# Explicit Authentication Response Considered Harmful*

Lianying Zhao and Mohammad Mannan
Concordia Institute for Information Systems Engineering
Concordia University, Montreal, Canada
{z_lianyi, mmannan}@encs.concordia.ca

## ABSTRACT

Automated online password guessing attacks are facilitated by the fact that most user authentication techniques provide a yes/no answer as the result of an authentication attempt. These attacks are somewhat restricted by Automated Turing Tests (ATTs, e.g., captcha challenges) that attempt to mandate human assistance. ATTs are not very difficult for legitimate users, but always pose an inconvenience. Several current ATT implementations are also found to be vulnerable to improved image processing algorithms. ATTs can be made more complex for automated software, but that is limited by the trade-off between user-friendliness and effectiveness of ATTs. As attackers gain control of large-scale botnets, relay the challenge to legitimate users at compromised websites, or even have ready access to cheap, sweat-shop human solvers for defeating ATTs, online guessing attacks are becoming a greater security risk. Using deception techniques (as in honeypots), we propose the user-verifiable authentication scheme (Uvauth) that tolerates, instead of detecting or counteracting, guessing attacks. Uvauth provides access to all authentication attempts; the correct password enables access to a legitimate session with valid user data, and all incorrect passwords lead to fake sessions. Legitimate users are expected to learn the authentication outcome implicitly from the presented user data, and are relieved from answering ATTs; the authentication result never leaves the server and thus remains (directly) inaccessible to attackers. In addition, we suggest using adapted distorted images and pre-registered images/text as a complement to convey an authentication response, especially for accounts that do not host much personal data.

## Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication, Unauthorized access (e.g., hacking, phreaking)*

## General Terms

Security, Human Factors

## Keywords

Authentication, Online password guessing, Deception

*Post-proceedings version: October 26, 2013


*NSPW'13,* September 9–12, 2013, Banff, AB, Canada.

http://dx.doi.org/10.1145/2535813.2535822.

## 1. INTRODUCTION

Automated online password guessing is a long-standing problem for password-based authentication. Nowadays, this problem is possibly getting worse for reasons including the following. (a) The growth of underground market for stolen credentials; i.e., attackers can turn stolen passwords into tangible profits; see e.g., Holz et al. [21]. (b) The value of user accounts increases over time, e.g., long-standing Facebook profiles, Gmail accounts, highly-reputed Paypal accounts. In many cases, user accounts are not as readily replaceable as in the past—i.e., create a new account if the old one is compromised. (c) User chosen passwords are not getting better in terms of complexity. New services requiring passwords are emerging, causing password fatigue or sharing across sites. Also, the increasing number of online participants (e.g., see [22]) makes the use of common passwords more possible. (d) Attackers are getting more organized than before, and have access to better tools and crackers; for example, they now maintain more robust botnets, and can use better techniques than just brute-forcing, e.g., optimized dictionary attacks [29].

Common countermeasures include: rate-limiting the number of allowed login attempts in a given period of time; the use of captchas to restrict automated attacks, see e.g., Pinkas and Sander [38]; and triggering a two-step authentication, e.g., one-time PIN sent to a pre-registered mobile phone, and personal challenge questions. In most cases, attackers can bypass the countermeasures, at least to a limited extent. For example, assuming a three-strike account locking technique is used, an attacker can still employ a large botnet (e.g., million-node) to test the most common passwords and possibly compromise some accounts; here, the attacker is successful if her goal is to access a few accounts (e.g., to use as intermediate money-transfer accounts), instead of compromising a targeted account. Captchas are mostly detested by human users as they are becoming increasingly difficult to decipher (see e.g., [8]); as a side-effect, login times also increase as legitimate users sometimes need to try more than one captcha for an exact match. Several real-world captcha schemes have been defeated by improved image recognition algorithms (see e.g., [9]). As a

result, service providers often leave with no option but to deploy more complex captchas. These limitations are known and several proposals in the past attempted to address the security-usability trade-off in captcha schemes (e.g., [38, 1]).

The fundamental problem here, as we see is that the attacker can learn the outcome of her guess with 100% certainty, using fully automated attacks or involving some trivial human help. Human-assisted captcha breaking services are available, for cheap (see e.g., [27]). As we are aware, verifiers in all known authentication schemes, output a success or failure message after a trial, and we argue that such explicit messages aid online guessing attacks. *Explicit messages* may include return codes from an authentication API, protocol data from the verifier, text string, or even the continuation/discontinuation of the attempted session.

We introduce here *Uvauth* (user-verifiable authentication) to reduce the attacker's confidence on the outcome of her guessed password by granting her access for any password she enters. For a given userid, the correct password will lead to the real user account, and all other passwords will provide *fake sessions* (i.e., with fake user data). To avoid detection by re-logging into the same account, same userid-password pair will always result in the same session. Likewise, different userid-password pairs should also lead to different sessions in order that the legitimate session cannot be distinguished from fake ones. The underlying assumption is that real users will implicitly understand the outcome of their authentication attempt by the presented data; i.e., an unfamiliar account will indicate that the entered password is incorrect, and they need to try again. On the other hand, a random attacker may have little or no idea what to expect as user data after being logged in, even if she launches a human-assisted attack. Attackers can perform different operations to discover a fake session, and our goal is to raise the bar for such attacks to succeed—e.g., by requiring non-trivial efforts from the attacker beyond simply solving a captcha. By increasing the attack cost, we choose to tolerate the attacks, instead of addressing them head-on. Users are also freed from "solving" captchas, or answering personal secret questions as part of their authentication.

Note that Uvauth is different than implicit authentication (see e.g., [44]), where a user is authenticated by her usual traits/actions. An explicit outcome is provided at the end of such an authentication attempt, which we would like to avoid. Our proposal is also independent of whatever secrets, features or tokens used to verify a user; it is the outcome of an authentication attempt that we would like to protect, where online guessing is a concern.

