You've Got (a Reset) Mail: A Security Analysis of Email-Based Password Reset Procedures

Tommaso Innocenti[1], Seyed Ali Mirheidari[2], Amin Kharraz[4], Bruno Crispo[2,3], and Engin Kirda[1]

[1] Northeastern University
360 Huntington Ave, Boston, MA 02115, United States
{innocenti.t,e.kirda}@northeastern.edu
[2] University of Trento
via Calepina, 14 - I-38122 Trento, ITA
{seyedali.mirheidari,Bruno.crispo}@unitn.it
[3] KU Leuven
Oude Markt 13, 3000 Leuven, Belgium
[4] Florida International University
11200 SW 8th St, Miami, FL 33199, United States
ak@cs.fiu.edu

Abstract. The password recovery process is a critical part of a website's functionality. Many websites that provide online services to their users also need to solve the problem of allowing their users to reset their passwords (e.g., if they have forgotten it). A popular, established technique for allowing a user to recover a lost account is to allow her to send a reset link to her own account via email. Although it might seem easy at a first glance, the security requirements of the password recovery process require web sites to carefully design each step of the process to be resilient even in the presence of an attack. In this paper, we present an in-depth security analysis of the email-based recovery mechanisms of a wide range of web applications. By manually registering accounts and triggering the password recovery process for each website, we were able to study the password reset mechanisms of web sites from three different groups in the Alexa Top 5K (i.e., popular sites, medium popular sites, low popular sites). In this work, we show that the lack of standards in the password recovery process plagues many websites with security weaknesses, and negatively influences the security of the reset process itself. We also show that concrete password-recovery reset attacks can be launched against a high percentage of websites that might even lead to account takeover.

Keywords: Web applications, Account recovery, Password resets

1 Introduction

Web applications have been historically an attractive target for adversaries. They are often open to the public-facing Internet and are designed to handle critical tasks and valuable data [27]. Therefore, any flaws in their identity and account access management can have a significant impact on the integrity and confidentiality of the protected resources. This paper primarily focuses on the security of the account recovery process, a set of steps users have to follow to re-gain access to their accounts. We provide an empirical study on what is missing in the current implementation of account recovery mechanisms and how some of the flaws introduce significant risks with consequential impacts.

Account management has never been an easy task for normal users. Reports show that users, on average, possess 80 accounts [12]. Hence, it is not very surprising if users lose their access to their accounts and generate password recovery requests from time-to-time [21]. Although modern password managers [25] have shown to be effective in facilitating account management, the need to have a well-designed recovery mechanism has not been diminished. Users may need to update their accounts and change their passwords for many reasons. Consequently, having a robust password recovery is undeniably a critical service to maintain the security posture of web applications and protect users' accounts from unauthorized access.

Despite the importance of password recovery, the necessary details to implement and enforce the mechanism do not seem to be sufficient. The main source of guidance is OWASP [26] which mainly focuses on general requirements without providing specifics on how to implement or assess the security of the procedure. Consequently, little knowledge is available to web developers on how to create a secure and usable account recovery process. This lack of guidance has provided a unique opportunity for adversaries [7,11,8] to steal sensitive information.

Our work is guided by three primary research questions. First, how do websites implement the account recovery process? Second, how prevalent are account recovery problems? And third, what are the immediate threats of misconfigured recovery processes? To answer these questions, we built an analysis pipeline to collect a real-world dataset of the account recovery process in 366 websites in the Top 5K Alexa list. [5] This dataset includes information about the login page, password reset request page, and recovery link. We defined eight implementation controls in account recovery based on OWASP guidelines and ran a semi-automated experiment to empirically analyze if websites satisfy these requirements.

Our analysis shows that insecure practices are prevalent. Our measurements revealed that the account recovery mechanism in 72% of the websites is affected by at least one implementation weakness. For instance, 147 (40%) of the websites were generating multiple valid tokens, providing an opportunity to issue unauthorized password reset requests. Among all websites, 163 (45%) did not send password change request notifications to the account owner, leaving users with no warning in case of unauthorized password reset. We also found that 82 (22%) websites suffering from login CSRF, where users can be tricked into performing the password reset and get authenticated as the attacker, allowing the attacker to observe all user's interactions with the site. Despite the broad

[5] Our data collection infrastructure will be made open source.

set of sites analyzed, our measurement findings represent a lower bound of the potential weaknesses in the wild.

Our analysis also shows that OWASP guidelines are only partially deployed in practice. Since there was no concrete implementation standard for account recovery, we used the OWASP guidelines as the "metric" to determine the adherence of websites to these basic security recommendations. Our results show that only 13% of the websites were following the OWASP guidelines. We observed that 82% of the password reset procedures on the test websites were based on sending reset links. In most cases, we observed violations of OWASP guidelines. For instance, 52 websites were sending the tokens via insecure links, 75 websites had a very long link expiration window (i.e., more than 24 hours). We were able to use the reset links after 24 hours. We also observed that in 21 websites, the generated token could be used multiple times. Among other empirical evidence, our data shows that the security controls in account recovery mechanisms are almost missing – leaving significant opportunities for abuses.

Finally, our experiments reveal that the weaknesses in the account recovery process can have consequential impacts on the security posture of the websites. For instance, we observed that attackers can mislead the remote server to generate poisoned reset mails during the password reset request. Users can be tricked to click on the poisoned reset link providing an opportunity for attackers to launch login CSRF attacks where they can collect the user's reset token and potentially perform an account takeover. Moreover, it also allows the adversary to retrieve almost all the activities on the target account (e.g., message exchange, credit card entry) by misleading the victim to reset an attacker-controlled account.

While the weaknesses mentioned in this paper may not always result in blatant security vulnerabilities, they are indicators that developers are failing to follow robust practices, leading to a more fertile environment for adversarial operations. We provide an empirical look at the security consequences of these unsavory practices such as login CSRF and header manipulation. We hope that this work serves to raise awareness about the importance of defining reliable mechanisms for the account recovery in the web ecosystem. We also hope our approach will prove useful to the web security community and open the door for future solutions.

This paper's contributions are summarized as follows.

- We propose a methodology to identify weakness in email-based password recovery process.
- We present a measurement of common weaknesses in password recovery among the Alexa Top 5K.
- We study a set of web-based attack scenarios on email-based password recovery, and quantify the prevalence of them among high-profile sites.

