a cookie is stolen, then within the cookie's validity period and while under the recommended cookie failure threshold (see Section 3.1 and Note 7), the attacker can try $\min(b_1, b_2)$ password guesses on the associated userid, without being asked an ATT that cannot be prudently abandoned, using a single-account attack as follows. The attacker guesses passwords, quits all guesses that return an ATT, and hopes to reach line 6 (Figure 2) with a lucky guess. The probability of success is $\min(b_1, b_2)/N$. (This attack may take place in conjunction with one that reduces the password space without answering an ATT, or one where the adversary is willing to answer $c \geq 1$ ATTs.)

4.4.2.2 *Original Protocol.* Similarly, the attacker gets free guesses up to the cookie failure threshold. A correct password guess on any such trial allows successful login with 0 ATT answers.

Note 7. (a) We expect that stolen cookies are less likely with the new protocol, since they would be expected to reside in fewer places, e.g., cookies from the original protocol would show up in Internet cafés. While many devices may be compromised by malicious software exploiting any of thousands of software flaws, under the assumption that the vulnerability to cookie theft is proportional to the number of cookies on devices, the original protocol is more vulnerable than the new to cookie theft. (b) A combined cookie and noncookie attack against a single account has lower probability of success in the new protocol, because of

the failed-login thresholds; moreover, in the original protocol, the attacker can reduce the password space to a p-fraction even before using the stolen cookie (cf. discussion on Q1 and Q2 in Section 4.2).

4.4.3 *Gaming of Failure Thresholds.* Another threat is the "gaming" of historical statistics. A determined attacker, over a long period of time, could possibly skew upward the average failure rate on high-value accounts through occasional intentionally failed login attempts. This might lead administrators to increase thresholds b_1, b_2, aiding the attacker (cf. Section 5.2). (This assumes that an adversary could generally estimate the frequency of account login.) A countermeasure is to alert users of the time(s) of last failed login(s) upon a successful login—a long-known technique. Possibly other information about the login session could be provided as well. Optionally, upon a successful login, user feedback regarding their knowledge of prior login attempts could be used to set the appropriate values for b_1, b_2, and q.

4.5 Discussion of Usability

For comparing usability between the original and new protocols, Table II notes the proportion of time a legitimate user is queried with an ATT on entering a correct or incorrect password, with and without a valid cookie. A case of particular focus for the new protocol is the legitimate "traveling user," who generally operates with an account in non-owner mode and without a valid cookie. The new protocol is significantly more userfriendly to such users. We also believe that such users are typically more likely to enter incorrect passwords (see Note 8) and, therefore, increasing usability in this case is significant as one would expect that "incorrect password" cases occur far less often in owner mode.

Also related to usability—the value of the parameter q may be reduced in the new protocol with little or no loss of security, due to the use of the failed login bound b_2 and depending on its value relative to q (see Table I). This further increases usability in the incorrect password case, independent of the discussion in the paragraph above.

Note 8. Regarding Table II, after the failed-login bound is crossed in the new protocol, in several cases, e.g., on incorrect passwords and correct passwords without valid cookies, ATTs occur more frequently (i.e., 100% of the time after the bound is crossed within period T). However for accounts in owner mode we expect a large number of users select a "Remember password" option (standard in many applications), which stores passwords locally on their regular machines. No failed passwords are expected from such users; but their failed login thresholds may still be crossed due to attacker activities.

5. TUNING PROTOCOL PARAMETERS WITH USER PROFILES

We enhance the basic version of the new protocol by adjusting protocol parameters based on historical user profiles (cf. Section 3), i.e., statistics derived from prior login attempts. This allows per-account tailoring of security and usability, improving overall system security against multi-account attacks, and reasonable usability adjustments for different types of users.

5.1 Single and multi-account Historical Statistics

In the definitions of characteristics comprising the historical user profiles below, we use the following terms. *Login session* means a sequence of one or more login attempts to a particular account within some fixed time window and which can reasonably be attributed to the same source, e.g., originating from the same network address, or supplying the same cookie, within a 5-minute time period. *Successful-login session* means a session resulting in account access, whereas a *failed-login session* does not. Suggested account characteristics comprising an account profile include the following.