Uvauth's fake sessions can be seen as a form of deception, which has been in use for centuries in traditional wars and conflicts; see e.g., "All warfare is based on deception" [18]. Deception as a cognitive defensive technology has been extensively studied by many researchers for years; see e.g., [43, 50, 12, 3]. In current computer security techniques, this methodology is well demonstrated in honeypots, where deception is used to influence the behavior of attackers, or to collect data for future use, e.g., to understand the attackers and their target systems and network resources (see e.g., [45, 10, 39, 31]).

Our use of deception is not to gain more insights into the attackers' behaviors, but simply to raise the difficulty of online guessing attacks against weak authentication secrets. The following analogy may further clarify the difference. Consider a virtual building with several locked rooms. Honeypots protect access to a room by generating a fake room on-the-fly or claiming that the room is unavailable. In contrast, we create a fake room to protect the lock of a room, assuming the lock is weak—i.e., given enough time, a lock-picker can easily open it. Our use of fake sessions can also be viewed as the no-information leakage property of a perfect one-time pad (OTP) encryption: attackers have no way of verifying a guessed key for an OTP scheme, as a valid key exists for every candidate plaintext, i.e., attackers do not know when they succeed.

Several challenges must be addressed for Uvauth. Generating fake sessions would require additional resources from the verifier, and non-trivial efforts to mimic a legitimate session. Protected accounts should have enough personal content so that legitimate users will easily learn whether they have logged in with the correct password. To address less-/non-personal accounts, we propose the use of distorted images / modified captchas as a communication channel from the verifier to a client. The crucial difference with existing captcha here is that: we do not require users to solve captchas verbatim (i.e., character-by-character) and type the result. Instead, users are expected to use the captcha messages as a second channel to verify their login (i.e., in addition to the content they can see). More challenging captcha schemes can be used in our setting, as users are not required to decipher each character in the exact form.

In summary, our contributions include:

1. In user-level authentication, we introduce the idea of programmably leaving the result of authentication on the server (verifier). Such hiding of authentication results may enable effective protection against online guessing attacks.

2. We propose the use of adapted distorted image as a computer-cipher/human-decipher channel to communicate short messages in human-machine interaction.

3. Our proposal requires no changes on the client side software or existing password input UI, and can be used with any authentication scheme vulnerable to online guessing attacks.

## 2. THREAT MODEL AND ASSUMPTIONS

In this section, we describe our goals, the conditions under which Uvauth works, and list situations that are considered out-of-scope.

**Goals.** The objective of our proposal is to make both machine-only and human-assisted attacks significantly more difficult than using the current state-of-the-art captchas. The level of difficulty can be set by the depth of deception in Uvauth's fake sessions.

**Assumptions.**

**a) User-level authentication.** We address authentication scenarios where a human user is the claimant and a computer is the verifier. We do not include machine-to-machine authentication, e.g., automated script for connecting to a database server.

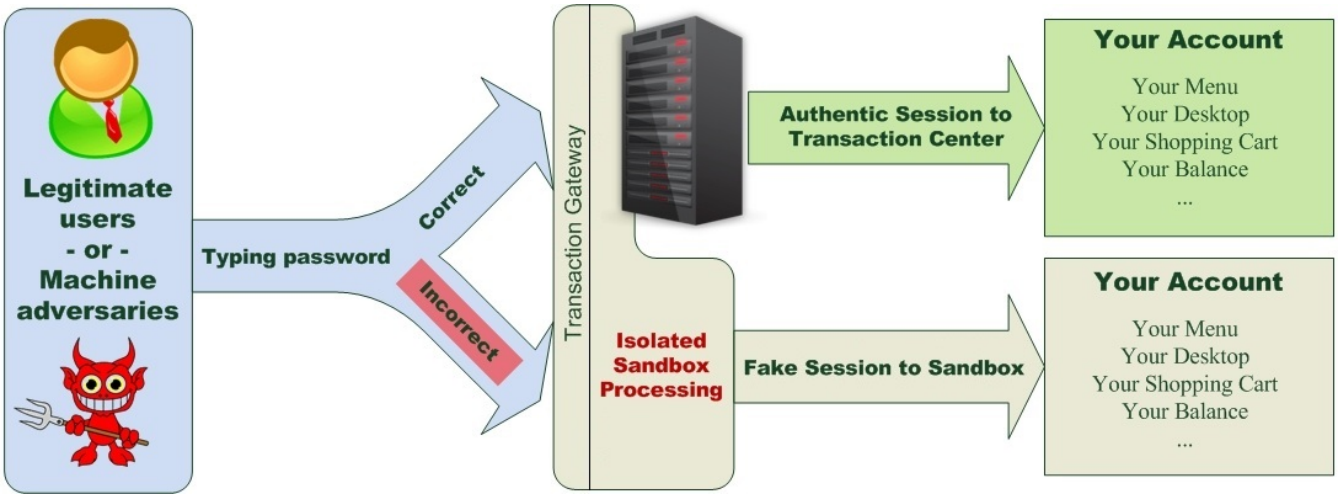


**Figure 1: Overview of user-verifiable authentication**

**b) Weak-secret-based, single-factor authentication.** Uvauth can be used independent of any existing authentication technique, e.g., text or graphical password schemes, certificate-based schemes. However, Uvauth's protection is intended for situations where weak-secrets are used that can be efficiently guessed through online attacks (e.g., a human-chosen password vs. a random 128-bit key). Multi-factor schemes that use an additional token or biometrics also may not need protection against guessing attacks, assuming the additional factors provide enough entropy. However, single-factor multi-stage schemes (e.g., SiteKey or personal questions with passwords) may benefit from Uvauth; e.g., the fake session can start right at the end of first-stage of authentication. However, most of our discussion here considers only commonly-used single-stage password authentication.

**c) Data-oriented sessions.** We focus on accounts that mostly deal with user data (e.g., banking, email), instead of providing some generic services to the user (e.g., Internet access). Implementing fake sessions for service-oriented accounts could be quite challenging, if not impossible.

**d) Separate machines.** The user/attacker software has no means of accessing the verifier's running environment other than via the network channel used for authentication. Otherwise, authentication results may leak from the verifier through side-channels (e.g., [7, 42]).

