IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 23 November 2023, accepted 12 December 2023, date of publication 15 December 2023,
date of current version 20 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3343411

RESEARCH ARTICLE

An Empirical Analysis of Incorrect Account Remediation in the Case of Broken Authentication

JEONGHO LEE[1], HYOUNG-KEE CHOI[2], JIN HEE YOON[3], AND SEONGJUNE KIM[3]
[1]Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, South Korea
[2]College of Software, Sungkyunkwan University, Suwon 16419, South Korea
[3]Department of Computer Science and Engineering, Sungkyunkwan University, Suwon 16419, South Korea

Corresponding author: Hyoung-Kee Choi (meosery@skku.edu)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government through Ministry of Science and ICT (MSIT) (Artificial Intelligence (AI) Platform to Fully Adapt and Reflect Privacy-Policy Changes) under Grant 2022-0-00688.

ABSTRACT One of the most critical vulnerabilities in authentication, commonly referred to as ''broken authentication,'' poses a harmful threat, leading to the compromise of user credentials and the unauthorized hijacking of sessions. Addressing these security breaches is imperative, necessitating effective remediation mechanisms. Our primary objective is to assess and enhance the security posture of remediation mechanisms by addressing the vulnerabilities associated with broken authentication. Our investigation reveals deficiencies in the implementation of the three prevailing remediation mechanisms across popular Service Providers (SPs), rendering manual remediation attempts futile. We demonstrate our claim by measuring post-compromise security preparedness across over 350 popular websites and applications. During the measurement, SPs were divided into three groups to compare the correctness of the remediation mechanisms across groups. Based on the measurement and evaluation results, we analyzed the root cause of such incorrectness and discussed possible mitigations and practical recommendations to solve the remedial problems. The scope of this study ranges from compromise to the immediate consequences of countermeasures. Hence, discussions of the causes of broken authentication and descriptions of attacks for breaking authentication are beyond the scope of this study. Detailed case studies of four popular SPs are included to discuss their unique reactive prevention behaviors. Observations and their meaningful results challenge us to render remediation mechanisms opaque and difficult to audit, which contributes to underestimating the security threats of ineffective revocations.

INDEX TERMS Cyber security, internet security, trust management, broken authentication, account remediation, security measurement.

I. INTRODUCTION

Security needs to catch up during the increased modernization of web services. The user account is a valuable target for attackers. One vulnerability that allows attackers to gain access is ''broken authentication,'' a year-long entry in the Open Web Application Security Project (OWASP) [1]. Authentication is ''broken'' when bad actors compromise user accounts, stealing their credentials and hijacking their sessions. If attackers successfully bypass authentication, they can misuse the victim server's privileges and resources.

Broken authentication is becoming increasingly common with many high-profile incidents reported in recent years. This could be caused by server implementation flaw [2], [3] and the user's failure to properly manage their credentials [4], [5]. While servers support various authentication methods to improve user experience, the threats to accounts through these methods become diversified. Federated identity is one of the methods that makes server management more complex and opens up opportunities for attackers to launch new types of attacks.

The associate editor coordinating the review of this manuscript and approving it for publication was Zijian Zhang.

In today's rapidly evolving security landscape, organizations strive to proactively anticipate emerging threats before a security breach takes place and reactively mitigate risks once the compromise is remedied. Numerous advanced studies have proactively conducted security assessments [6], [7], [8], [9]. By comparison, we hope to better understand the safety implications of post-compromise countermeasures and reactive prevention.

In "post-compromise" security, service providers (SPs) are tasked with helping users reclaim access to breached accounts. "Remediation" involves undoing unauthorized access by bad actors and restoring user control. However, the correct implementation of this is challenging. The complex of account information adds to the difficulty. This complexity makes both credential management and session inventory in SP difficult. To extend their dwell time, bad actors actively launch new attacks in the time interval between compromise and remediation. As a result, the victim's account remains uncontrolled. Their sessions become unreachable, even after remediation is supposedly complete.

In our study, we claim that post-compromise security implementations are flawed in popular SPs. There are still sessions left uncured, even if all remediation efforts have been made. This vulnerability arises from the failure of SPs to thoroughly implement remediation mechanisms and ineffective manual remediation efforts. The goal of this study was to take significant steps toward understanding abnormal remediation mechanisms. We explored the session management mechanisms available to users who desperately needed them after their accounts were compromised. We also comprehensively measured the remediation mechanisms implemented by popular SPs on various platforms and use cases.

Our study aimed to ask the following three research questions by measuring the degree of readiness to ensure post-compromise security in the wild.

Q1. What are the common remediation mechanisms in SPs and how do the SPs implement them?

We examined common remediation mechanisms across popular websites and applications and verified their implementation. Our measurement is comprehensive and includes different platforms and authentication scenarios. Next, we examined the session and account management options available to users after their accounts had been compromised. Three mechanisms were the most common for 200 websites, 100 mobile applications, and 50 games.

Q2. How well do SPs handle remediation mechanisms and what difficulties do they encounter?

We evaluated each SP's remediation mechanism for all SPs and validated their implementation. We present case studies of vulnerable services to understand the challenges and potential service threats. Case studies include Microsoft, Quora, PayPal, and TikTok. We addressed how the mishandled remediation mechanisms could threaten the victims' privacy and security.

Q3. What causes SPs to misuse the remediation mechanism, and how can SPs correct the failures?

Based on the measurement and evaluation results, we analyzed the root cause of such incorrectness and discussed possible mitigations and practical recommendations to solve the remedial problems. The scope of this study ranges from compromise to the immediate consequences of countermeasures. Hence, discussions of the causes of broken authentication and descriptions of attacks for breaking authentication are beyond the scope of this study.

Our key contributions are summarized as follows:

- We defined the state changes of a user's authenticated session based on user and attacker actions. Based on this, we outlined potential issues that can arise from the remaining session when remediation fails.
- We encompassed a thorough survey of the most employed remediation mechanisms across various popular websites and applications. We rigorously validated the implementation of these mechanisms and further explored the session and account management options available to users.
- We conducted a comprehensive evaluation of the remediation mechanisms used by SPs. We investigated how many SPs are suffering from the problems we present, and the trends in how these issues arise.
- We conducted a case study which revealed how improperly handled remediation methods could pose significant risks to user's privacy and security.
- We conducted an in-depth analysis to pinpoint the root causes of the observed inaccuracies in remediation methods. Based on these findings, we have proposed practical recommendations to resolve issues related to the remediation process.