1. *Password failure rate.* An account's *password failure rate* is the average number of failed login attempts per successful login session. Excluding failed-login sessions here limits an attacker's ability to manipulate password failure profiles by submitting invalid passwords to user accounts. As one variation, this statistic is tracked separately for owner and non-owner mode, to allow tuning if users employ mechanisms for reliable password entry on owned machines (e.g., browser auto-fill features). As a second, this statistic is tracked separately for valid cookie and cookieless successful sessions, i.e., separating those sessions which involve receipt of a valid cookie.
2. *Borrowing rate.* An account's *borrowing rate* is the ratio of successful login sessions without submitting a valid cookie to those with a valid cookie.
3. *Group failed login count.* For a set of accounts, the *group failed-login count* is the total number of failed logins for the user accounts in question, over some time window, associated with failed login sessions involving no valid cookie. The set of accounts may, e.g., be the entire account space, a statistical sampling, or some number of subsets of accounts viewed as attractive targets (e.g., financial accounts of high monetary value, or accounts recently active in an online auction).

The above statistics are computed over (one or more) reasonably chosen recent time windows, long enough to model the predominant failure modes. For example, to account for password failures because of infrequent logins (with users simply forgetting passwords), the window for password failure rate could be set on the order of months or more.

5.2 Protocol Enhancements Using Historical Statistics

As mentioned earlier, we now suggest four custom enhancements (C1–C4) to usability and security, by dynamically adjusting protocol parameters based on historical profiles.

C1: in absence of a valid cookie, set b_1 proportional to the password failure rate for invalid cookies. (This is for non-owner mode accounts only.)

Justification: For accounts in non-owner mode, a valid cookie not being received is consistent with the user continuing to travel and log in from a borrowed machine. This customization improves usability for the case where the user has a history of password failures when traveling. The refinement on cookie validity addresses variability in password failure rates

based on whether the user is on an untrusted or trusted machine (the latter being identified by requests accompanied by a valid cookie). A significant difference is expected, in part, because of the use of browser features that store and suggest the username and password associated with a particular web page; these tend to reduce password failures rates from trusted machines, and amplify failure rates on untrusted machines (as the user must recall an even less frequently used password).

C2: set b_1 proportional to the borrowing rate. (This is for owner mode accounts only.)

Justification: For accounts in owner mode, the value of b_1 is relevant when the user begins to travel and the account is to transition from owner to non-owner mode. For accounts with a zero (or very low) borrowing rate, setting b_1 to zero provides maximum security, at the expense of requiring the user to answer an ATT in the (unlikely) case of using a borrowed device. Recall $b_1 = 0$ forces an ATT (Figure 2, line 7.2) challenge upon valid userid–password entry without a valid cookie—the latter being characteristic of both logging in from a borrowed device and an attack whereby the attacker does not have access to cookies stored on a trusted machine. Setting b_1 proportional to the borrowing rate (with a suitable upper bound) is consistent with the reasoning that leniency (i.e., user-friendliness in the form of fewer ATT challenges) should depend on the chances that a legitimate user is making the login request.

C3: on receipt of a valid cookie, set b_2 proportional to the password failure rate. (This is for owner mode accounts only.)

Justification: If an account is historically prone to password failures and a valid cookie is received, then it may be likely that the login attempt came from a valid user. For users prone to login errors, usability is increased by selectively increasing b_2.

C4: increase q (for group members) if the group failed login count rises substantially and, in this case, also decrease b_1 and b_2.

Justification: Such an increase may suggest a multi-account attack, in which case lowering b_1 and b_2 and raising q will make the attacker's task more dificult (at the cost of increased inconvenience to associated users).

6. ADDITIONAL TECHNIQUES AUGMENTING ATT-BASED AUTHENTICATION

Here we describe a relay attack and propose a number of techniques to both address it and to augment the original protocol (Figure 1) without changing its basic functionality. These techniques are intended primarily to improve security and are independent of (complementary to) the changes proposed in Section 3. We present them briefly without additional analysis.

6.1 ATT Relay Attack

A relay attack (see also Section 7) may be executed on online protocols involving an ATT by relaying the ATT challenge to an auxiliary location or "workforce," which generates responses, which are relayed back to the challenger. The original ATT target thus avoids the ATT work.

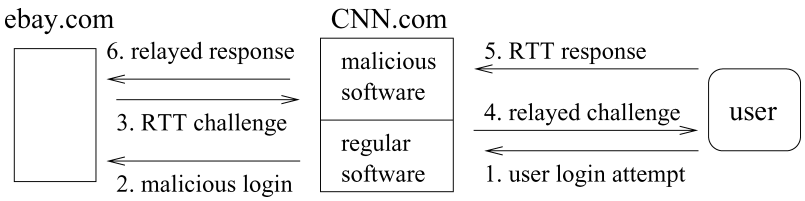

Fig. 4. ATT relay attack.