**e) Random attacker.** Attackers in our model are assumed to be random individuals, i.e., unrelated to a target user. If the user is known to the attacker, fake sessions in Uvauth may be detected by known information (e.g., Facebook profile information, email contacts). However, the attacker may know all valid userids of a target service.

**f) No offline attacks.** We assume that data at rest is safe, e.g., password databases are inaccessible to attackers. Otherwise, simpler offline attacks can be mounted to reveal the passwords (if hashed or encrypted under a weak key).

**g) Other password-unrelated security issues.** Our proposal only addresses online password guessing; so, if a website or application is vulnerable to other types of attacks such as SQL injections, Uvauth's protection may not help. We also do not address several other threats, including: phishing, malicious software on the client or verifier, and session hijacking attacks.

# 3. UVAUTH: USER-VERIFIABLE AUTHENTICATION

In this section, we discuss Uvauth and the underlying self-evidence of authentication that may make the scheme feasible. By analyzing some account properties, we also provide a list of considerations for designing fake sessions, and discuss scenarios where Uvauth may be more applicable.

**Overview.** Figure 1 shows an overall architecture of Uvauth. Legitimate users and potential attackers are treated equally, in terms of authentication results. A transaction gateway accepts all incoming authentication requests; the gateway is also configured to authenticate users (e.g., it has access to user credentials). When a correct userid-password pair is received, processing is handed over to the transaction center and a legitimate session is established. Otherwise, when the given password is incorrect, the user/attacker is redirected to a sandbox-enabled environment that hosts fake user sessions. The established sessions in both cases appear to be (almost) the same to a machine. A random human attacker may also be unable to judge the content of the fake account without performing some non-trivial tasks.

## 3.1 Implicit detection of an authentication outcome

We first consider authentication sessions where users can distinguish success/failure without explicit messages from the verifier. This is the basic type of authentication considered in Uvauth, and requires user-knowledge of the target account. In Section 4, we discuss less-personal accounts where some explicit hints from the verifier are needed.

### *3.1.1 Self-verification*

If the data fed to end users after a login request is personal and of relatively high-entropy, the presented data itself may be enough for a straightforward and effortless decision by the real data owner. In this case, the authentication result is implicit, i.e., requires no indication of failure or success. Consider the following as examples of this type of authen-

tication. For most active users of a social networking site (e.g., Facebook), users can (possibly) easily identify their own accounts after a successful login—e.g., from the profile info, page layout, friends list and messages. The same is possibly also true for online banking login, identified by e.g., user info, account balance, transaction history and registered bills. These types of accounts are highly personal and quite unique to a user. More importantly, these accounts can be populated with fake information to make them indistinguishable even to *non-owner* human users (in addition to automated bots).

User-verifiability obviously requires that the same user experience is provided for a specific credential used. Therefore, to implement a user-verifiable authentication scheme that is both user-acceptable and attacker-indistinguishable, we must consider the following issues. First, each fake session generated for a specific userid-password pair (even if the userid is non-existent), must appear to be the *same* for a certain period of time. If randomness of fake sessions is distinguishable for login attempts with the same userid with different passwords, attackers can easily detect the difference, and then learn the authentication outcome. On the other hand, the fake session for a specific userid must change with time, as is the case for many user accounts (e.g., new messages and friends in a Facebook account; updated balance and new transactions in a banking account).

### 3.1.2 Additional login help for legitimate users

To aid users and help identify a successful login, a combination of the following methods can also be used.

**a) Customized messages.** A user customized welcome message may be used for the identification of a valid session. During account registration, a user can set up some personalized information so that when a correct password is entered, it will be displayed; otherwise, a random message is displayed. Such customized messages may be an image, or excerpts from a book. Note that, our use of customized message/image is different than existing anti-phishing solutions such as SiteKey [4], and Verified-by-Visa personal message [47]. We do not address phishing, and security of Uvauth is not dependent on users' noticing the messages correctly or all the time. If the user does not pay heed to the displayed image/message, they may be mislead into believing a successful login, which eventually will be detected when they check carefully their account information. In contrast to known vulnerabilities in SiteKey (e.g., [49]), no authentication secrets are leaked for the user's mistake in Uvauth.

**b) Secondary channels.** An out-of-band signalling, e.g., SMS/twitter/email messages can also be used to notify when a login is successful. Mobile SMS is widely used for user status indication in many businesses, such as successful credit card transactions (see e.g., MasterCard inControl [26]). We assume here that the secondary channels are not compromised; otherwise, an attacker can use such a channel for verification. Periodically, users may also be notified about failed login attempts through secondary channels (e.g., in the form of a daily digest).

**c) Warning messages.** A warning message may be displayed so that the user is reminded that Uvauth is in place, and verify whether they can access their data. An example message is as follows: "Please check your account data; in case you do not see your expected data, try again with the correct password."

**d) Dynamic security skins.** Anti-phishing techniques such as synchronized random dynamic boundary [48] and dynamic security skins [15] can be used as a means to identify an authentic server, and to communicate success/failure messages to a client browser. Note that, Uvauth's security does not require these visual cues to be 100% reliable, or always correctly matched by users; they simply provide an additional channel for session verification.

**e) Limiting fake sessions for *known* devices.** Authentication attempts from known devices with prior successful logins for a specific userid can be exempted from fake sessions when an incorrect password is entered, and given directly a traditional failure message (e.g., incorrect userid or password). User devices may be whitelisted by IP addresses, cookies, geolocation services as enabled in popular browsers including Google Chrome[1] and Mozilla Firefox,[2] or through other web-based device fingerprinting mechanisms (see e.g., [34]). Assuming that most legitimate users access their accounts from a relatively fixed set of devices (computers at home or office, or mobile devices), such exemptions from fake sessions may aid usability; similar mechanisms have been explored in prior work (see e.g., [38, 1]; more in Section 6). However, to counter guessing attacks from infected whitelisted devices and cookie theft, such exemptions must be limited (e.g., by the number of allowed attempts without fake sessions).

## 3.2 Designing fake sessions

Uvauth's effectiveness depends on attackers being unable to detect fake sessions *efficiently*. Below, we discuss few considerations and account properties for designing effective fake sessions.