The remainder of this paper is structured as follows. In Section 2, we present the related work. In Section 3, we provide the background information introducing the OWASP guidelines and the related threat model. In Section 4, we explain the adopted methodology and the data collection infrastructure used to conduct the study. In Section 5, we present our findings.

2 Related Work

The security community has introduced several fallback authentication mechanisms for account recovery, ranging from knowledge-based authentication (e.g., security questions) to possession-based authentication (e.g., OAuth, tokens, security cards, and email verification). In the following, we will discuss works that primarily focus on the security of account recovery mechanisms.

Password Reset via Email Verification. Although email-based account recovery [4] has reached a wide adoption among sites [29], the procedures to deploy the mechanism is not very well-defined [14,13,28,30,3,29]. The motivation for some of the prior work was to analyze the effectiveness of account recovery emails [2,1,20,15]. The research showed that malformed recovery emails that do not inform users of the reset link's validity time, or that do not warn users about keeping the link confidential may introduce a vulnerability. In a different approach, Raponi et al. [29] proposed a technique to protect users from service provider-level attacks. Despite the open-source availability of the solution, the study is limited in scope and the vulnerabilities covered in the work. In our work, we investigated eight common weaknesses and attack scenarios based on OWASP guidelines [26] as described in Section 4.

Password Reset via SMS OTP. Researchers have also performed several security analyses of the implication of SMS OTP methods in the authentication process [10,24,33,22,19]. For instance, AUTH-EYE [22] proposed an automated approach to detect implementation flaws in the authentication modules of a program. The analysis showed that in 98.5% of the test cases, the applications violated different security rules in generating (OTP randomness, length) and verifying the SMS OTP code (e.g., allowed retry attempts and renewal interval). In a similar study, Mulliner et al. [24] conducted a measurement study of the SMS OTP security architecture by introducing several weaknesses and attack scenarios. They demonstrated that intruders could obtain the SMS OTP utilizing SIM swapping or the wireless interception attacks. Dmitrienkoet et al. [10], investigated two-factor authentications of high profile Internet service providers and discovered several weaknesses which could be exploited to circumvent SMS-based authentication in four larges online banks applications as well as Google. In a recent study, Zeyu Lei et al. [19] performed a systematic study on mobile apps' authentication schemes based on SMS OTP. Their study not only discovered vulnerable mobile apps with hundreds of millions of installations, but also revealed several flaws in core API implementations of mobile operating systems.

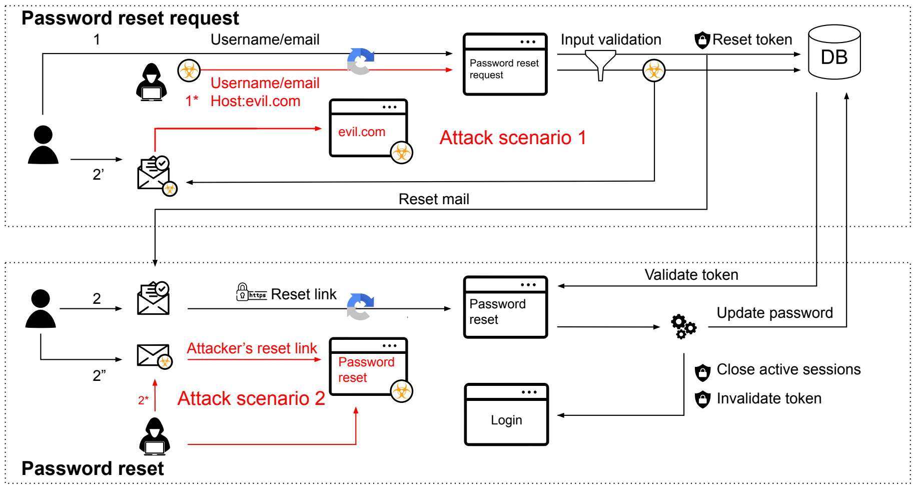


Fig. 1. Password reset flow

3 Background

In this section, we first discuss the OWASP guidelines [26] for resetting user passwords. Then, we provide information on the attack scenarios that we used as complementary security checks for our measurement.

3.1 OWASP Guidelines

The main reference that is publicly available for password reset is provided by OWASP. The guideline divides the password reset process into two parts; (1) initiating the password reset request, (2) processing the password reset request. In the following, we briefly describe each step.

Initiating the Password Reset Requests. The first part of the procedure is to initiate the password reset as presented in Figure 1. For this step, the OWASP guidelines suggest providing to the users a consistent message for existing and non-existing accounts with a constant time for each reply. This approach reduces the risk of classic timing side-channel attacks on password management [5] and also protect websites from possible user enumeration attacks [6,4]. Malicious web bots can target the password reset mechanism. To reduce the impact of such adversarial attempts on the password reset, the guidelines suggest the implementation of a protection mechanism against automated submission as CAPTCHA or other rate-limiting controls [34]. Moreover, before processing the request, the received input should be properly sanitized with an input validation alongside an SQL injection prevention method that will protect the website's database. After the input validation, the website first stores the generated reset token in a database and then includes it in the reset link. Finally, a reset email containing the reset link is sent to the user.

Responding to Password Reset Requests. The second part of the procedure is the actual password reset that starts with the Action 2 of Figure 1 where the user clicks on the reset-link received and accesses the password reset page. The guidelines suggest that users should confirm the new password twice – respecting a consistent password policy with the rest of the web application. The site should then invalidate the reset token and store the new password following secure practices (e.g., using a proper hashing function). Finally, the site should inform the user of the successful password reset with an email that should not include the new password.

After the successful password reset, the guidelines suggest redirecting the users to the normal authentication procedure instead of automatically logging them since this will complicate the session code handling, and could potentially introduce CSRF vulnerabilities. The last suggestion is to provide the users the option to close the open sessions of the account or automatically invalidate them requesting them to provide the newer credentials.

3.2 Attack Scenarios

The attack scenarios presented are derived from the OWASP guidelines [26]. These attacks are directly related with email-based password recovery process that represent the majority of the recovery methods adopted in the wild. With the studied attack scenarios, we demonstrated how misconfigurations in the recovery process could be used to directly target a website's users. We first explain header manipulation issues, and then discuss login CSRF problems.