The remainder of this paper is organized as follows. Section II defines our service model, dividing it into the service architecture and security model. In Section III, we introduce the remediation mechanism, that we focus on in the study, and define its security requirements. Section IV presents the criteria for selecting a set to measure the safety of remediation implementations and explains our experimental method. Section V presents our experimental results for top websites, applications, and games. Section VI introduces a case study of popular services. In Section VII, we delve into several technical challenges encountered during our research and discuss the resulting limitations. We also engage in a detailed discussion on how our study distinguishes itself from other research, highlighting our unique contributions. In Section VIII, we review the existing literature in the field. This includes examining prior studies related to our research focus, where we compare and contrast our methodology, findings, and the implications of our work with those of previous research. Finally, Section IX. concludes this paper.

II. SERVICE MODEL

We define a security model based on the motivation of our research to delineate the security issues at hand. In this

section, we define the architecture of the service discussed in the paper and characterize the attacker aiming to compromise it. From this, we present a novel problem that our work addresses.

A. SERVICE ARCHITECTURE

There are four entities in the service architecture. The user subscribes to the service by setting up an account in SP. A user agent is a user interface application that interacts with a remote server on behalf of the user. The server responds to the user by processing requests from user agents and records the data on the backend to serve subscribers. Servers can be classified into two groups depending on their role. The identity server authenticates users and authorizes user agents. A resource server stores and manages the resources and data that are accessible to authorized user agents.

SP provides resources and services to authenticated users by authorizing them. The user initiates the authentication process by providing a username and password to the user agent. The user agent redirects the user's credentials to the identity server. Upon successful authentication, the SP creates an authenticated session for the user. On subsequent requests to the server, the user agent is implicitly authenticated for the services by including a valid token or cookie with the request [5]. This session-based authentication is only as effective as the validity period of the access token. This improves the user experience by eliminating the need for repeated authentication procedures when using the service.

After authentication, the user requests authorization for resources on the server from the user agent. SP authorizes user agents according to their subscriptions and premiums. ''Authorization'' is the process of granting permission to access specific resources and perform actions on those services. The user agent sends a request to the identity server, which returns an access token to the user agent as a sign of approval. The user agent then forwards this access token to the resource server.

Single Sign-On (SSO) allows users to access multiple services and resources using a single login credential. The user has an account with an identity provider (IdP), which is a trusted source for SPs. The user logs in to the IdP to acquire an access token to authorize the SP. When users attempt to access services in the SP, they send an access token to the SP. The SP then forwards the token to the IdP for verification. Upon receiving a positive response, the SP logs the user in and gains access to the user's information stored in the IdP, but only within the specified scope authorization of permissions. The SSO relies on a trusted third party to arrange authentication between the user and the application.

SP then remembers the user by creating a session. A session is a series of contiguous interactions between a user and an SP within a given time frame. SP uses the session to remember user preferences, track user behavior on the website, and enable personalization of advertising. There are two types of sessions: stateful and stateless, where saving the user's state determines the type of session [10].

SP issues a token to the user agent. This token, which contains session information, is saved in the user agent. This is stateless because the server does not maintain the session state [11]. Instead, the server restores the state of the session by parsing the information contained in the token when it is returned on the next visit. The token lives for a lifetime, which means it is persistent until expiration. This is called a persistent token.

In contrast, a stateful session stores session information directly from the server's database. Session information is secure because it is centralized. Stateful session information is composed of attributes that define a specific session. These attributes are marked with a session identity (ID). The session ID is chosen by the SP and provided to the user agent in the form of a cookie. The session ID contains no authentication information but is a random string of characters generated by the SP.

B. SECURITY MODEL

1) ATTACKER'S CAPABILITIES

The adversary's goal is to leak personal information stored on the server and gain unauthorized access to resources to which the adversary does not have permission to connect. To accomplish this goal, the adversary must break the authentication to take over the accounts. Our research focuses on security during the post-compromise period; thus, we assume a scenario where the attacker has already taken over the account. Our security model does not limit the methods an attacker might use to do this. Exploiting broken authentication, the attack typically takes advantage of unattended credentials and login sessions through phishing [12], server exploitations [13], social engineering [14], device relocations [15], and other techniques.

Malicious actors rely on diverse techniques to steal credentials, guess them, and trick users into revealing them. Typically, phishing scams send victims emails pretending to come from a trusted source and then trick users into sharing their credentials. Credential stuffing involves injecting stolen credentials into websites to test credentials found in a list of unencrypted emails and passwords. This tactic often works because people frequently use the same password across applications [16]. Password spraying is similar to credential stuffing, but instead of working through a list of stolen passwords, it uses a set of weak or shared passwords like ''123456'' and ''password'' to break into a user's account.

The most common instance of session hijacking occurs when a user forgets to log out of the application and walks away from the device or browser. Such an oversight could allow another individual to exploit the established session. A session fixation attack can prevail when the server does not change the session ID, instead of giving the user the same ID before and after authentication. In this attack, the bad actor predetermines the session ID and sends the victim a link containing the predetermined session ID. If the SP has other vulnerabilities, such as cross-site scripting (XSS),

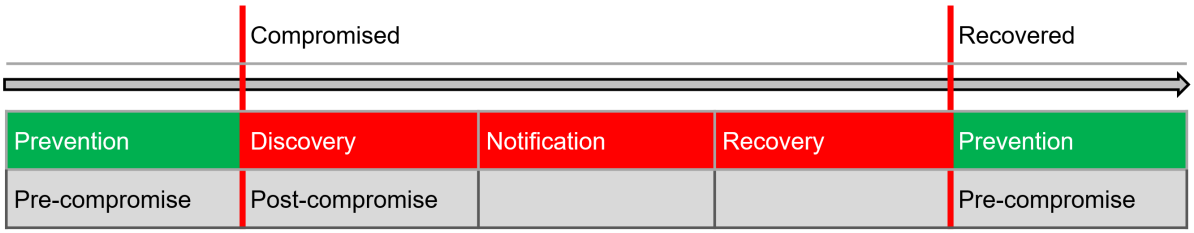

FIGURE 1. A life cycle of the compromised session.

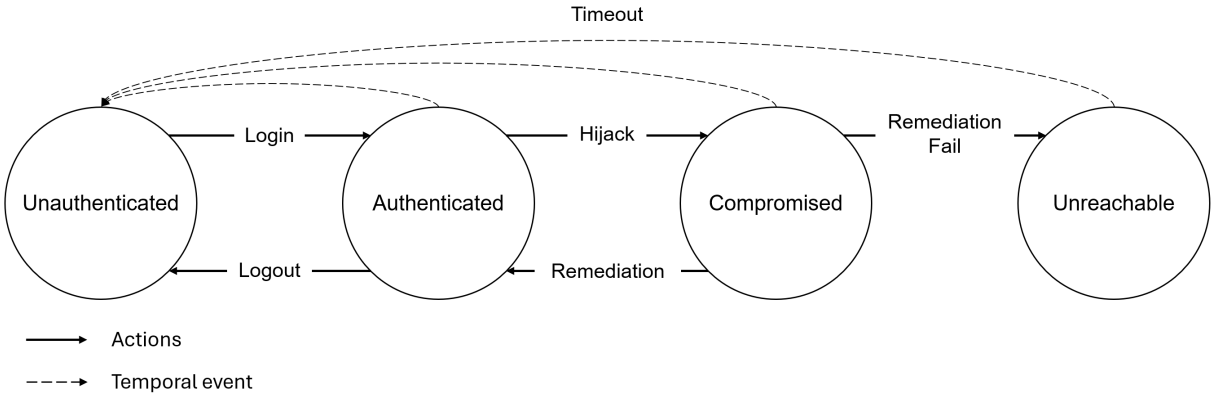

FIGURE 2. Changes in session state based on user actions and events.

an adversary can inject malicious client-side scripts into the webpage. The attack occurs when the victim visits the SP that executes malicious code in the browser and delivers authentication cookies to the adversary.