### 3.2.1 Account properties

Here we list four factors that may be used to categorize account types. We also discuss how these factors may be considered during the generation of fake sessions.

**a) Server-side data retention.** Here we consider whether the user is allowed to make changes after logged in and to what extent the changes are kept and accessible when she logs back in at a later time. This feature of a user account could be resource-intensive, as fake sessions may also need to store attacker-initiated changes. If no changes are stored, inconsistent fake sessions may still be useful to some extent; cf. Neagoe and Bishop [31]. For read-only accounts (e.g., call logs of a pre-paid phone card), generating fake sessions could be much easier. However, most online accounts generally allow at least some changes (e.g., profile parameters). If the size of updateable data is small, the cost of consistent fake session generation may still remain affordable.

**b) Client-side data representation.** For most account types, users get access to some data after logged in. How much an attacker can understand the meaning of user data, determines how easily she can detect a fake session. For highly-personalized data (e.g., photos, blogs, and calendars), fake session detection would be significantly difficult for an attacker, even if human assistance is used; the attacker has no obvious means to distinguish between fake and real data. For impersonal, human-readable data (e.g., magazine subscriptions), fake sessions should be populated with context-

[1] https://support.google.com/chrome/answer/142065
[2] http://www.mozilla.org/en-US/firefox/geolocation/

aware, meaningful data. For impersonal data with machine semantics (e.g., protocol traffic or command responses), it may be more difficult to generate fake sessions, and sometimes specific restrictions should be applied to limit the cost of fake sessions (e.g., running processor-intensive jobs in a fake ssh session).

**c) Update types.** Some accounts are update-driven, i.e., frequently updated directly by both the account owner and others for the purpose of communication; examples include email and social networking accounts. Some accounts are activity-driven, i.e., indirectly updated by user transactions; examples include credit card accounts. Some accounts may be of mixed type; e.g., a seller's Paypal account is updated by Paypal (e.g., transaction logs) and other users (e.g., comments). These different account types should be modeled correctly to design realistic fake sessions.

**d) Externally-modifiable data.** If anyone can influence the content of a target account, the account is considered externally-modifiable; examples include email accounts (e.g., anyone can send an email), social networking accounts (e.g., public posts). These accounts are susceptible to the post-and-check attack as discussed in Section 5.

#### 3.2.2 *Considerations for fake session generation*

**a) Verisimilitude.** There is a trade-off between the deployment of more realistic/consistent fake sessions with more functionality and resource consumption on the server. We define the depth of verisimilitude as the levels of operation a fake session would allow, before it may be detected by an automated attacker. Also, not all functions are equal in terms of costs–e.g., allowing the update of a profile parameter vs. searching for a transaction. As an example, consider a fake session at an online banking portal; an automated attempt can be performed by an attacker to transfer a certain amount of money to an account that directly/indirectly belongs to him, as such tasks are not far down in the operations hierarchy. A countermeasure is to output deceptive statements such as "Transfer-out is not activated for this account", "USB token is required for this transaction". See e.g., Rowe [43] for an in-depth discussion on how to design good deceptions for intruders with a probabilistic model of belief and suspicion. Moreover, text strings (e.g., names and messages) used in fake sessions should meet certain criteria; existing work on generating (somewhat meaningful) random words/phrases may be used (see e.g., [13, 2]). Note that, for Uvauth to be effective, detection of fake sessions must be non-trivial, but it is non-essential to deploy highly complex fake sessions to make detection very difficult.

**b) Timing characteristics.** Sometimes due to network delay or processing on the server side, logging in or operations on a website are subject to different levels of responsiveness. Fake sessions should insert lags when required to simulate timing characteristics of different operations in the operations hierarchy, hours of the day, or even seasons in a year. This may also help confuse intruders as they cannot detect fake sessions by profiling timing characteristics. The freed time slots can be used for scheduling more fake sessions.

**c) Data sanitization.** Data sanitization (also known as redaction for printed documents) is to hide or transform confidential information before publishing. Examples include erasing customer names, randomizing figures, or disrupting the order of user behaviors. In some scenarios, it may be necessary to reuse parts of the real production/user data for generating fake sessions, especially accounts with a lot of user data. Up-to-date operating data from a real system may be sanitized by removing all privacy/security-sensitive parts, while retaining interrelated rationality (see e.g., [6, 35]). For instance, in the case of a web portal of a mobile phone subscriber, the prefix of a login phone number may indicate some regional information; therefore, the presented information, such as, the numbers in the call log and the address of residence must also appear legitimate after sanitization. The account balance can be randomized to some extent, but the call/message logs could be pulled, sanitized and mixed from a group of real users (i.e., individual identifiers are removed but group characteristics are preserved). However, special care must be taken to sanitize data to avoid exposure of sensitive data (see e.g., [30, 5]). For Uvauth, a significant amount of fake data can be mixed with user data before applying sanitization, which may reduce the risk of privacy exposure.

**d) Virtualization.** As Uvauth may need to deal with a large number of fake sessions (e.g., when under guessing attack from a botnet), virtualization technologies can be used for creating and hosting those sessions efficiently. We have not tested generating such large-scale VM deployment for evaluating Uvauth; cf. CLAMP [36]. Virtualization may also help limit resources allocated to fake sessions, especially when under heavy-load (e.g., due to DoS attack).

## 4. DISTORTED IMAGE AS A COMMUNICATION CHANNEL

In this section, we discuss the possibility of using captchas as a one-way communication channel (server-to-user), and propose few variations of existing captchas for this purpose. These captcha variants may be considered when techniques in Section 3.1 are not preferred (e.g., for deployability or usability reasons). Less personalized accounts (e.g., movie streaming websites), and managed-systems in batch (e.g., remote administration), may benefit from the proposed methods. We assume that these accounts would be attacked primarily by bots (i.e., no human assistance), as they may be less valuable compared to personal/financial accounts.