Header Injection. The first attack scenario starts at the initiation of the password reset procedure [17,16,18] due to improper processing of the network headers. That is, the attacker sends a poisoned password reset request marked as 1* in Figure 1 and manipulates the host field in the network header of the requests. This can mislead the site's server that, without a proper sanitization, will generate the reset link using the poisoned field. As a result, the attacker without needing any interaction with the victim, mislead the websites' server to send a poisoned reset mail to the victim. The only information the attacker needs to initiate the attack is the victim email. This attack thanks to the generation of the poisoned email by the website's server will easily bypass victims' inbox spam filter. Moreover, since most of the resetlink included in reset mail is hide behind a button, the poisoned email generated can easily mislead the victim in click the malicious link(Action 2') increasing the chance of a successful attack.

Login CSRF. As depicted in Figure 1, a user would choose a new password in the final step of the password reset process. Since the user is expected to submit a valid reset link that is confidential, some applications immediately set an authorization cookie and redirect the user to the dashboard, bypassing the normal authentication. Skipping the authentication phase occurs in many applications as they presume that only the valid user would have access to this concrete reset link. Unfortunately, this assumption creates a security risk. In login CSRF attacks, an attacker would request a password reset link for its own account, and then send this link to the victims(Action 2*), encouraging them to click on the link to update the password.

The application would then validate the password reset token, and would set the proper authorization cookie associated with the reset token. This process is very dangerous since the victims(Action 2”) use the attacker’s authorization cookies (i.e., the owner of the reset link) to interact with the website. In fact, the application recognizes the victim as the account owner because of the possession of the password reset link. As a result, any of the user’s activities in the attacker’s account (e.g., web search history, exchanged messages, etc.), can later be accessed by the attacker. Moreover, if the user saves any personally-identifiable or financial information (e.g., credit card numbers), this information would be accessible to the attacker as it is saved in an account under the attacker’s control.

Motivating Example. It is critical to evaluate how websites initiate and process account recovery requests. Account recovery problems are less well-studied security issues with significant impact. For instance, springer.com, a well-known scientific publisher, has multiple issues in initiating the password requests. We observed that password reset tokens in this website do not expire. OWASP strictly suggests single-use reset tokens to minimize the risk of account takeover and the abuse of the reset mechanism. We also observed that users are provided with no notification about the password recovery requests. The notification allows users to identify unauthorized attempts for hijacking accounts. As other examples, Seattlenews.com, a very popular news website in the US Northwest region, and rakuten.co.jp, a known Japanese shopping website all suffer from similar types of account recovery problems. The examples we discussed here are among the most popular websites with a significant number of users and web traffic. Consequently, issues in password recovery can have consequential impacts on the security of users as well as the web application. Our experiments, discussed at length in Section 5, show that these issues occur frequently. In Section 4, we describe eight different classes of weaknesses in the account recovery mechanism, and incorporate them to evaluate the security of real-world recovery mechanisms.

4 Methodology

This section introduces our data collection infrastructure, and the methodology we used to study the security of the password reset procedure in real-world websites.

4.1 Measurement Setup

The initial step of the measurement is to select websites and form a test corpus. We selected 900 websites from the Alexa Top 5,000 websites. We divided the sites into three groups based on their popularity. As presented in Section 5.3, this site selection enabled us to analyze the behavior of each site group allowing us to identify how sites with different popularity exhibit similar results.

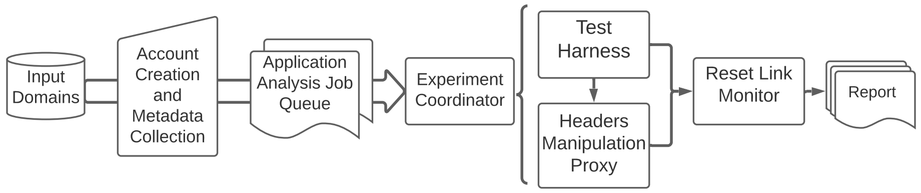

Input Domains
Account Creation and Metadata Collection
Application Analysis Job Queue
Experiment Coordinator
Test Harness
Headers Manipulation Proxy
Reset Link Monitor
Report

Fig. 2. Data collection infrastructure.

However, we were able to create accounts only on 513 websites since for some websites, we were not able to provide all the required information during the account creation process. For instance, some websites did not have a public login page or required a pre-approval process such as bank account information. Also, some of the 900 sites required a specific class of data such as a local phone number or a Social Security Number (SSN). In some cases, the confirmation link was not received during the account generation process. To measure the security of the password reset procedure, we created a user account for each site under test. During our measurements, we observed that some data deduplication was necessary. That is, institutions such as Google Inc. had several websites with different Country Code Top-Level Domains (ccTLDs) where the reset link was redirected to the main domain. For instance, the password reset requests on google.es or google.it were redirected to google.com. We removed these cases from our analysis which were 147 websites to avoid identical measurements. This initial setup allowed us to collect metadata for each site under test such as the presence of CAPTCHAs. The metadata helped us in the account recovery process where we could decide how to interact with the target websites. Figure 2 shows the overall data collection pipeline.

4.2 Data Collection Infrastructure

As presented in Figure 2, the data collection infrastructure is coordinated by the *Experiment coordinator* that takes as input the filtered domains and performs test probes using several high-level components: (1) the *test harness*, (2) the *reset link monitor*, and (3) the *header manipulation proxy*.

Test Harness. is in charge of managing all the *account interfaces* and communicating with the header manipulation proxy to generate the *manipulated account reset*. This component is responsible for reproducing the user's interaction with the website. The interaction with the website was done by writing a Python script based on Selenium Chrome web driver. The crawler uses *CSS selectors* to identify the login element in the page and issue the password reset request. To verify the correct validation of a correct login, a visual check was performed to make sure that the browsing session works well in practice. Moreover, we note that 44 sites were using CAPTCHA services in their login page and that 90 sites require an explicit CAPTCHA verification before initiating the password reset

request. Since automatic interaction was not possible in these cases, we manually ran the experiment to find and reset the password.
Header Manipulation Proxy. is in charge of modifying the request generated by the test harness. The header manipulation proxy modifies the generated requests from the test harness including markers in two different header types: Host header and non-Standard Headers. The markers are used to modify the requests that are subsequently used by the reset link monitor to pinpoint the creation of a poisoned reset mail from the web applications.
Reset Link Monitor. monitors email accounts corresponding to user accounts created in the setup step. Received emails are scanned using a combination of regular expressions and inspection of HTML mail body to extract all links contained in the message body and check the presence of injected markers. The extracted links are then filtered using a keyword heuristic to retain password reset links and discard all others.