2) REMEDIATION PROCESS

In our security model, the user attempts remediation after some time has elapsed and realizes that an attacker has compromised their account. The methods for conducting remediation depend on the nature of the attack and may involve regaining control of credentials or sessions. Credentials, being user-configured values, are intuitively understood and can be updated in a direct and comprehensible process.

Session management determines and manages session attributes, such as how long a session lasts before users log out or how the server issues session IDs. Effective session management is difficult because SP creates several sessions of different types. Users can have different authorization levels for different parts of the service. As the number of users increases, it becomes almost impossible to track these sessions in a timely and accurate manner. Most SPs have historically focused on strengthening authentication and preventing compromise [17], [18], [19]. Despite these detection and prevention efforts, account and session takeovers did eventually occur in the wild.

In the epoch of compromise, the session lives in either the pre- or post-compromised period, as shown in Fig. 1. During the pre-compromised period, service participants strengthened server security to prevent authentication breaches by proactively identifying the threat. Nevertheless, a compromise may occur, and a post-compromise period should begin. This period comprises three phases: compromise detection, user notification, and account recovery. After recovery, the pre-compromised period begins again. During this period, the SP continues to seek to protect any further compromises.

During the detection phase, the user observes suspicious activity in the account and service that indicates a possible compromise. The logging capability and intelligent logic in SP can detect unauthorized and suspicious activities to notify users of these incidents. Common incidents include sudden password and email address changes, unusual attempts to access forbidden resources, and illegal connections to third-party accounts.

This study did not include the discovery and notification phases because many versatile tools and methods were already available. We assumed that the SP detected broken sessions using existing, well-known methods [20], [21] and timely alerted users about the broken authentication [22], [23]. Our study mainly focuses on the account recovery phase, during which the user uses the mechanisms offered by the SP to revoke the broken authentication. We examined how popular SPs handle account recovery and found slight differences in each process. We surveyed various remediation mechanisms to recover stolen credentials, session IDs, tokens, and cookies and discussed their strengths and weaknesses.

3) PROBLEM OF REMEDIATION FAILURE

Sessions maintain various states throughout their lifecycle. We define the session states when an attacker compromises a session and subsequently recovers through remediation.

We define our novel problem within these states. Fig. 2 illustrates how the session state changes based on user actions. The user maintains an authenticated session by logging into their account. If an attacker hijacks an authenticated session, the user attempts to recover the authenticated session through remediation. If the remediation fails, a session that the user cannot control is created. The name *"unreachable session"* comes from the fact that the session is left unattended and uncured. Each session times out and returns to an unauthenticated state after its unique expiration time.

An unreachable session is mainly caused by two reasons: 1) inappropriate implementation of remediation mechanisms by the SPs and 2) no options available to the user to terminate the illegal access occupied by the attacker. The victim cannot update session-related attributes stored in the server database because these attributes are generated and configured by the SP. For instance, a device ID is an attribute used by an SP to allow requests from a device. Since the device ID is not a user-configurable attribute, the update can only be performed if the SP explicitly offers associated functions to the user. Furthermore, the user is limited in exercising session remediation because sessions are created, maintained, and terminated solely by the SP. The SP must identify illegitimate sessions among the many existing benign sessions and offer users the option to selectively close sessions.

III. REMEDIATION MECHANISMS

The user initiates the recovery process using a single or a combination of remediation mechanisms provided by the SP. Our interest in the remediation mechanism is aligned with the adversarial model that specifies nefarious behavior in terms of credential theft, session hijacking, and cookie intercepting [24].

Our remediation mechanism did not include account deletion and recovery assisted by customer service. This is because the remediation we define aims to restore compromised sessions to a pre-compromise period. At the same time, deletion causes the session to change for an indefinite period to which even the user has no access. Human actions are involved in customer service. This type of account recovery is too unpredictable for modeling.

A. CORRECTNESS OF REMEDIATION MECHANISMS

Deploying remediation mechanisms in the natural system creates some challenges. This may be insufficient for any single mechanism could resolve compromises completely. A combination of remediation mechanisms is necessary to rectify these security issues.

1) CREDENTIAL UPDATE

The remediation mechanism for credential theft is an immediate update of secrets to regain access to the account. The general procedure is to first identify which user we want to update and select which secrets to update. The identification procedure allows the victim to log into the server. Sometimes, authentication fails at this stage as a pre-planned attacker changes the password to their secret value. In this case, password reset is the only viable option.

Password reset is a common mechanism in online services that allows users who have forgotten their password or triggered an intruder lock to authenticate with secondary validation. Of course, if the attacker can, a smart attacker will also change the secrets used when resetting the password. In this case, the victim cannot recover their account after the security compromise; therefore, the session becomes unreachable to the user.

When the user resets the password, previous sessions must be ignored and destroyed. SP reminds users to review all existing logged-in sessions. The best approach is to present the user with a list of all ongoing sessions, giving them the option to revoke either all or selected sessions. The SP should allow the user to revoke all extant sessions.

2) SESSION MANAGEMENT

To remediate session hijacking, the SP terminates stateful and stateless sessions occupied by the attacker. It closes stateful sessions by deleting session information from the server database. SP requires special care when closing stateless sessions. The SP identifies all persistent tokens that have been issued for this session and invalidates them individually. Tracking them is another daunting task, as several users are simultaneously connected to popular services.

The SP is not always sure whether the user has quit the service unless the passive user actively and explicitly logs out of the service. Therefore, the SP must remove sessions that have been idle for a while. This session termination can be accomplished by idle timeout when there is no session inactivity [25]. With an active session, the SP has no way to determine whether a legitimate user is using the session or whether a bad actor has stolen it. Hence, the SP must implement an absolute timeout that restricts the overall session duration. Absolute timeout is critical when a bad actor remains valid by sending periodic requests.