### 4.1 Captchas as a cipher

Most current captcha techniques are based on the use of distorted images (or similar methods), and are used before authentication, to verify the presence of a human user. In contrast, we propose a post-authentication use of captchas. The idea is to utilize the generation and recognition of distorted images to communicate the authentication result back to end users. End users will not be *tested* with our schemes below, and no user response is needed; users simply become recipients of the *ciphered* information. Note that, similar use of captchas has been proposed earlier for different purposes, e.g., verification of message integrity in an untrusted terminal [24], and NSA-proof fonts [28].

Using captchas to communicate messages is relatively immune to relay attacks (as compared with regular captchas). A machine adversary can still make use of exploited popular websites, and have a large number of innocent users to solve the distorted images. However, for Uvauth captchas, only recognizing all characters is not enough, and semantic interpretation is required to learn whether the feedback is

positive or negative. We discuss few captcha variants in Section 4.2 that may make regular captchas more difficult for machine attackers.

## 4.2 Adaptation of regular captchas

For regular captchas, the content can be arbitrary and randomized, without carrying any meaningful information, e.g., an irrelevant mix of letters and numbers. However, for Uvauth, we need to transmit messages in natural languages with predefined meanings for conveying authentication results. Existing captcha breaking techniques (e.g., [17]) would perform even better against Uvauth captchas due to the limited entropy of our messages (resulting mostly from the fixed nature of the messages). To address this, the captcha generation may be adapted as follows.

**a) Randomized padding.** Humans have the ability to semantically interpret a message even if the message is garbled to some extent. Most people do not read all the characters in a word, or even all the words in a sentence (see e.g., [40]). As an example, consider the following sentences: "hke It uu is qKd k9l2 fine vMab weather.", "If You Can Raed Tihs, You Msut Be Raelly Smrat"; in most cases, humans can understand the meaning without much difficulty, but for machines it is not straightforward to extract the meaning from these sentences, especially when such messages appear in a distorted image. As an example, see Figure 2.

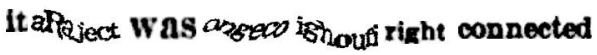

**Figure 2: Distorted image with random padding**

**b) Indirect expression.** Emotional tones in indirect positive or negative expressions such as "Everything goes well!" (correct password entry), or "Your password makes me angry!" (incorrect password entry) are quite self-evident for humans, but not so straightforward for machines. Existing work shows that machines can also learn to identify emotions in text (e.g., [46]), but requires non-trivial resources (e.g., a large knowledge database).

**c) Display anywhere.** Automated attacks on a captcha somewhat depends on the ability to locate the captcha on a screen. In regular usage, captchas are generally placed in a deterministic location, to facilitate the ease of processing by human users. As Uvauth's communication channel is one-way (i.e., no response back from the user), the distorted image can be placed anywhere on the screen (as long as it is visible to the user). It can also be embedded into a larger bitmap (e.g., banners, ads, backgrounds) to make automated identification difficult. Random delays (e.g., few seconds) may also be added before displaying the message (after the authentication phase), to frustrate the attacker even further.

## 4.3 An example with VNC

We now discuss how adapted distorted images may be used with a Virtual Network Computing (VNC [41]) application for remote desktop management.[3] When the remote machine is not personal to the user (e.g., accessed as a sysadmin), login feedback via distorted images may be used. Figure 3 shows a VNC session when an adapted distorted image (with "display anywhere") is used for authentication feedback. Here a legitimate user may expect such a string to be displayed anywhere on the screen. In contrast, for a machine attacker, it may be difficult to identify the distorted message from a screen-capture, specifically, when the message is blended with the background. Additionally, there is no need to display the distorted image right after login; e.g., a short, random delay can be added to confuse the attacker even further. The attacker may need to forward a video clip to a human solver to perform a relay attack, which would increase the cost of such an attack.

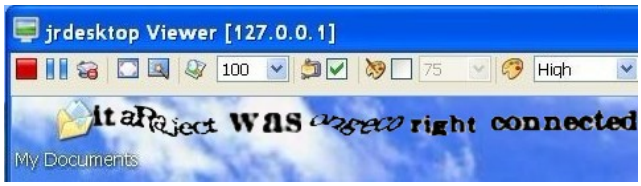


**Figure 3: A VNC session with an adapted distorted image**

# 5. LIMITATIONS AND ATTACKS

In this section, we evaluate Uvauth from an attacker's perspective and list possible attacks. Some of these attacks can be mitigated if special care is taken, while others are limitations of our current design.

**a) Post-and-check attacks.** For certain accounts, attackers can first post a message to the target account, and then check for the posted message when launching a guessing attack on that account. For example, an attacker can post a comment on the target's Facebook account, and by checking whether this specific post is seen, the detection of a fake session becomes easier. Similarly, an email can be sent to a victim's Gmail account for the same purpose, and the attacker then just checks whether the email has been received when in the fake session. We term these attacks as post-and-check attacks, which can be automated and can make designing fake sessions significantly difficult. Application-specific defenses can be designed. For example, for a fake Facebook session, the target user's publicly-visible content, including posts from non-friends should be used. Assuming the attacker is not socially-connected to the user (i.e., not a Facebook friend), post-and-check attacks can be restricted.

Designing a similar mechanism for email is less straightforward, as no explicit social connections exist in email. However, email services (e.g., Gmail) are currently quite effective against spam email accounts; recently-received spam emails for a targeted account can be used in fake sessions (albeit with the risk of some information leakage, as sometimes legitimate emails are labelled as spam). Attackers also must send an email to the target account immediately before launching the guessing attack; otherwise, they would not know whether the target user has deleted the unwanted email, or they are in a fake session. Emails received from first-time contacts in a recent period (e.g., in the last five minutes) may be included in fake sessions. This can restrict post-and-check attacks for email accounts, at the expense of occasional information leakage. Email contacts as displayed in a fake session could also be problematic. If fake email addresses are used, by sending emails to these addresses, an attacker may identify the fake session (e.g., if an immediate

[3] `jrDesktop`, see: `http://jrdesktop.sourceforge.net`

delivery failure message is received). On the other hand, the use of real email addresses would cause obvious privacy exposure (e.g., harvesting of emails).