4.3 Password Reset Testing Methodology.

Using OWASP guidelines [26] as a benchmark, we derived the following tests to evaluate the OWASP adherence of the website's recovery process.
Reset Link Validity Window($\mathbf{T}_W$). The length of time a reset link is valid after being issued can have security consequences. A long validity window can introduce opportunity for an attacker to abuse a stolen link. That being said, there is also an inherent usability trade-off in this mechanism. That is, a too short validity window makes it difficult to legitimately use a link before it expires. Since OWASP guidelines only suggest an "appropriate time validity", we chose 24 hours as a reasonable trade-off between reset link validity window security and usability. To test the validity window T_W for a site s, a fresh link l is requested. This reset link is then stored for 24 hours before it is used to attempt a password reset. Effectively, the test determines whether or not $T_W(s) > 24h$.
Multi-Use Reset Links($\mathbf{T}_U$). Another security-relevant property of reset links is whether the reset links are valid for multiple uses. OWASP strictly suggests a single use reset link for each reset token. This prevents, for example, an attacker from gaining access to an email account containing older reset links and reusing them to perform a password reset without generating new reset link emails. This test measures whether a site s allows resetting links to be used multiple times, denoted by T_U. First, a reset link l is requested. Then, we attempt to use it to perform a password reset twice in succession. If the second attempt succeeds, then s does not invalidate l on first use and so $T_U(s) > 1$.
No Password Change Confirmation($\mathbf{T}_C$). As suggested by OWASP, websites should notify users after a successful password change. These notifications allow users to recognize when an attacker is attempting an account takeover via password reset requests. Since these notifications are sent as a password change confirmation email, we measure this property for each website. The test measures whether a site s confirms password changes, denoted by T_C. To carry out the test, we first request a reset link l and then use it to perform a password

reset. Next, the email monitor waits to receive a confirmation email. If one is not received within 1 hour, we consider the test to have failed, i.e., $T_C(s) = \mathsf{false}$.

No Session Invalidation($\mathbf{T}_I$). After a password reset has been performed, OWASP suggests allowing the users to shutdown all the account's active sessions. This is a useful practice because if a password reset was requested due to an account compromise, failing to invalidate existing sessions would allow an attacker to maintain persistence. Hence, we tested which websites by default do not close the account's active session after a password reset. To measure whether the website s invalidates existing sessions after a password reset (T_I), a reset link l is requested. Then, a fresh session is created by authenticating to s. In a *separate* unauthenticated browser instance, l is used to reset the account's password. Then, we check whether u_1 is still authenticated to s. If so, then $T_I(s) = \mathsf{true}$. To prove that the site s closes the session u_1, we performed a manual verification interacting with the open session for a maximum of 60 seconds. The manual verification is necessary due to the heterogeneous behavior of websites where in some cases a page refresh would invalidate the session and in some others, a more complex interaction with the site is needed for session invalidation.

Insecure Reset Link($\mathbf{T}_S$). OWASP suggests that all the reset links should use HTTPS. Otherwise, users run the risk of falling victim to several attacks. For instance, an attacker could act as a man-in-the-middle and intercept the new account password. Furthermore, a passive network attacker could sniff the new account password if HTTPS redirection is not performed and subsequent requests are transmitted in the clear. This test, which we denote T_S, simply involves checking whether a reset link l for site s uses the HTTPS scheme. If not, then $T_S(s) = \mathsf{true}$.

Multiple Valid Reset Links($\mathbf{T}_N$). If a site receives multiple password reset requests, it may issue multiple reset links.[6] However, each new reset link represents another opportunity for leakage and abuse by an attacker. Thus, although not specified in the OWASP guideline [26], from a security perspective, it is preferable to limit the number of reset links that are valid at any point in time, ideally to one. This test measures whether a site s allows multiple, simultaneously valid reset links; we denote the number of simultaneously valid links allowed by a site as T_N. First, two reset links l_1, l_2 are requested. Then, we attempt to use l_1 to initiate a password reset. If the reset succeeds, then this indicates that $T_N(s) > 1$. In particular, it means that s does not invalidate older links when a new one is issued.

HTTP Header Injection($\mathbf{T}_H$). As suggested by OWASP, websites should validate the user's input before processing it. This includes secure handling of HTTP headers since malicious content injection of various kinds could be reflected in reset emails. These vulnerabilities can enable several distinct attacks. For instance, if a site uses a password reset request's Host header to generate reset links, an attacker could explicitly set the Host header when issuing a reset request to point to an attacker-controlled origin. This would allow the attacker

[6] In principle, a site could also refuse to issue a new reset link while prior links are still valid. In our experiments, we never observed this to occur.

to phish victims and intercept both the old and new account passwords. Alternatively, arbitrary content injection vulnerabilities could be abused to carry out XSS attacks against victims that receive reset emails.

To test whether a site s is vulnerable to HTTP header injection (T_H), we used our test harness to issue a password reset request, but this time, using the Headers manipulation proxy, we intercept it, inject unique markers and then send it to the site s. In particular, we selected a number of standard and non-standard HTTP headers to inject markers into. These headers include Host, Origin, Referer, X-Host, X-Forwarded-Host, X-Forwarded-For, X-Forwarded-Server, Proxy-Host, Destination, True-Client-IP, Client-IP, X-Client-IP, X-Real-IP, X-Originating-IP, CF-Connecting-IP, X-Original-URL, X-HTTP-DestinationURL, X-Arbitrary, X-Forwarded-Proto, Proxy, Contact, From, Forwarded, X-Wap-Profile, and Profile. The email monitor collects the password reset email. Once it is received, it is scanned for the presence of an injected marker. If one is found, then the site is considered to be *conditionally* vulnerable to HTTP header injection; that is, $T_H(s) = \mathsf{true}$. The reset email is flagged for later manual confirmation.