These two timeout values must be set according to the purpose and use of the service and the balance between security and usability. However, the SP is set too low for idle timeout and too high for absolute timeout.

After the privilege level changes within the associated session, the SP must regenerate the session ID. Session ID regeneration is the first requirement of session management. The most common scenario where session ID regeneration is mandatory is during the authentication process, when the user's privilege level changes from an anonymous state to an authenticated state. Another common scenario to consider is a user changing a password.

3) SIGN-OUT EVERYWHERE

Some SPs unconditionally close all ongoing sessions associated with the victim's account. This remediation mechanism is called "Sign-out everywhere." This is required by the user when logging out of the service and by the SP, partly

to prepare the user for password reset and partly when recovering a compromised account. Whether to close all sessions or part of a session depends on whether the user is aware of the compromised sessions.

This is a repetition of individual session management for all sessions established for a given user. One implementation of the SE mechanism could be to remember all transient cookies issued for individual sessions. However, this contradicts the fact that there is a persistent token for a stateless session and remembering the token is equivalent to maintaining the state.

The SP identifies the requesting user and invalidates the user's cookies and tokens on demand. User identification depends on the SP's interpretation of the user. The SP provides cookies and tokens to the user, device, IP address, browser, application, and a combination of these. The multitude of user interpretations makes the overall implementation quite complex and sometimes renders the SP unable to associate the session with its owner. This failure causes the SP to disregard certain cookies and tokens when finalizing all this information, creating a risk of an unreachable session.

B. SECURITY REQUIREMENTS OF REMEDIATION MECHANISMS

The definition of what constitutes a security requirement must be carefully chosen depending on the purpose of the security model. This brings design choices with significant implications. These decisions were made after considering both practical implementation and security issues.

1) OWNERSHIP

The SP does not design a token securely, so the SP cannot validate ownership of the returning token. A more effective design would involve embedding user information into the token. However, many SPs still prefer stateful session management and use a random string as the session ID value. This creates management difficulties in associating a compromised account with sessions derived from that user. Consequently, the SP may fail to terminate all sessions and eliminate authentication information spread across outstanding cookies.

2) AVAILABILITY

The SP should be able to invalidate any type of session. Furthermore, the SP should explicitly offer these functions to the user in the part of the service that enables preventive termination of selected sessions. However, all of these functions do not have to be explicit, as some functions must be performed to implicitly perform other functions to complete account recovery. These implicit functions are not known to the user or used by him when needed.

3) CORRECTNESS

The remediation mechanism should be able to terminate the victim's entire authenticated sessions. In addition, any new authenticated sessions derived from a compromised session should be discarded as part of the remediation mechanism. Once an attacker has compromised an account, they can set up new authenticated sessions to extend the dwell time on the compromised account. For the correctness of account recovery, both the old and new authentication information created by the attacker must be invalidated.

One suggested remediation mechanism for the situation could be to include all authenticated sessions associated with the victim in the invalidation, provided that the attacker has impersonated the victim. This is not an easy task for the following reasons. An attacker may intentionally change user identification to avoid censorship. Some SPs would not implement the suggested option due to user experience questions. Some SPs cannot implement the suggested option because they assign multiple random identifications for a single account. Some authenticated sessions are critical for a service so that it cannot be deleted in any case.

4) FEEDBACK

The SP must notify the user to let them know that the recovery is complete. If there is no such notification, the user does not know the result and usually assumes that the account recovery was successful. If the remediation failed, the naïve user would revert to normal services without knowing it. A bad actor can extend their dwell time on a compromised account, making these unreachable sessions more difficult to remediate.

IV. MEASUREMENTS

We measured the number of unreachable sessions following individual remediation mechanisms offered by many SPs. Our measurement of the mishandled remediation mechanism was comprehensive because we included multiple user agents and various SPs and considered different authentication scenarios. This experiment offers insights into the study of discrepancies in the implementation of remediation mechanisms in SP.

A. DEMOGRAPHICS

The SP provides the service to the user who can access the services with a web browser, a native application or both. If an SP ran multiple services and managed separate accounts for each service, we considered them as two different services. We counted one service if they shared one account. For instance, the Meta company is an SP that has two services, Facebook and Instagram, and a user can have two separate accounts for these two services. In our measurements, we treated them as two services.

To measure the prevalence of diversity that can lead to hijacking attacks, we analyzed some of the most popular SPs whose services are based on traditional browser-interacting websites. We selected the top 200 websites in the *Alexa* global ranking [26]. We measured 105 websites after excluding 95 websites for reasons.

Twelve websites are irrelevant to the adversarial model as they do not authenticate the user or maintain an authenticated session state. When a user registers an account on a website,

TABLE 1. Reasons and number of SPs in different service platforms were excluded from the experiment.

Reasons	# of Top 200 Websites (%)	# of Top 100 Applications (%)	# of Top 50 Games (%)
Irrelevant model	12 (12.6)	9 (26.5)	0
Tall order	32 (33.7)	16 (47.1)	0
Sibling account	16 (16.8)	4 (11.8)	14 (37.8)
SSO only	0	2 (5.9)	21 (56.8)
Harmful	8 (8.4)	1 (2.9)	0
Buggy	27 (28.4)	2 (5.9)	2 (5.4)
Total	95	34	37

32 websites challenge the user with a domestic phone number and a national identification number. Consequently, we were unable to create an account on these websites. Sixteen websites are owned by the same SPs and share accounts with sibling websites. We counted these sibling websites as one. We also excluded eight harmful websites as they contained scams and adult content. We could not create an account on 27 buggy websites due to unknown errors. One example is that the website did not respond promptly, so we were unable to complete account registration.

With the increasing use of mobile devices, mobile services have become an integral part of our daily lives and a significant revenue generator for companies. The security of mobile services has different requirements, and critical features are required in mobile application development. Motivated by the same reasons as the website, we would like to understand how secure SPs in mobile services deal with remediation mechanisms in the face of compromises. Furthermore, it is interesting to observe the differences and discrepancies between traditional web services and mobile services in handling the three remediation mechanisms.

We collected the top 100 applications from the Google Play Store [27]. In the interest of unique behavior in the game category, we have collected the top 50 game applications in the Google Play Store Revenue. Thirty-four on the mobile service and 37 on the game service were not selected because of the same conditions as the web services. Table 1 summarizes the six reasons to exclude from the experiment and the corresponding number of SPs on the web, mobile, and game platforms. There are 44 SPs in each of the top two lists, and 96 SPs are uniquely popular in each top list: 61 on the website, 23 in the mobile application, and 12 in the game application. A total of 140 SPs were uniquely identified after excluding duplicates.