**b) Targeted attacks.** If an attacker knows a victim in person (real-world or online-only), she may also know one or more contacts in the victim's Facebook friend list, or the account number / address for online banking. When such information can be expected by the attacker, a fake session can be easily detected. We focus on restricting large-scale automated guessing attacks, and exclude targeted attacks (although these attacks may also be significant in some scenarios; see e.g., [25]).

**c) New denial-of-service attacks.** Uvauth fake sessions may be exploited to launch algorithmic/complexity-based DoS attacks (e.g., [14]). An adversary can initiate many fake sessions with resource-intensive operations on the server-side to overload the server, e.g., text search in an email account. So fake sessions must be designed carefully, and the allowed activities therein should not consume too much resources; i.e., the trade-off between verisimilitude and resource consumption must be chosen with care.

**d) Adapted relay attacks.** Paid human solver services (e.g., as discussed in [27]) can be used to attack Uvauth messages that rely on adapted captchas. We can alleviate such risk by applying "Display Anywhere", so that the attacker has to forward the whole screen or even a video clip to the human solvers which incurs more effort.

**e) Inconsistency attacks.** If states in fake sessions are not saved, then the attacker may detect a fake session by making some changes to it, and checking for those changes after a re-login. This is a known problem in deception, and referred as inconsistency of deception. Neagoe and Bishop [31] argue that even inconsistent deception can still effectively confuse an attacker.

**f) Acquired targeted attacks.**[4] Assume that a random attacker wants to guess the password for a specific account $A$ and the attacker has already compromised another account $B$ from the same user. Also assume that the password for $A$ is different than that of $B$. Now, similar to the targeted attacks discussed above, the attacker can use extracted information from $B$ to detect fake sessions for $A$. Note that, the attacker may need only temporary / one-time access to $B$. If the attacker can successfully guess the password for $A$, she can now use information from both accounts to brute-force other accounts from the same user (even when password reuse is avoided). As users generally maintain several password-protected accounts, this attack may be quite realistic—e.g., through the compromise of a large-scale, popular service provider (for some recent incidents, see e.g., [32]).

**g) Legitimate users in a honeypot.**[4] If an attacker succeeds in compromising an account (e.g., through password guessing), she could then (maliciously) change the password, e.g., to keep the account in her control and deny access to the legitimate user. Now, when the user tries to log in with the old password, he will be confused; by not seeing his data, the user might assume that he has mistyped the password, and keep trying several times before realizing the attack. Without Uvauth, the user will be denied access, and possibly try account recovery methods immediately.

[4]An anonymous NSPW2013 reviewer pointed us this attack.

A similar issue arises even when the account password remains uncompromised. If an incorrect password is tried (e.g., due to typos), users must detect the resulting fake session, and then log out for another attempt. Such wrong password entries would cost more time for users due to the additional step of detecting fake sessions. This usability issue is a side-effect of Uvauth, and does not happen with an explicit feedback, as in regular authentication. Note that typos can be avoided by displaying the password in cleartext (cf. [33]), specifically when shoulder-surfing is not an issue (e.g., the user sitting alone in her office). However, misremembered passwords may not be readily detected by such password unmasking, and the user may still be delayed in discovering the situation, partly due to Uvauth's fake sessions.

Other limitations include: we have not evaluated the server-side load for generating and running a large number of fake sessions. We also have not tested how effectively users can detect implicit results from an authentication attempt, or whether messages via adapted distorted images can be used in practice.

## 6. RELATED WORK

Uvauth falls in the intersection of password security and deception techniques. Here we highlight a few related projects from both areas.

Pinkas and Sander [38] first proposed the use of Reverse Turing Tests (RTTs, e.g., captchas) to restrict large-scale online password dictionary attacks. The protocol challenges users with RTTs for a small fraction (e.g., 5%) of all possible userid-password pairs to reduce the server-load (of generating RTTs) and usability impact (of answering RTTs), while keeping the cost of launching a large-scale guessing attack significantly high. Correct passwords always require an RTT, unless a valid cookie from past successful login is found. In Uvauth, deploying fake sessions only for a small fraction of all login attempts, will also significantly reduce server-side load. However, if attackers use a small password dictionary (e.g., top 500 words), the number of fake sessions they must process may be too small to provide any significant protection. Assuming many users use common/weak passwords that may be found in small dictionaries, we recommend the use of fake sessions for all failed login attempts.

Later RTT-based proposals further improved security and usability aspects of the original Pinkas and Sander [38] scheme. For example, the password guessing resistant protocol (PGRP [1]), where more RTTs are imposed on *unknown* (possibly attack) machines than *known* (possibly legitimate) ones; machines are categorized using source IP addresses and cookies. As discussed in Section 3.1.2, item (e), the use of known devices may reduce the number of fake sessions for legitimate users. Unlike RTT-based schemes, Uvauth does not provide explicit authentication feedback, and avoids challenging users with RTTs. Recall that, even for our use of distorted images as a communication channel, we do not require a response from the user.

Goyal et al. [19] extend the *pricing via processing* paradigm (introduced by Dwork and Naor [16]) to address online password guessing; the proposed protocol (*CompChall*) imposes a significant amount of computation for the client on each authentication attempt. CompChall would not adversely affect legitimate users since their authentication attempts are expected to be limited. In contrast, the scheme may nega-

tively impact an attacker when a large number of attempts are made from a single machine. However, CompChall may not be effective against attacks from a botnet.

The idea of closely monitored network decoys (honeypots), to distract/deceive adversaries from real targets and to collect analytical information about an attack, has more than two decades of history (see e.g., [45, 10]). Our methodology resembles honeypots in the sense that the attacker is also given deliberate access, and fed with false information. However, in contrast to honeypots, our use of deception focuses on hiding the result of an authentication attempt, instead of detecting/analyzing malicious activities. Similar to the generation of fake sessions in Uvauth, the deployment of a honeypot is also time-consuming and resource-intensive. Provos designed Honeyd [39], a framework for virtual honeypots that simulates virtual computer systems at the network level. It saves physical resources in terms of resource consolidation and tolerance of high destructiveness. Additionally, it is more flexible to configuration changes, and thus allows more complicated behaviors to be implemented. Uvauth's fake session generation may benefit from such existing honeypot work.