One attack variant might proceed as follows (see Figure 4). Assume there are two web sites.[5] The first, say ebay.com, is assumed to be the target of regular online dictionary attacks and, consequently, requires correct responses to ATT challenges before allowing access. The second, say CNN.com, is a popular high-volume web site, which for our purposes is assumed to be vulnerable to compromise. The attack begins with an adversary hacking into the CNN.com site and installing attack software.

Upon a user-initiated HTTP connection to CNN.com, the attack software receives the request and initiates a fraudulent login attempt to ebay.com. The attack software, presented with an ATT challenge from ebay.com, redirects it to the CNN.com user connection, instructing that user to answer the ATT to gain access to CNN.com. (Many users will follow such instructions; most users are nontechnical, unsuspecting, and do as requested.) The CNN.com user responds to the ATT challenge. The attack software relays the response to ebay.com, completing the response to the challenge to the fraudulent login attempt. In conjunction with replying to eBay's ATT challenge, after a sufficient number of passwords guesses (e.g., dictionary attack), an eBay account password can be cracked. The procedure is repeated on other accounts, and the attack program summarizes the online dictionary attack results for the adversary.

The attack is easy to perform if the adversary can control *any* high-volume web site, e.g., a popular legitimate site the attacker compromises (as above), or an owned malicious site to which traffic has been drawn, e.g., by illegally hosting popular copyrighted content, a fraudulent lottery, or free software. A related attack involves attack software, which relays ATTs to groups of human workers ("sweatshops"; see also Gentry et al. [2005] on-distributed human computation), exploiting an inexpensive labor pool willingly acting as a mercenary ATT-answering workforce. An unconfirmed real-world variant was reported [von Ahn 2003] to involve an "adult web site," requiring users to solve ATTs before being served the content; presumably those running the site relayed the answers to gain access to legitimate sites, which posed the original ATT in the hope of preventing automated attacks.

6.2 ATT with Embedded Warning

Here we propose a simple method to prevent ATT relay attacks. A drawback of the proposal is that it requires some thought on behalf of users (which, in some

[5]The authors have no affiliation with ebay.com or CNN.com, and no reason to believe either site is insecure. These sites are used as examples simply because of their popularity.

cases, may lead to errors and thus login failures, by legitimate users). However, we believe the general idea may be adapted to significant advantage.

The general idea is to rely upon self-awareness of legitimate users to prevent unwitting participation in an ATT relay attack. One approach is to make ATT challenges user-directed by incorporating a user's specific userid *within the ATT itself*. Preferably, removing this information is of comparable difficulty to answering the ATT itself.

For example, as part of answering a text ATT, a portion of the text is a userid field, which the user is warned to compare to their own userid, to confirm that the ATT is targeted specifically at them (within the embedded warning, the user is instructed to not answer the ATT if the match fails). A variant instead embeds the name of the site being visited (i.e., for which the ATT is being solved for), with similar operation; the choice between web site name and userid could be made dynamically, e.g., selecting the shorter of the two. In another variant, the challenge might also be customized to include the site's graphical logo possibly as a background image, with the user asked to verify this to confirm that the RTT comes from the intended site.[6] Optionally, the ATT might also contain an embedded short "help URL," for a site giving further instructions on the use of this type of ATT.

6.3 Notification Regarding Failed Logins

Here we propose a simple method to detect automated dictionary attacks and trigger counteractive measures (this expands on the idea of administrators manually sending out-of-band messages [Pinkas and Sander 2002]). Once a small threshold (e.g., 3–10) of login failures occurs for any single account, an automated, out-of-band communication (e.g., email) is sent to an address-on-record of the associated legitimate user. If the failed logins resulted from the user's own actions, the user will be aware of the failures and can safely ignore the message; otherwise, it signals malicious activity, and may lead the user to take such actions as to request[7] changes to server-side user-specific login protocol parameters (see Section 3), or to change their own password to a more secure password using the normal change password method.

As an alternative, albeit less desirable,[8] after some larger number of failed logins (e.g., 25), the system might automatically reset the user's password to a computer-generated secure password emailed to the user. This would prevent a user's typically weak self-chosen password from being cracked through standard dictionary attacks. (Depending on the security policy in use, the user might be allowed to change the password back to a weak one if they wish, but, at this point, they may also be motivated to follow recommended password rules.)