The web browser is a typical and popular user agent. Alternatives are applications dedicated to SP running on desktops and mobile devices. There is no binary answer to the question of which user agent is more secure and safer than others. Some SPs are vigorous about the login process and password recovery, while others place more importance on convenience. A natural question is how user agents impact SP security during account recovery. This paper presents individual SP's security measures on different user agents. Some SP providers offer services via both browser and application. In this case, we present SP security measures applied to both the browser and the application.

B. METHODOLOGY

Our experiment hinges on exploring and observing user authentication and authorization options to establish as many authenticated sessions as possible. In our experiments, we simulated the user and the attacker on different devices running on virtual machines.

In the first step, a user creates an account for each SP in all SPs by registering a login ID and password. The user then logs into the SP for the first time to create an authenticated session. In this study, this is called the first authentication session. The user opens many authenticated sessions derived from the first authentication session in browsers and applications. The user also establishes authenticated sessions on different devices. In this way, the SP associates each device with an independently authenticated session, allowing the user to bypass the explicit login process on these devices.

Second, we manually surveyed all websites to identify the three types of remediation mechanisms discussed in the security model. Finally, we triggered each SP countermeasure multiple times for all SPs.

We observed whether any unreachable sessions may have remained after the remediation mechanism was completed. After a while, we counted the remaining sessions to extrapolate incorrect implementations of remediation mechanisms and diagnose symptoms of incorrectness. Unless SP specifies otherwise, the waiting period defaults to one hour.

C. VALIDITY

In assessing the effectiveness of remediation mechanisms following account compromises, it is critical to consider the validity of our experimental approach and the reliability of our results. We employed a comprehensive approach to measure the remediation mechanisms. This involved simulating both user and attacker scenarios, creating multiple authenticated sessions, and rigorously testing the remediation mechanisms provided by the SPs. The depth of our experimental design enhances the robustness of our findings.

Our study, carried out manually, ensured that each service was measured across all its platforms concurrently. This comprehensive approach bolsters the reliability of our results, particularly in assessing the consistency and effectiveness of security remediation mechanisms across different platforms. By simultaneously evaluating all platforms of a service, we minimized the variability that could arise from testing platforms in isolation.

D. ETHICAL CONSIDERATIONS

One of our main priorities during the experiments was to ensure that the impact was limited to the user accounts that were created. This has been prioritized to minimize any potential negative effects on other users and/or systems. Another important consideration was not to use any

TABLE 2. Thirty-five services in the Alexa top 200 that implement remediation mechanisms incorrectly.

No.	Alexa Rank	Domain Name	Types	Is incorrect			Platform	Is unreachable?	Unreachable Platform	
				CC	SE	SM			Browser	Native App
1	4	facebook.com	Type 4	✓	✓	✗	Multiple	✗	-	-
2	12	reddit.com	Type 3	✗	-	✓	Multiple	✗	-	-
3	34	tiktok.com	Type 3	✓	-	✓	Multiple	✓	✓	✓
4	40	aliexpress.com	Type 1	✓	-	-	Web	✓	✗	✓
5	59	quora.com	Type 4	✓	✓	-	Web	✓	✓	✗
6	81	wordpress.com	Type 3	✓	-	✗	Web	✗	-	-
7	83	nytimes.com	Type 1	✓	-	-	Web	✓	✓	✓
8	84	bbc.com	Type 1	✓	-	-	Web	✓	✓	✗
9	86	edition.cnn.com	Type 1	✓	-	-	Web	✓	✓	✓
10	89	indeed.com	Type 1	✓	-	-	Web	✓	✓	✓
11	95	aws.amazon.com	Type 1	✓	-	-	Multiple	✓	✓	✓
12	96	force.com	Type 1	✓	-	-	App	✓	✗	✓
13	98	coinmarketcap.com	Type 1	✓	-	-	Web	✓	✓	✗
14	107	espn.com	Type 2	✓	✓	-	Multiple	✓	✓	✗
15	114	medium.com	Type 2	✓	✗	-	Web	✗	-	-
16	118	kompas.com	Type 1	✓	-	-	App	✓	✓	✓
17	127	cctv.com	Type 1	✓	-	-	Web	✓	✓	-
18	130	rakuten.co.jp	Type 1	✓	-	-	App	✓	✓	✗
19	132	freepik.com	Type 3	✓	-	✓	App	✓	✓	✓
20	136	tistory.com	Type 3	✓	-	✓	Web	✓	✓	✗
21	144	booking.com	Type 2	✓	✗	-	Multiple	✗	-	-
22	145	kumparan.com	Type 1	✓	-	-	App	✓	✓	✓
23	151	patreon.com	Type 1	✓	-	-	Web	✓	✓	✓
24	154	nicovideo.jp	Type 1	✓	-	-	App	✓	✓	✓
25	158	archive.org	Type 1	✓	-	-	Web	✓	✓	-
26	159	shutterstock.com	Type 1	✓	-	-	Web	✓	✓	✓
27	169	aliyun.com	Type 1	✓	-	-	Web	✓	✓	-
28	170	yts.mx	Type 1	✓	-	-	Web	✓	✓	-
29	177	formula1.com	Type 1	✓	-	-	Multiple	✓	✓	✓
30	178	dropbox.com	Type 4	✓	✗	✗	Multiple	✗	-	-
31	181	speedtest.net	Type 1	✓	-	-	App	✓	✓	✗
32	182	ikea.com	Type 1	✓	-	-	App	✓	✓	✓
33	190	namu.wiki	Type 1	✓	-	-	Web	✓	✓	✓
34	195	samsung.com	Type 2	✓	✓	-	App	✓	✓	✗
35	196	crunchyroll.com	Type 3	✓	-	✓	Multiple	✓	✓	✗

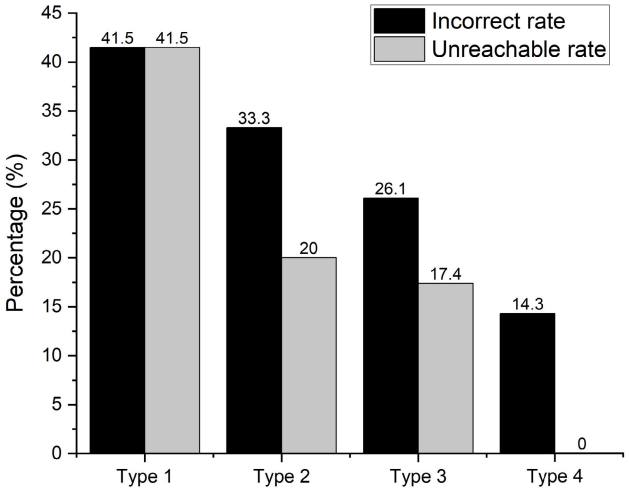

FIGURE 3. Comparison of incorrect and unreachable rates based on remediation type.

automated tools that could send a large number of requests. We could avoid negatively impacting site performance for other users by limiting the number of requests and performing manual experiments. Although we have not yet completed the research and prepared a report, we plan to report incidents to the related services in due time. This is to ensure that the vendor is aware of identifying the issues and taking appropriate steps to resolve the vulnerability. By reporting vulnerabilities, we help improve the overall security of the website for all users.