Herley and Florêncio [20] propose the use of honeypot credentials to restrict brute-force guessing attacks on online banking accounts. During account creation, for each userid, a large number of honeypot passwords ($n$, a subset of all possible passwords) are also registered along with the correct password. The userid with honeypot passwords are considered honeypot credentials, and all such credentials will lead to honeypot sessions, which are especially tracked by the bank server for money transfer attempts. To reduce the probability of mistyping by a real user, all honeypot passwords are chosen to be more than two characters apart from the correct password; however, a brute-force attacker is still $n$ times more likely to try a honeypot password. Honeypot sessions are created from real user data (e.g., attributes, transactions) with fake identification information such as names and addresses. In comparison, Uvauth's scope is broader, and it considers the use of small password dictionary with known userids (instead of trying all possible entries from the userid-password space).

Pavlovic [37] re-visits the idea of security by obscurity, assuming attackers, like defenders, also have limited logical or programming resources. It is argued that the behaviour of defenders can also be hidden to gain tangible security advantages. Uvauth's use of deception is limited to hiding only the defenders' verification outcome from attackers.

Most work on deception focuses on maintaining consistency of the false reality as presented to attackers. Neagoe and Bishop [31] explore inconsistent deception for defending computer resources, and argue that these techniques may still be effectively used to track and monitor attackers. Forgoing consistency may also make the design of deception techniques simpler and less resource-intensive. Such techniques may significantly reduce the cost of deploying fake sessions in Uvauth.

Clark and Hengartner introduced panic password [11], where a separate password is used to indicate a duress situation to the server without soliciting an authentication failure; the primary goal is to protect both the victim's safety and sensitive information residing on the server. On the entry of a panic password, the observable response is to deceive the adversary with panic responses that are indistinguishable from the real response. While panic passwords are proposed to be used by a legitimate user under duress, Uvauth is targeted towards protecting passwords from being guessed using a botnet, or by (random) human-assisted attackers.

Juels and Rivest recently proposed *honeywords* [23] (false passwords) to address offline attacks against hashed password databases. For each account, the legitimate password is mixed with several honeywords; thus, when an attacker cracks a hashed password, she cannot be sure if it is the real password or a honeyword. Also, the use of a honeyword will trigger an alarm on the server-side (cf. panic password).

# 7. CONCLUSION

We propose Uvauth to hide authentication results from attackers to mitigate the risk of online password guessing. It can effectively deceive an attacker assuming fake sessions can be efficiently generated (as an attacker may launch many authentication attempts from a large-scale botnet). Most current authentication schemes would fail to an adversary who is willing to use human help to break into existing techniques that are designed to limit only automated attacks. As user accounts generally become more and more valuable with the duration of use, it may be worthwhile for attackers to invest in cheap human labor as a means to compromise user credentials. In designing Uvauth, we explicitly consider such threats and provide limited protection (possibly significantly more than existing technologies). Implementing Uvauth fake sessions would require server-side support, but no changes are needed on the client-side software or existing password input UI (including browser mechanisms such as "keep me logged in" and cookies). However, Uvauth, as presented, has not been fully evaluated, and has a number of limitations. Our goal is to attract attention to an important drawback of existing authentication schemes that enables large-scale guessing attacks.

## Acknowledgements

We thank our shepherd Michael Franz, anonymous NSPW2013 reviewers, NSPW2013 attendees, Jeremy Clark, Julie Thorpe, and members of the Concordia's Computer Security Lab for their insightful suggestions and advice. The second author is supported in part by an NSERC Discovery Grant and Concordia University Start-up Program.

# 8. REFERENCES

[1] M. Alsaleh, M. Mannan, and P. van Oorschot. Revisiting defenses against large-scale online password guessing attacks. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 9(1):128–141, 2012.

[2] J. Aycock. @transformitt. *Leonardo*, 46(5):482–483, Oct. 2013.

[3] G. Bakos and S. Bratus. Ubiquitous redirection as access control response. In *Annual Conference on Privacy, Security and Trust (PST'05)*, St. Andrews, NB, Canada, October 2005.

[4] Bank of America. SiteKey authentication: An additional layer of online and mobile banking security. https://www.bankofamerica.com/privacy/online-mobile-banking-privacy/sitekey.go.

[5] M. Bishop, R. Crawford, B. Bhumiratana, L. Clark, and K. Levitt. Some problems in sanitizing network

data. In *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2006)*, Manchester, UK, June 2006.

[6] M. Bishop, J. Cummins, S. Peisert, A. Singh, B. Bhumiratana, D. Agarwal, D. Frincke, and M. Hogarth. Relationships and data sanitization: A study in Scarlet. In *New Security Paradigms Workshop (NSPW'10)*, Concord, MA, USA, Sept. 2010.

[7] D. Brumley and D. Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, Aug. 2005.

[8] E. Bursztein, S. Bethard, J. C. Mitchell, D. Jurafsky, and C. Fabry. How good are humans at solving CAPTCHAs? A large scale evaluation. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2010.

[9] E. Bursztein, M. Martin, and J. C. Mitchell. Text-based CAPTCHA strengths and weaknesses. In *ACM Computer and Communications Security (CCS'11)*, Chicago, IL, USA, Oct. 2011.

[10] B. Cheswick. An evening with Berferd, in which a cracker is lured, endured, and studied. In *Winter USENIX Conference*, San Francisco, CA, USA, Jan. 1992.

[11] J. Clark and U. Hengartner. Panic passwords: Authenticating under duress. In *USENIX Workshop on Hot Topics in Security (HotSec'08)*, San Jose, CA, USA, July 2008.

[12] F. Cohen. The use of deception techniques: Honeypots and decoys. *The Handbook of Information Security*, 3:646–655, 2006.

[13] H. Crawford and J. Aycock. Kwyjibo: automatic domain name generation. *Software: Practice and Experience*, 38(14):1561–1567, 2008.