Login CSRF($\mathbf{T}_L$). Immediate redirection of a user to the authenticated page after a successful password reset without an intervening re-authentication represents a potential login CSRF vulnerability. For this reason, OWASP suggests redirecting users to the login process instead of automatically logging in users. In fact, by targeting a vulnerable site, an attacker could send a password reset email to a victim for an attacker-controlled account. If the victim is tricked into performing the password reset, she will be authenticated as the attacker, allowing the attacker to observe all of the victim's interactions with the site. To test for the presence of a login CSRF vulnerability at site s, we request a rest link l and then use it to perform a password reset. After the password, reset we manually verify if the site si, without requesting the new password, automatically logs in the user. If so, then $T_L(s) = \mathsf{true}$.

4.4 Limitations

Due to the inherent limitations of the proposed approach, the reported findings and results in this paper should be considered as a lower bound. For instance, we exclude websites that require entering the social security numbers, corporate email addresses, or credit card information. Furthermore, in the header manipulation assessment, our crawler tested only fifteen popular HTTP headers, while the origin server might be still vulnerable to other headers which have not been tested in our methodology. Since our crawler should send a new request for each test case, some websites blocked our test accounts after a number of password reset attempts. We stopped testing these websites. However, the implementations of these websites might still be vulnerable.

4.5 Ethical Considerations

We defined a set of security controls in our experiments to make sure that the measurements will not cause any damage to the sites under analysis. For instance, we ran a minimum number of password reset requests to limit the amount of traffic generated by our measurement tools. During our measurements, we never injected any malicious code in the network traffic, nor tried to access accounts that were not under our control. Since our study measures the actual presence of weaknesses in real-world websites, we did not publicly disclose the vulnerable websites and contacted the vulnerable websites directly.

We followed the recommendations proposed by works such as [23,32,31]. We first used an open-source vulnerability disclosure and bug bounty program database to obtain the sites' contact[9]. From the database, we were able to obtain the contact information of 62 vulnerable sites. In addition, we identified sites that use a broker (e.g., HackerOne or Bugcrowd) as a designed channel to disclose vulnerabilities. We then manually contacted each site following the platforms' processes.

Unfortunately, not all the vulnerable sites we identified were included in the database. For the remaining 200 sites, we used the WhoisXML API to gather the sites' contact associated with the registered domain. Once we obtained all sites' contact information, we used these as input for our automated email script to automatically generate and send a custom email for each site. In the communication, we used a verified mail account, as well as our contact information and a method to verify our identity. Along with the weakness that afflicted each site, we included a link to the OWASP guidelines used to measure the process security.

Of all the contacted sites, we received an acknowledgment from 38 sites (14.5%). The sites contacted by email showed lower responsiveness compared to the sites that used a broker. In fact, we received 19 replies out of 243 sites. 8 sites reported the issue to their dev team and another 8 marked our report as informative. 3 sites fixed the issue that we reported.

All the 19 sites contacted through a broker replied to our report, with 8 of them marking our report as duplicated. Some of the duplicated reports were dated as far back as 2014 – indicating that the website lacked a fix. A known site of video streaming confirmed that they will fix the issue. The remaining sites acknowledged the issue, but due to restricted bug bounty program scope, our report was marked as informative without providing any information about possible future fixes.

5 Security Measurement

In this section, we provide the result of our measurement, and present the security implications of our findings.

5.1 Account recovery implementation

Password Recovery Type. Since analyzing email-based recovery processes are the main focus of this work, the first step is to identify the recovery methods adopted by the selected websites. The most reliable way to discover the recovery

Table 1. Recovery types summary

Recovery Type	**Channel**	**# Sites**
Text-Msg	SMS	5 (1.4%)
Original Password	E-mail	7 (1.9%)
One-time Security Code	E-mail	27 (7.4%)
Temporary Password	E-mail	25 (6.8%)
Password Reset Link	E-mail	302 (82.5%)
Total		366 (100%)

Table 2. Common Weaknesses Statistics

Weakness	**All Sites**
No Change Notification (NCN)	163 (44.5%)
Multiple Valid Tokens (MVT)	147 (40.2%)
No Session Termination (NST)	139 (38.0%)
Login CSRF (LC)	82 (22.4%)
No Expiration (NE)	75 (20.5%)
Insecure Reset Link (IRL)	52 (14.2%)
Multi Use Token (MUT)	21 (5.7%)
Headers Manipulation (HM)	6 (2.0%)
Total	262 (71.6%)

method is to perform a password reset request on each of the selected websites. This step of the initial filtration, described in Section 4.1, allowed us to remove websites that do not provide a password recovery mechanism as well as websites that consisted of multiple domains. The output of the setup is **366 websites** that constitute our study's site seed. A summary of the password recovery type discovery is presented in Table 1. Our dataset's analysis confirms the findings of [4] that the majority of websites adopt an email-based password recovery procedure. Moreover, it answers our first research question by defining how many websites implement the account recovery process. In fact, the email-based recovery type (361 websites) represents 98.6% of the whole recovery types with only **five** websites that adopt *SMS* as the primary recovery method.

5.2 Recovery Procedure Analysis

Since 302 sites (82.5%) used the reset-link as the recovery method, we focused on the security implementation of the reset tokens included in the reset-link. The reset token is a secret code generated by the website, and included in the reset-link. The token allows opening a temporary session to reset the account's password. Due to its central role, the reset token's security checks play a fundamental part in ensuring the password reset procedure's security. Hence, we allocated four out of six implementation checks to verify its correct implementation and the

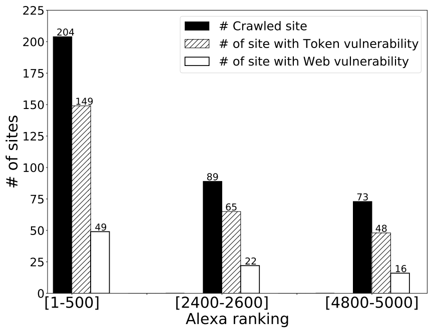


Fig. 3. Sites distribution in Alexa ranking

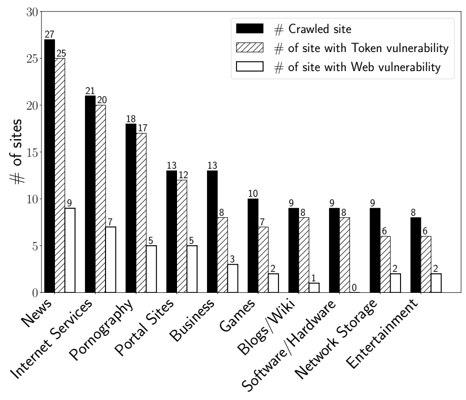


Fig. 4. Top 10 categories

remaining two to verify a correct implementation after password reset (i.e., no session termination, no change notification). The result of our measurement are presented in Table 2.