V. EXPERIMENTAL RESULTS

We aimed to observe differences and discrepancies in the implementation of corrective actions across three service platforms: a web platform, a mobile platform, and a platform for both services. We have found that the game on the mobile platform is particularly important in terms of statistics and measurements.

A. ACCORDING TO TOP ISTS

We repeatedly executed the three-step methodology for SPs in the *Alexa* top 200.

1) ALEXA TOP 200

There were 53 SPs capable of only a single remediation mechanism. The single mechanism is unanimously Credential Change (**Type 1**). Thirty-eight SPs supported a combination of the two remediation mechanisms. Fifteen SPs were enabled to Credential Change and Sign-out Everywhere (**Type 2**), and 23 SPs were enabled to Credential Change and Session Management (**Type 3**). Fourteen SPs supported all remediation mechanisms (**Type 4**). Table 3 shows the statistics of the SP's security measures for the remediation mechanism.

We found that 35 websites had at least one incorrect implementation. Table 2 presents the results for all of the tested websites that provided incorrect remediation

TABLE 3. Statistics of the SP's security measures on the remediation mechanism in Alexa top 200 websites.

Types	Remediation Mechanism			EN*	IN†	UN‡
	CC	SE	SM	# (%)	# (%)	# (%)
Type 1	✓	-	-	53 (50.5)	22 (41.5)	22 (41.5)
Type 2	✓	✓	-	15 (14.3)	5 (33.3)	3 (20.0)
Type 3	✓	-	✓	23 (21.9)	6 (26.1)	4 (17.4)
Type 4	✓	✓	✓	14 (13.3)	2 (14.3)	0
EN (%)	105	29 (27.6)	37 (35.2)	105	35 (33.3)	29 (27.6)
IN (%)	34 (32.4)	4 (13.8)	5 (13.5)		-	

*: Enable, †: incorrect, ‡: unreachable

mechanisms. However, only 29 of the 35 websites ultimately resulted in unreachable sessions. The remaining six SPs are, in fact, incorrect but did not generate any unreachable sessions at the end. This is because SPs can combine two or three remediation mechanisms. The following correct mechanism terminates sessions that survive the first buggy mechanism.

Significantly, 14 websites supporting all three remediation mechanisms did not leave unreachable sessions. Fig. 3 shows the comparison of incorrect and unreachable rates depending on the remediation type. Our observations suggest that if a website had to choose only two mechanisms, it should choose **Type 3** over **Type 2**. Besides, there are better choices than relying on a single remediation mechanism.

Another way to appreciate these statistics is that 105, 29, and 37 SPs, respectively, support the remediation mechanisms of Credential Change (CC), Sign-out Everywhere (SE), and Session Management (SM) as shown in the last two rows of Table 3. Approximately 32 percent of websites implemented the CC method incorrectly. About 14 percent of websites mishandled SE and SM methods, respectively.

2) THROUGH SERVICE PLATFORMS

We divided the 140 SPs into three groups based on the platform on which services are delivered from the SP to the user: a web platform, an application platform, and a platform using both. There were 14 web platforms, 13 application platforms, and 112 multi-platforms.

We had to revise this platform definition to reflect actual service patterns. For instance, Uber belongs to a multi-platform SP as the user can manage their account on the web and mobile application. However, almost all car-sharing services are delivered on mobile devices. We could guess that a user often attempted to manage accounts in the mobile application. Hence, we decided to basically include Uber in the SP application platform. If the number of users on any platform exceeded 60 percent, the SP was designated to that platform. If neither platform meets this threshold, the SP is classified as multi-platform.

TABLE 4. Security measures on the remediation mechanism for the entire service based on the type of platform.

Types	Web (%)			App (%)			Multiple (%)		
	Enable	Incorrect	Unreachable	Enable	Incorrect	Unreachable	Enable	Incorrect	Unreachable
Type 0	0	0	0	9 (16.7)	9 (16.7)	9 (16.7)	0	0	0
Type 1	40 (66.7)	17 (28.3)	17 (28.3)	24 (44.4)	19 (35.2)	19 (35.2)	7 (26.9)	1 (3.8)	1 (3.8)
Type 2	7 (11.7)	1 (1.7)	0	5 (9.3)	2 (3.7)	2 (3.7)	5 (19.2)	2 (7.7)	1 (3.8)
Type 3	11 (18.3)	1 (1.7)	0	8 (14.8)	2 (3.7)	2 (3.7)	8 (30.8))	3 (11.5)	2 (7.7)
Type 4	2 (3.3)	0	0	8 (14.8)	1 (1.9)	1 (1.9)	6 (23.1)	2 (7.7)	0
Total (%)	60	19 (31.7)	17 (28.3)	54	33 (61.1)	33 (61.1)	26	8 (30.8)	4 (15.4)

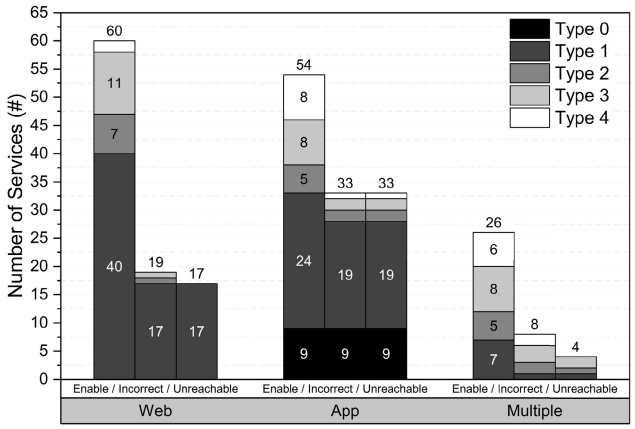

FIGURE 4. Comparing the security of remediation mechanisms based on the type of platform.

TABLE 5. Correctness of remediation mechanisms in mobile and game applications.

Types	App (%)			Game (%)		
	EN*	IN†	UN‡	EN	IN	UN
Type 0	5 (11.9)	5 (21.7)	5 (21.7)	4 (33.3)	4 (40.0)	4 (40.0)
Type 1	19 (45.2)	14 (60.9)	14 (60.9)	5 (42.6)	5 (50.0)	5 (50.0)
Type 2	5 (11.9)	2 (8.7)	2 (8.7)	2 (16.7)	0	0
Type 3	6 (14.3)	2 (8.7)	2 (8.7)	2 (16.7)	0	0
Type 4	7 (16.7)	0	0	1 (8.3)	1 (10.0)	1 (10.0)
Total	42	23 (54.8)	23 (54.8)	12	10 (83.3)	10 (83.3)

*: Enable, †: incorrect, ‡: unreachable

We leveraged the statistics for user populations from the website *data.ai*. One interesting example is that TikTok belongs to the multi-platform according to our revised rule. Statistics show that many user populations use both browsers and applications. Neither the browser nor the application itself made a difference of 60 percent. Twelve SPs are not listed on the referencing site; therefore, we had to manually inspect the 12 SPs to determine the platform type. According to the revision, there are 60 web platforms, 54 application platforms, and 26 multi-platforms.