---

[6]These latter two variants were suggested by anonymous referees.

[7]For example, this may be done through an authenticated channel such as an email to an unadvertised pre-arranged address, or a hidden URL provided in the email alert to the user.

[8]This may raise customary issues related to system-generated passwords and system-initiated password changes. If used, this alternative must be crafted so as not to generate additional customer service calls, which are not tolerated within our scope. Also, if poorly implemented, such techniques can be abused (by allowing an attacker to force the site to send mail to its users).

This proposal is less effective against multitarget attacks, and *slow-channel dictionary attacks*, wherein an automated program tries passwords on a certain account after there is likely to have already been a successful login attempt (e.g., waiting for a random, but minimal delay, such as one-day intervals). In some systems, an attacker can confirm if a user has logged in recently (e.g., an eBay user), and mount only a limited number of trial password guesses some fixed period after each such successful login. This proposal may nonetheless be helpful and other parameters may limit the success of slow-channel attacks. A small amount of per-user server-side state is needed, but the original protocol has a similar requirement to address cookie theft [Pinkas and Sander 2002]. A remaining drawback of this proposal is degraded usability (additional user attention is required).

6.4 Consuming Client Resources Using Zero-Footprint Software Downloads

We propose that login protocol variants (e.g., see Section 3) be augmented by known techniques requiring that clients solve "puzzles" consuming client resources and return answers prior to the server verifying a login. This follows research lines to combat junk mail (e.g., [Dwork and Naor 1992; Abadi et al. 2003]) and denial-of-service attacks [Juels and Brainard 1999]. Another augmenting technology is to harden passwords with auxilliary protocols that can interact directly with the server [Ford and Kaliski 2000].

Since functionality for performing client puzzles is not resident in standard client software (e.g., browsers), this proposal requires allowing Java applets, JavaScript, or other zero-footprint downloads. Rather than dismissing special client-side software outright [cf. Pinkas and Sander 2002], we see opportunity for advantageous use. Although perhaps worrisome, most users and organizations now operate under the assumption that Java, and certainly JavaScript, are turned on.[9] Nonetheless, since popular web services should work for near 100% of potential users, to accommodate those who cannot use zero-footprint software, ATT-based login protocols can be designed as follows. Client puzzles (or the like) are sent to users. For those unable to answer the puzzles for any reason (in some case the server may learn this *a priori*), the protocol branches to a path replacing the puzzle by an (extra) ATT. This ATT will be less convenient to the user (requiring user attention, versus machine resources), but we expect this to be a relatively small percentage of users, and thus viable.

Another approach to strengthening login protocols involves "strong authentication protocols" like EKE (see Section 7), which, in general, would also require extra client-side software. However, these techniques do not appear to be of use for our problem. EKE-like protocols are designed to preclude off-line (versus online) dictionary attacks, and typically for systems with passwords of very low entropy, which thus rely on account lock-out (as very low entropy passwords can be cracked in a relatively small number of guesses). In contrast, we are interested in environments where account lock-out is not viable.

[9]These are, in fact, the settings that result from the Internet Explorer default ("medium" security), and which we expect remain unchanged by most users.

7. FURTHER BACKGROUND AND RELATED WORK

The ATT relay attack of Section 6.1 is related to general classes of *middle-person attacks* and *interleaving attacks* involving an active attacker inserting itself between legitimate parties in a communications protocol and/or using information from one instance of a protocol to attack a simultaneous instance. Such attacks are well known in cryptographic protocols and have a long history [Diffie and Hellman 1976; Diffie et al. 1992; Menezes et al. 1997].

For example, challenge-response protocols have long been used to identify military aircraft in *identify-friend-or-foe* (IFF) systems. IFF challenges from enemy challengers have reportedly been forwarded in real-time to the enemy's own planes, eliciting correct responses, which were then successfully used as responses to the enemy's original challenges [Anderson 2001]. Note that responses in such systems are typically automatic; the protocols do not involve entity authentication of the querying party.

Related to this is the well-known grandmaster postal-chess attack: an amateur simultaneously plays two grandmasters by post, playing white pieces in one game and black in the other, using his opponents' moves against each other, resulting in an overall outcome better than the two losses he would have achieved on his own.