Combining the result of Table 2 with the data of Figure 6 shows that 57.7% of websites (211 websites out of 366) misimplemented a security check on reset token. For the remaining two implementation checks, we found that 54.0% (198 out of 366 websites) of websites wrongly managed the active sessions after a password reset, or missed a confirmation email after a successful password reset. The tests performed show a large diffusion of misimplementation of the password reset procedure among websites. Moreover, the widespread number of websites with at least one weakness (i.e., 262 out of 366 – 71.6%) answers the second research question affirmatively; That is, the missing of a standard clearly causes a degradation of the password reset procedure's implementation security on websites. The diffusion of weaknesses among websites is a clear indicator of a much-needed strict regulation of this critical procedure urging a practical effort from websites to the research community to solve this overlooked problem.

5.3 Weakness Analysis

Alexa Ranking. As presented in Figure 3, the percentage of websites affected by a weakness is, in the first group, **97%** (198 sites) of the most popular websites, **97.7%** (87 sites) for the second group and, **87.7%** (64 sites) for less popular websites. The result of our measurement shows an homogeneous diffusion of weaknesses among all three different groups.

Category. The analysis of the website categories reveals that all the categories have at least 60% of websites that are vulnerable, with six categories that have at least 89% of the websites that are vulnerable. As presented in Figure 4, from the composition of the sites' weaknesses, it is clear that token weaknesses have

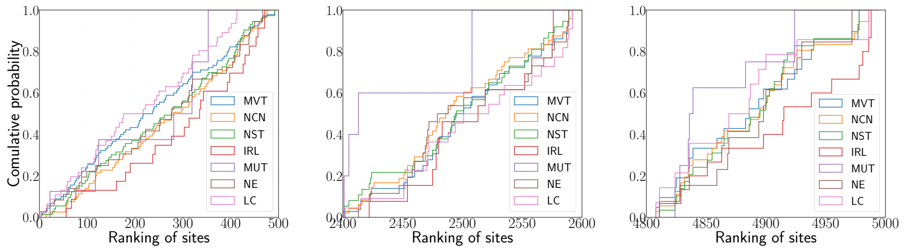


Fig. 5. CDF weaknesses comparison

a higher incidence with respect to the web vulnerabilities. Even though some categories showed a lower percentage in weaknesses, the result obtained shows how the weaknesses we analyzed affect all the website categories indiscriminately, reinforcing the conclusion that the security of password reset procedure is an open and widespread problem.

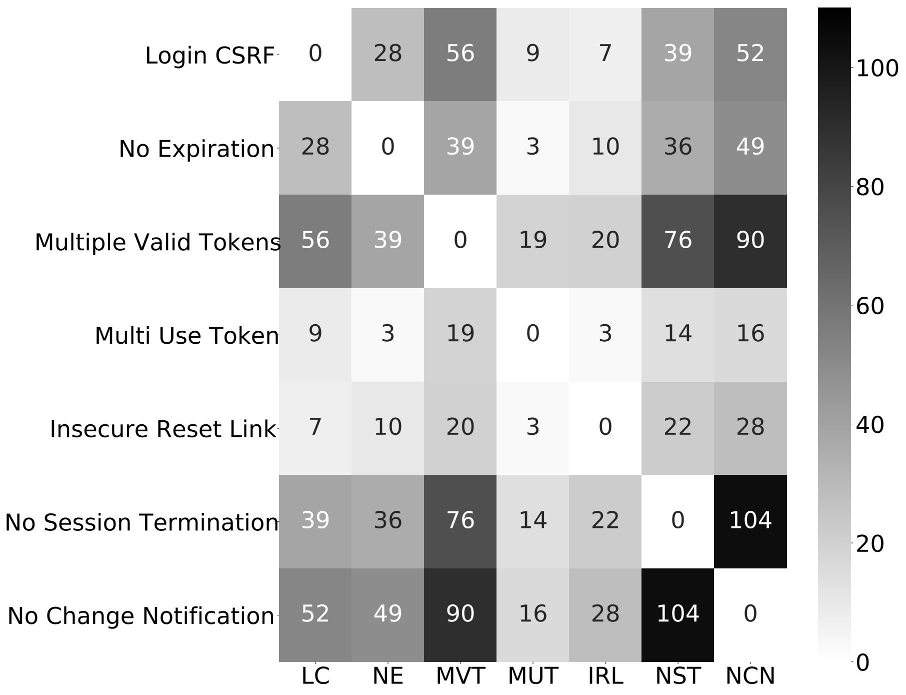
Weakness Distribution. Figure 5 shows the cumulative distribution of websites that exhibit an implementation weakness. For each weakness measured, we represent here the distribution of vulnerable sites in all three sites bucket. A more inclined line shows that a particular weakness affects more popular sites than less popular sites. As shown in the graph, the majority of the weaknesses show a similar trend among sites. Further analyzing the distribution of each bucket of sites, we performed a Mann-Whitney U test and, with a p-value of 0.01, we cannot reject the null hypothesis that all three buckets belongs to the same distribution. This result suggests a counter-intuitive result; that websites are affected regardless of their ranking and popularity.

Furthermore, we investigated the diffusion of weaknesses among websites with the co-occurrence matrix presented in Figure 6. The matrix shows three weakness clusters: Multiple Valid Token, No Change notification, and No Session Termination. These clusters suggest that websites that did not pass one of the security checks have a higher chance of having issues in other security check implementations. The consequence is that even a single misimplementation could endanger the security of the reset procedure. However, websites that expose multiple weaknesses enable the chaining of weaknesses, considerably increasing the severity of the problems.