Table 4 shows the experimental results for the four remediation types and one addition. If an SP does not support any remediation mechanism, it is classified as remediation **Type 0**. There are only nine **Type 0** SPs on the application platform. Some of these SPs only accept a single session per account at a time, so the account is only used to check a single active session. Some offer trivial services like synchronizing a history across multiple devices and activity-based rewards. Some SPs bind an account to a registered device, and this binding is immutable and transparent to the user. These nine SPs manage the session simply by identifying the user and do not implement any account recovery mechanisms. This inevitably generated unreachable sessions.

Fig. 4 shows the distribution of incorrect and unreachable services based on the remediation mechanism on the three platforms. The application platform is the worst at dealing with unreachable sessions compared to the web platform and the multi-platform. There is one SP in the application platform that is unable to prevent an unreachable session even if it supports all three mechanisms.

SPs across multi-platforms support rich remediation mechanisms in both qualitative and quantitative terms. Seventy-three percent of SPs had more than two remediation mechanisms enabled across the multi-platform, while around 30 percent were enabled in the web and application platforms. Only 15 percent of SPs left sessions unreachable compared to 28 percent and 61 percent for web and application platforms, respectively.

However, individual remediation mechanisms in the multi-platform perform similarly to those on web platforms. Eight SPs in the multi-platform mishandle had at least one remediation mechanism. This number drops to four SPs in generating an unreachable session, as multiple remediation mechanisms treat potential unreachable sessions. In contrast, multiple mechanisms in the application platform did not significantly improve, as five incorrect mechanisms resulted in five unreachable sessions.

The overall session management of native applications is worse than the other two platforms. Native applications are built for a specific platform, such as iOS and Android. They have access to system resources, such as GPS and camera

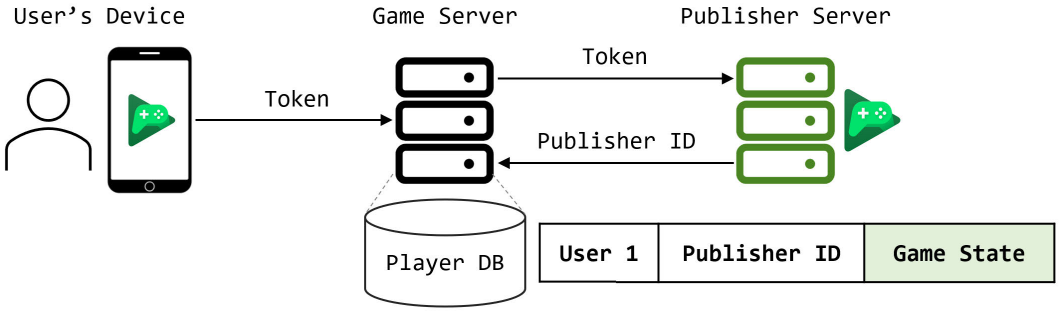

FIGURE 5. Integration process of authentication information between the game server and the user's publisher account.

functions. An application platform is the best choice if SP needs special functions and specific purposes in the device. By creating a native application, developers can implement session management from scratch or build it based on proven frameworks.

Native applications call framework application programming interfaces (API) to retrieve data and business logic stored on the back-end server. A serious problem occurs because calling an API requires using tokens, and several APIs need to deal with it [28]. Many mobile applications do not automatically log users out. The reasons may vary, such as inconvenience to customers or decisions made when implementing stateless authentication. Although the application should implement a logout functionality according to best practices, destroying all locally stored persistent tokens is critical to their performance.

Web applications are significantly more secure than native ones since they run on a browser, and popular browsers offer good protection and privacy. Web applications commonly use stateful authentication with transient session IDs. In contrast, stateless approaches with persistent tokens are becoming popular in many applications [29]. The stateless method improves scalability and developer's flexibility by allowing authentication to be separated from application logic. All unreachable sessions on the web platform are due to SPs only supporting a single mechanism of Credential Change. SPs supporting multiple remediation mechanisms did not generate unreachable sessions.

B. MOBILE AGAINST GAME

Mobile games possess a distinct ecosystem among various mobile applications. Google Play categorizes apps into game and non-game applications. Among numerous unique aspects, we focused on the login process of mobile games and conducted our experiments on mobile game apps.

1) TRADITIONAL LOGIN IN GAME

Table 5 compares the correctness of remediation mechanisms between 42 mobile applications and 12 game applications. Game applications are less capable of recovering stolen accounts than mobile applications. Over 80 percent of SPs incorrectly implemented remediation mechanisms and left the same number of unreachable sessions. Four of the twelve games did not support any mechanism. Five games only supported the Credential Change mechanism and all five games incorrectly implemented the remediation mechanism. This is because a game account is nothing more than an identification; therefore, setting and updating a credential for an account is optional in the game.

Seventy-six percent of SPs were successfully launched using either one remediation mechanism, or none of them. Many game applications allow a single session per account for the lifetime of the service. Functions to revoke deprived sessions are no more critical than functions to distinguish sessions from the same account. This is because the bad actor cannot duplicate the session from the current session, and, more importantly, remediation is as simple as the user restarting the game application.

2) SSO LOGIN IN GAME

Mobile applications in the game category have adopted significantly different session management than applications in other categories. Of the 50 game applications we collected, 12 used traditional login and 21 used SSO login. Game applications are available in the application store, and their revenues also come from them. The application store, in order to share financial profits with SP, forces customers to use the same store account when registering accounts in the game application. This makes it easier for the application store to monitor downloads and purchases. For instance, Google Play Service users can download games using only a Google account. SSO is becoming the dominant login implementation in games.

As part of an SSO implementation, an application can transparently create an ephemeral guest account for a user and internally link the user's SSO account to the guest account. Linking an account is illustrated in Fig. 5, where User 1 is an ephemeral guest account and Publisher ID is the user's SSO account [30]. Linking information is saved in the game server database until it is unlinked or deleted. The user signs into the game application automatically after logging into SSO accounts. The game application is exempt from implementing many complex housekeeping functions in account management. The applicant may also link a user's

SSO account to each session instead of an ephemeral account. The game server can easily detect multiple sessions and avoid the overhead of frequent conversions between ephemeral and SSO accounts.