The term *strong authentication protocols* is often used for protocols designed to preclude attacks, which first obtain appropriate data related to one or more protocol runs and then proceed to crack passwords *offline* (i.e., without further online interaction). This line of research began with the early work of Gong and co-authors [Lomas et al. 1989; Gong 1990; Gong et al. 1993]. The EKE protocol [Bellovin and Merritt 1992] then inspired a number of others (e.g., SPEKE [Jablon 1996]; SRP [Wu 1998]; see also Kaufman et al. [2002]). Offline exhaustive password-guessing attacks typically proceed by trying potential passwords in (perceived) order of decreasing likelihood, with the more probable passwords often in conventional dictionaries, or modified dictionaries specially tailored to this task. Offline attacks are thus often called *dictionary attacks*, although dictionaries are also used in online attacks (if account lock-out and time-delays are not used; see Section 6.1).

Use of system-generated passwords can provide higher security (by better password choices), but suffers severe usability issues. Pass*phrases* have also been proposed (e.g., see [Zimmermann 1995; Yan et al. 2004]). Other approaches include system administrators running password-crack tools on their own systems (*reactive* password checking); enforcement of simple password rules or policies at the time of new password selection, and, at such time, checking for its presence in large customized dictionaries built for this purpose (*proactive* password checking, e.g., see Yan [2001]).

8. CONCLUDING REMARKS

We expect that a large number of human-in-the-loop and mandatory human participation schemes, unrelated to the ATT-based login protocol discussed here, are also subject to the ATT relay attack of Section 6.1.

A major feature of our new protocol is the additional flexibility and configurability, including failed login thresholds and potentially lower ATT challenge probabilities (e.g., for suitable b_2 lowering q does not decrease security). This allows the protocol to be tailored to match particular environments, classes of users, and applications; while determining the optimal parameters for specific user profiles appears nontrivial, we expect further analytical study will be fruitful. Another new aspect is storing cookies only on trustworthy machines. As mentioned in Section 3, the new protocol can be parameterized to give the original protocol as a special case. While the configurability does complicate protocol implementation, we note that a number of the parameters, which are optionally dynamic, can be managed by automated tools, reducing the additional human administrative costs. For example, an automated tool can keep a running ratio of successful logins to failed logins for the entire system and alter a system-wide (or account-specific) parameter q, or system-wide (or account-specific) failed login thresholds b_1 and b_2, based on this ratio. A significant improvement of our protocol over prior work concerns protecting against relay attacks by forcing an ATT challenge on all login attempts after the number of failed logins reaches a threshold. Previous work enabled a significant fraction of the password space to be eliminated with an automated attack. Per-user failed-login counts (as used in Figure 2) also provide protection against sweatshop attacks and ATT relay attacks, especially such attacks targeting a particular account. Note that embedding warnings within ATTs (Section 6.2) does not by itself protect against sweatshop attacks.

For practical protection in Internet-scale live systems, we recommend combining techniques from Section 6 with those of Section 3. We see a large number of ways to expand on the ideas of Section 3. In particular, we encourage others to explore the use of dynamic parameters (ideally managed by automated tools) and other ways to gain advantage by treating users logging in from nonowned devices (e.g., traveling users) different from those continually using their regular login machines.

ACKNOWLEDGMENTS

We thank the anonymous referees whose comments helped improve this paper.

REFERENCES

ABADI, M., BURROWS, M., MANASSE, M., AND WOBBER, T. 2003. Moderately hard, memory-bound functions. In *Proceedings of the 2003 Network and Distributed System Security Symposium*. The Internet Society, Reston, VA. 25–39.

ANDERSON, R. 2001. *Security Engineering: A guide to building dependable distributed systems*. Wiley, New York.

BELLOVIN, S. AND MERRITT, M. 1992. Encrypted key exchange: password-based protocols secure against dictionary attack. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*. IEEE Computer Society, Los Alamitos, CA. 72–84.

BYERS, S., RUBIN, A., AND KORMANN, D. 2004. Defending against an internet-based attack on the physical world. *ACM Transactions on Internet Technology 4*, 3 (Aug.) 239–254.

DIFFIE, W. AND HELLMAN, M. 1976. New directions in cryptography. *IEEE Transactions on Information Theory 22*, 644–654.

DIFFIE, W., VAN OORSCHOT, P., AND WIENER, M. 1992. Authentication and authenticated key exchange. *Designs, Codes and Cryptography 2*, 107–125.

DWORK, C. AND NAOR, M. 1992. Pricing via processing or combatting junk mail. In *Advances in Cryptology—CRYPTO'92*, E. Brickell, Ed. Lecture Notes in Computer Science, vol. 740. Springer-Verlag, New York. 137–147.