5.4 Attack Scenarios

3.2

In this section, we present the result of our attack scenarios investigation. While our analysis represents a lower bound of vulnerable sites, it still shows an urgent need for a better password reset procedure implementation. Furthermore, sites should account for every potential attack when designing their re-



	LC	NE	MVT	MUT	IRL	NST	NCN
Login CSRF	0	28	56	9	7	39	52
No Expiration	28	0	39	3	10	36	49
Multiple Valid Tokens	56	39	0	19	20	76	90
Multi Use Token	9	3	19	0	3	14	16
Insecure Reset Link	7	10	20	3	0	22	28
No Session Termination	39	36	76	14	22	0	104
No Change Notification	52	49	90	16	28	104	0

Fig. 6. Co-occurence Matrix

covery procedure to reduce the risk that malicious actors could exploit security assumptions.

Header Injection. In this experiment, our proxy, in two separate account reset requests, manipulated Host header as well and fifteen standard and Non-Standard HTTP headers through injecting specific values. We later investigated all reset email contents and reset links for the presence of a marker (manipulated headers value). Our crawler discovered 6 out of 366 (2.0%) websites echoed our injected marker either in reset link, or inside email contents. The details of each category are described below.

Host Header Injection. In the first step, our proxy replaced the `Host` header in all password reset requests with a specific marker. By investigating all reset emails, our crawler discovered 3 out of 366 (1.0%) websites to be vulnerable, i.e, the generated password reset link pointed to the injected domain name by the crawler. It is noteworthy that our crawler did not detect any changes in the content of the password reset email, which implies `Host` header manipulation would only affect reset links. Leveraging this vulnerability enables an attacker to steal the password reset token and compromise user's accounts. Our results show that only 66 out of 366 (18%) websites accepted password reset requests with manipulated `Host` header and sent the password reset emails, whereas only 3 were found to be vulnerable. For the rest of the websites, our crawler recorded three different behaviors. 96 websites handled requests properly and redirected our crawler to custom error pages. 70 websites returned different 400 error codes (mostly `403 Not Found`), and 8 websites returned `500 Internal Server Error`. For websites that do not accept password reset requests, it is plausible that a variety of middle HTTP components (e.g., Load balancer, Web application firewall, etc.) have blocked requests with manipulated `Host` header and do not route the request to the server. Then, the presented result in this section should be considered as a lower bound because the server can still be vulnerable, but other in the middle protections avoid the server's exposure to vulnerability.

Other HTTP Headers Injection. In the final step, our proxy manipulated the password reset requests by adding fifteen standard and Non-Standard HTTP headers and discovered 3 out of 366 (1.0%) websites to be vulnerable. During the measurement, 198 websites out of 366 (54%) accepted our poisoned request showing a lower bouncing rate compared to the host headers poisoning an older and better documented vulnerability. The lower popularity and documentation of Non-Standard HTTP headers open the door to unexplored weakness as showed by the result of our measurement. Two of three cases were vulnerable to `X-Forwarded-Host` header, and the servers used these header values to create a password reset link. Obtaining the password reset link would enable the attacker to steal the password reset token and perform an account take over.

The third site was exploited with manipulation of a different header, namely `x-originating-ip`, while the site was vulnerable to altering neither `Host` nor `X-Forwarded-Host` headers. The result of manipulating `x-originating-ip` was notable in that the marker we injected in the field was echoed in the contents of the email. Therefore, it would be possible to inject arbitrary phishing contents in the password reset email. Another notable observation was specifically observed in `anonymous.com` site, which was including not only the password reset link in the password reset email, but also the IP address of the requester. This IP address would be set as the value of `x-originating-ip` header. Interestingly, when we populated `x-originating-ip` with our marker, this value was the one which was shown in the anonymous password reset email. We think this observation could be justified with assuming the existence of some middle network components or servers, which are responsible to find the requester's IP and then append that IP address to the header, without overwriting the value we set. When the results are sent to the server, it shows the first value of `x-originating-ip`, which had been set by us. These results confirmed our hypothesis that manipulating Non-Standard headers are an effective variation of header manipulation and can target password reset processes either by reset link poisoning or email content manipulation.

Login CSRF. The attack scenario presented in Section 3.2 is a variation of the well-known Login CSRF(i.e., Auth-CSRF). Login CSRF was first introduced in 2009 [15]. In the attack, an attacker forces the victim to login to the attacker's account by sending a forged authentication request with the attacker's credentials through the victim's browser. As described in Section 4.3, after completing the password reset process, we manually investigated the landing pages to verify if the browser session is authenticated. In case of an authentication landing page, the site would be tagged as vulnerable. As a result of our experiments, 82 out of 366 (22.4%) websites were vulnerable to login CSRF.

6 Conclusions

One of the most security-critical aspects of a website's functionality is the password recovery service. It has been long known that malicious actors often target the password recovery process to hijack a victim's account (e.g., by guessing their secret recovery questions). A popular, established technique for allowing a user to recover a lost account is to allow her to send a reset link to her account via email. The security requirements of the password recovery process requires web sites to carefully design each step of the process to be resilient even in the presence of an attack.

This paper presented a security analysis of the email-based recovery mechanisms of a wide range of web applications from the Alexa Top 5K. We studied groups of popular, medium popular, and low popular websites, and manually registered accounts on these websites. Our work shows that the lack of standards in the password recovery process plagues many websites with security weaknesses, and also negatively influences the security of the reset process itself. We also show that concrete password-recovery reset attacks (e.g., login CSRF, header manipulation) can be launched against a significant number of websites that might even lead to account takeover. We hope that this paper will pave the way in highlighting the importance of improving the email-based account recovery mechanisms in real-world websites.

7 Acknowledgments

We would like to thank anonymous reviewers for reading the paper carefully and making helpful comments. This work was supported by National Science Foundation under grant CNS-1703454. This work was also partially supported by Secure Business Austria.