[14] S. A. Crosby and D. S. Wallach. Denial of service via algorithmic complexity attacks. In *USENIX Security Symposium*, Washington, DC, USA, Aug. 2003.

[15] R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. In *Symposium on Usable Privacy and Security (SOUPS'05)*, Pittsburgh, PA, USA, July 2005.

[16] C. Dwork and M. Naor. Pricing via processing or combating junk mail. In *Advances in Cryptology - CRYPTO'92*, Santa Barbara, CA, USA, August 1992.

[17] Futurity.org. Gotcha! captcha security flaws revealed. Available at: http://www.futurity.org/science-technology/gotcha-captcha-security-flaws-revealed/.

[18] L. Giles. *Sun Tzu on the Art of War: The Oldest Military Treatise in the World*. London Luzac, 1910. Chapter 1: verse 18.

[19] V. Goyal, V. Kumar, M. Singh, A. Abraham, and S. Sanyal. Compchall: Addressing password guessing attacks. In *International Symposium on Information Technology: Coding and Computing (ITCC'05)*, Las Vegas, NV, USA, April 2005.

[20] C. Herley and D. Florencio. Protecting financial institutions from brute-force attacks. In *Proceedings of The Ifip Tc 11 23rd International Information Security Conference*, volume 278, pages 681–685. Springer US, 2008.

[21] T. Holz, M. Engelberth, and F. Freiling. Learning more about the underground economy: A case-study of keyloggers and dropzones. In *European Symposium on Research in Computer Security (ESORICS'09)*, Saint Malo, France, Sept. 2009.

[22] Imod Digital. Social network numbers, May 2012. http://www.imoddigital.com/iD-Social-Network-Statistics-2012-eBook.pdf.

[23] A. Juels and R. L. Rivest. Honeywords: Making password-cracking detectable. Technical report (May 2, 2013). http://people.csail.mit.edu/rivest/pubs/JR13.pdf.

[24] J. King and A. dos Santos. A user-friendly approach to human authentication of messages. In *Financial Cryptography and Data Security (FC'05)*, Roseau, Dominica, February 2005.

[25] Knowthenet.org.uk. More teenagers are being hacked by friends online but did you know it could be illegal? News article (Jan. 6, 2012). http://www.knowthenet.org.uk/.

[26] MasterCard. MasterCard inControl service now available from Barclaycard. News release (Jan. 21, 2010). http://www.mastercard.com/us/company/en/newsroom/pr_mc_incontrol_service.html.

[27] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage. Re: CAPTCHAs – understanding CAPTCHA-solving services in an economic context. In *USENIX Security Symposium*, Washington, DC, USA, August 2010.

[28] S. Mun. Making democracy legible: A defiant typeface. Blog post (June 20, 2013). http://blogs.walkerart.org/design/2013/06/20/sang-mun-defiant-typeface-nsa-privacy/. Project website: http://z-x-x.org.

[29] A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *ACM Computer and Communications Security (CCS'05)*, Alexandria, VA, USA, November 2005.

[30] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2008.

[31] V. Neagoe and M. Bishop. Inconsistency in deception for defense. In *New Security Paradigms Workshop (NSPW'06)*, Dagstuhl, Germany, Sept. 2006.

[32] NewYorker.com. The inevitable downfall of your password. News article (July 17, 2013). http://www.newyorker.com/online/blogs/elements/2013/07/tumblr-vulnerability-how-to-secure-your-passwords.html.

[33] J. Nielsen. Stop password masking. Online article (June 23, 2009). http://www.nngroup.com/articles/stop-password-masking/.

[34] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, May 2013.

[35] S. R. M. Oliveira and O. R. Zaiane. Protecting sensitive knowledge by data sanitization. In *IEEE International Conference on Data Mining (ICDM 2003)*, Melbourne, FL, USA, November 2003.

[36] B. Parno, J. M. McCune, D. Wendlandt, D. G. Andersen, and A. Perrig. Clamp: Practical prevention of large-scale data leaks. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2009.

[37] D. Pavlovic. Gaming security by obscurity. In *New Security Paradigms Workshop (NSPW'11)*, Marin County, CA, USA, Sept. 2011.

[38] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *ACM Computer and Communications Security (CCS'02)*, Washington, DC, USA, November 2002.

[39] N. Provos. A virtual honeypot framework. In *USENIX Security Symposium*, San Diego, CA, USA, Aug. 2004.

[40] K. Rayner, S. J. White, R. L. Johnson, and S. P. Liversedge. Raeding wrods with jubmled lettres: There is a cost. *Psychological Science*, 17(3):192–193, Mar. 2006.

[41] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[42] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM Computer and Communications Security (CCS'09)*, Chicago, IL, USA, November 2009.

[43] N. C. Rowe. Designing good deceptions in defense of information systems. In *the Annual Computer Security Applications Conferencen (ACSAC'04)*, Tucson, AZ, USA, December 2004.

[44] E. Shi, Y. Niu, M. Jakobsson, and R. Chow. Implicit authentication through learning user behavior. In *Information Security Conference (ISC'10)*, Boca Raton, FL, USA, Oct. 2010.

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

[46] C. Strapparava and R. Mihalcea. Learning to identify emotions in text. In *ACM Symposium on Applied Computing (SAC 2008)*, Fortaleza, Brazil, March 2008.

[47] Visa. Verified by Visa FAQ & credit card security. Online FAQ. http://usa.visa.com/personal/security/visa_security_program/vbv/verified_by_visa_faq.html#anchor_15.

[48] Z. E. Ye, S. Smith, and D. Anthony. Trusted paths for browsers. *ACM Transactions on Information and System Security (TISSEC)*, 8(2):153–186, 2005.

[49] J. Youll. Fraud vulnerabilities in SiteKey security at Bank of America. Technical article (July 18, 2006). http://www.cr-labs.com/publications/SiteKey-20060718.pdf.

[50] J. Yuill, D. Denning, and F. Feer. Using deception to hide things from hackers: Processes, principles, and techniques. *J. Information Warfare*, 5(3):26–40, 2006.