FIPS PUB 112, 1995. Password Usage. Federal Information Processing Standards Publication 112, U.S. Department of Commerce, NIST.

FIPS PUB 181, 1993. Automated Password Generator. Federal Information Processing Standards Publication 181, U.S. Department of Commerce, NIST.

FORD, W. AND KALISKI, B. 2000. Server-assisted generation of a strong secret from a password. In *Proceedings of the 9th IEEE International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2000)*. IEEE Computer Society, Los Alamitos, CA. 176–180.

FU, K., SIT, E., SMITH, K., AND FEAMSTER, N. 2001. Do's and don'ts of client authentication on the web. In *Proceedings of the 10th USENIX Security Symposium*. USENIX Association, Berkeley, CA. 251–269.

GENTRY, C., RAMZAN, Z., AND STUBBLEBINE, S. 2005. Secure distributed human computation. In *Proceedings of 6th ACM Conference on Electronic Commerce*. ACM, New York. 155–164.

GONG, L. 1990. Verifiable-text attacks in cryptographic protocols. In *Proceedings of INFOCOM'90*. IEEE Computer Society, Los Alamitos, CA. 686–693.

GONG, L., LOMAS, T., NEEDHAM, R., AND SALTZER, J. 1993. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications 11*, 648–656.

JABLON, D. 1996. Strong password-only authenticated key exchange. *ACM Computer Communcation Review 26*, 5 (Oct.), 5–26.

JUELS, A. AND BRAINARD, J. 1999. Client puzzles: a cryptographic defense against connection depletion attacks. In *Proceedings of the 1999 Network and Distributed System Security Symposium*, S. Kent, Ed. The Internet Society, Reston, VA. 151–165.

KAUFMAN, C., PERLMAN, R., AND SPECINER, M. 2002. *Network Security: Private Communication in a Public World*, 2nd ed. Prentice Hall PTR, Englewood Cliffs, New Jersey.

LOMAS, T., GONG, L., SALTZER, J., AND NEEDHAM, R. 1989. Reducing risks from poorly chosen keys. *ACM Operating Systems Review 23,* 5, 14–18.

MENEZES, A., VAN OORSCHOT, P., AND VANSTONE, S. 1997. *Handbook of applied cryptography*. CRC Press, Boca Raton, FL.

NAOR, M. 1997. Verification of a human in the loop or identification via the Turing test. Unpublished manuscript.

PINKAS, B. AND SANDER, T. 2002. Securing passwords against dictionary attacks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, V. Atluri, Ed. ACM, New York. 161–170.

STUBBLEBINE, S. AND VAN OORSCHOT, P. 2004. Addressing online dictionary attacks with login histories and humans-in-the-loop (extended abstract). In *Proceedings of Financial Cryptography, 8th International Conference*. Lecture Notes in Computer Science, vol. 3110. Springer-Verlag, New York. 39–53.

TURING, A. 1950. Computing machinery and intelligence. *Mind 59*, 236, 433–460.

VON AHN, L. 2003. Eurocrypt'03 presentation of VON AHN ET AL. [2003].

VON AHN, L., BLUM, M., HOPPER, N., AND LANGFORD, J. 2003. CAPTCHA: Using hard AI problems for security. In *Advances in Cryptology—Eurocrypt 2003*, E. Biham, Ed. Lecture Notes in Computer Science, vol. 2656. Springer-Verlag, New York. 294–311.

VON AHN, L., BLUM, M., AND LANGFORD, J. 2004. Telling humans and computers apart automatically. *Communications of the ACM 47,* 12 (February), 57–60.

WOLVERTON, T. 2002. Hackers find new way to bilk eBay users. CNET news.com. March 25 2002.

WU, T. 1998. The secure remote password protocol. In *Proceedings of the 1998 Network and Distributed System Security Symposium*. The Internet Society, Reston, VA. 97–111.

YAN, J. 2001. A note on proactive password checking. In *Proceedings of the 2001 New Security Paradigms Workshop*. ACM, New York. 127–135.

YAN, J., BLACKWELL, A., ANDERSON, R., AND GRANT, A. 2004. Password memorability and security: empirical results. *IEEE Security and Privacy Magazine 2*, 5, 25–31.

ZIMMERMANN, P. 1995. *The Official PGP User's Guide*. MIT Press, Cambridge, MA.

Received July 2004; revised October 2005; accepted March 2006