References

1. Al Maqbali, F., Mitchell, C.J.: Email-based password recovery-risking or rescuing users? In: 2018 International Carnahan Conference on Security Technology (ICCST). pp. 1–5. IEEE (2018)
2. Al Maqbali, F., Mitchell, C.J.: Web password recovery: A necessary evil? In: Proceedings of the Future Technologies Conference. pp. 324–341. Springer (2018)

3. Bonneau, J., Bursztein, E., Caron, I., Jackson, R., Williamson, M.: Secrets, lies, and account recovery: Lessons from the use of personal knowledge questions at google. In: Proceedings of the 24th international conference on world wide web. pp. 141–150 (2015)
4. Bonneau, J., Preibusch, S.: The password thicket: Technical and market failures in human authentication on the web. In: WEIS (2010)
5. Cao, Y., Chen, Z., Li, S., Wu, S.: Deterministic browser. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 163–178 (2017)
6. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-channel leaks in web applications: A reality today, a challenge tomorrow. In: 2010 IEEE Symposium on Security and Privacy. pp. 191–206. IEEE (2010)
7. Conikee, C.: Case study : Exploiting a business logic flaw with github's forgot password workflow. https://medium.com/@chetan_conikee/case-study-exploiting-a-business-logic-flaw-with-githubs-forgot-password-workflow-discovered-d4d36ee3dd16 (December 2019)
8. Corporation, T.M.: Cwe-640: Weak password recovery mechanism for forgotten password. https://cwe.mitre.org/data/definitions/640.html
9. Disclose.io: Open-source tools to help hackers and organizations make the internet safer, together. https://disclose.io, online; accessed 20 February 2021
10. Dmitrienko, A., Liebchen, C., Rossow, C., Sadeghi, A.R.: Security analysis of mobile two-factor authentication schemes. Intel Technology Journal **18**(4) (2014)
11. Gracey, J.: Hacking github with unicode's dotless 'i'. https://eng.getwisdom.io/hacking-github-with-unicode-dotless-i/ (November 2019)
12. Hanamsagar, A., Woo, S.S., Kanich, C., Mirkovic, J.: Leveraging semantic transformation to investigate password habits and their causes. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. pp. 1–12 (2018)
13. Jakobsson, M., Stolterman, E., Wetzel, S., Yang, L.: Love and authentication. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 197–200 (2008)
14. Just, M.: Designing and evaluating challenge-question systems. IEEE Security & Privacy **2**(5), 32–39 (2004)
15. Karlof, C., Tygar, J.D., Wagner, D.A.: Conditioned-safe ceremonies and a user study of an application to web authentication. In: NDSS (2009)
16. Kettle, J.: Practical http host header attacks. https://www.skeletonscribe.net/2013/05/practical-http-host-header-attacks.html (May 2013)
17. Kettle, J.: Cracking the lens: targeting http's hidden attack-surface. https://portswigger.net/research/cracking-the-lens-targeting-https-hidden-attack-surface (July 2017)
18. Kettle, J.: Collaborator everywhere. https://github.com/PortSwigger/collaborator-everywhere (May 2018)
19. Lei, Z., Nan, Y., Fratantonio, Y., Bianchi, A.: On the insecurity of sms one-time password messages against local attackers in modern mobile devices. In: Proceedings of the 2021 Network and Distributed System Security (NDSS) Symposium (2021)
20. Li, Y., Wang, H., Sun, K.: Email as a master key: Analyzing account recovery in the wild. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. pp. 1646–1654. IEEE (2018)
21. Lovisotto, G., Malik, R., Sluganovic, I., Roeschlin, M., Trueman, P., Martinovic, I.: Mobile biometrics in financial services: A five factor framework. University of Oxford, Oxford, UK (2017)

22. Ma, S., Feng, R., Li, J., Liu, Y., Nepal, S., Bertino, E., Deng, R.H., Ma, Z., Jha, S.: An empirical study of sms one-time password authentication in android apps. In: Proceedings of the 35th Annual Computer Security Applications Conference. pp. 339–354 (2019)
23. Mirheidari, S.A., Arshad, S., Onarlioglu, K., Crispo, B., Kirda, E., Robertson, W.: Cached and confused: Web cache deception in the wild. In: 29th {USENIX} Security Symposium ({USENIX} Security 20). pp. 665–682 (2020)
24. Mulliner, C., Borgaonkar, R., Stewin, P., Seifert, J.P.: Sms-based one-time passwords: attacks and defense. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 150–159. Springer (2013)
25. Oesch, S., Ruoti, S.: That was then, this is now: A security evaluation of password generation, storage, and autofill in browser-based password managers. In: USENIX Security Symposium (2020)
26. OWASP: Forgot password cheat sheet, https://cheatsheetseries.owasp.org/cheatsheets/ForgotPasswordCheatSheet.html
27. Preibusch, S., Bonneau, J.: The password game: negative externalities from weak password practices. In: International Conference on Decision and Game Theory for Security. pp. 192–207. Springer (2010)
28. Rabkin, A.: Personal knowledge questions for fallback authentication: Security questions in the era of facebook. In: Proceedings of the 4th symposium on Usable privacy and security. pp. 13–23 (2008)
29. Raponi, S., Di Pietro, R.: A longitudinal study on web-sites password management (in) security: Evidence and remedies. IEEE Access **8**, 52075–52090 (2020)
30. Schechter, S., Brush, A.B., Egelman, S.: It's no secret. measuring the security and reliability of authentication via "secret" questions. In: 2009 30th IEEE Symposium on Security and Privacy. pp. 375–390. IEEE (2009)
31. Stock, B., Pellegrino, G., Li, F., Backes, M., Rossow, C.: Didn't You Hear Me? - Towards More Successful Web Vulnerability Notifications. In: Proceedings of the 2018 Network and Distributed System Security (NDSS) Symposium (2018)
32. Stock, B., Pellegrino, G., Rossow, C., Johns, M., Backes, M.: Hey, you have a problem: On the feasibility of large-scale web vulnerability notification. In: 25th {USENIX} Security Symposium ({USENIX} Security 16). pp. 1015–1032 (2016)
33. Yoo, C., Kang, B.T., Kim, H.K.: Case study of the vulnerability of otp implemented in internet banking systems of south korea. Multimedia Tools and Applications **74**(10), 3289–3303 (2015)
34. Zhang, Y., Gao, H., Pei, G., Luo, S., Chang, G., Cheng, N.: A survey of research on captcha designing and breaking techniques. In: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 75–84 (2019). https://doi.org/10.1109/TrustCom/BigDataSE.2019.00020