Disconnecting an account is a remediation mechanism in the game application. The mechanism can be considered as deleting an entry from the game server database. The remediation mechanism may fail if the session is linked to an SSO account and unlinking removes the ephemeral account from the database. In this case, an unreachable session may have occurred.

VI. CASE STUDY

This section describes how an unreachable session is generated and explains how these sessions can potentially cause other attacks. We do not address how traditional account hacking attacks are carried out and their impact.

A. HIDDEN SESSIONS

Some remediation mechanisms inadvertently fail to terminate sessions due to a session management oversight.

1) QUORA

Quora is a social question-and-answer website and online knowledge market. Users can collaborate by editing questions and commenting on other users' submitted answers. As of 2020, the website was visited by 300 million users per month. Quora SP supports the **Type 2** remediation mechanism. It is a web platform in our category.

a: IMPLEMENTATION

The server on Quora establishes a stateful session and maintains at least two pieces of information about the user: a browser ID for tracking and an account ID for authentication. The browser ID is issued to first-time visiting users, regardless of whether they are authenticated. Quora may assign multiple browser IDs to the same user if they visit its site from different devices and browsers. The account ID is created when the user logs into the website and deleted after the user logs out. In contrast, the browser ID, remains undeleted even if the user logs out. These two IDs are delivered to the user in the form of transient cookies.

b: ATTACK

An authenticated user can sign on to the website using an authentication ID or browser ID. A bad actor who has hijacked the browser IDs can log in to the website. Once the user logs out from the website, neither the user nor the lousy actor can log in. The user signs into the website on the next visit, and a new account ID is assigned for the current session. The old browser ID is reinstated for the current session if available. The lousy actor holding the old browser ID can steal a new session again. The problem becomes even more acute if bad actors can steal the browsing ID on a public computer. They should be able to compromise future sessions of other users initiated from this computer.

The browser ID used for authentication is the main cause of the incorrect remediation mechanism. The browser ID is one of the session attribute fields. The field is initialized to an unauthenticated status and automatically switches to an authenticated status after the user is authenticated. Then the user can sign in with any ID. The remediation mechanism does not delete the browser ID because we believe the Quora site needs the browser ID to track sessions.

2) PAYPAL

PayPal operates an online payment system that supports online money transfers for online vendors, auction sites, and other commercial users. PayPal supports the **Type 1** remediation mechanism and is assigned to the web platform.

The notification function is critical in the PayPal service as it confirms recent transactions with the user. The notification function is only available in the mobile application yet unavailable in the browser. The delivery of push notifications is handled by dedicated third-party companies. Once a user is authenticated, the PayPal SP uniquely registers the user's device with the notification server, and then the notification server establishes separate sessions for each device.

Assume an attacker steals a victim's credential and signs on the victim's PayPal account. The victim triggers available remediation mechanisms to recover the PayPal account. Once the user updates their credentials, the PayPal SP attempts to terminate all active sessions owned by the victim. Some sessions fail to be deleted, leaving the notification session unreachable. This is because the remediation mechanism skips deregistering the victim device with the notification server. In the meantime, all notifications will be delivered to the attacker.

The notification session is finally terminated when the victim initiates the service from a new device. The SP recognizes service requests from the new device and tries to register or replace the old device with the notification server. At this time, notification messages begin to be delivered to the user.

B. DELAYED TERMINATION

SPs let victims' sessions expire according to their individual expiration times. Asynchronous session timeouts can cause users to mistakenly believe that the recovery process has been completed before all sessions have expired.

1) MICROSOFT

Microsoft offers an integrated account for Windows operating systems, applications and services. Users can set up this account as their desktop, purchase applications from the Microsoft Store, and synchronize them across multiple devices. Microsoft SP supported the **Type 4** remediation mechanism.

a: IMPLEMENTATION

Microsoft SP offers a unique option for managing sessions device-by-device. This option is useful for Microsoft to

limit the number of devices authorized to use Microsoft's applications and services. The user can browse a list of devices to disconnect selected devices once the number of devices reaches the maximum. This option can provide a remediation mechanism for account compromises on known devices.

We tested and confirmed that Microsoft SP correctly implemented the three remediation mechanisms, as ultimately no unreachable sessions remained. However, the sessions were revoked due to delays. Sessions occupied by an adversary remain valid for some time after revocation. The adversary can gain access to the victim's information, including payments and history. Microsoft has officially announced that the grace period can last up to 24 hours [31]. The adversary can extend this period even longer by waiting for the user's responding actions. In the experiment, we maintained the unreachable session for 24 hours after the remediation was completed.

b: ATTACK

The user lost a notebook computer that was running Windows and signed into the Microsoft account. The user triggered one of the remediation methods. The user unlinks the lost device from the authorized devices list, updates the Microsoft account credentials, and entrusts the Sign-Out-Everywhere option to close any outstanding sessions. The user considers the recovery complete and may update credit card information or upload files to the OneDrive server within 24 hours after the remediation. In this case, the attacker can still gain access to new information before the timeout expires, which is a maximum of 24 hours. Even if the user detects suspicious activity, he will not be able to stop the attacker's access because the device whitelist does not include the attacker's device.

This incorrectness can lead the user to mistakenly believe that the account has been recovered. Consequently, the user resumes service while the attacker holds unreachable sessions. In the worst-case scenario, these unreachable sessions can cause permanent damage that existing remediation mechanisms cannot reverse.

2) TIKTOK

TikTok is a video-sharing application that allows users to create and share short-form videos on any topic. Additionally, users can chat live with their peers and send instant messages. This was a multi-platform in our category.

a: IMPLEMENTATION

The chat functionality creates a bilateral session with a peer. The chat session is prone to hijacking. TikTok supports the **Type 3** remediation mechanism. Unlike other sessions used in the control channel, the chat session does not expire immediately after the recovery mechanism is triggered. The closure lasted up to ten minutes. One security concern is that the delayed timeout period is automatically extended without a limit when a new chat enters the session. This is attributed to an implementation called ''sliding expiration'' [32]. This means that each time the session is accessed, the timeout will be reset to N minutes again. The timeout period is extended by N minutes from the user's last access time. This implementation can be especially useful in preventing a session from abrupt termination while the user is still actively using it, providing a better user experience.

b: ATTACK

After stealing a chat session, a bad actor sporadically sends chat messages to peers as if they were sent from a legitimate user. Even if the victim can recognize the lousy actor's activity, the user has no way to stop it.

C. BACKUP CODES

The backup code replaces the user's credentials when the credentials are unavailable. Typically, it is designed to be failsafe when users lose access to their primary credentials. Unlike other forms of authentication, such as session IDs and tokens, backup codes typically do not have an expiration time. While this is convenient for account recovery, it must be deleted at the same time as the remediation mechanism deletes cookies.

1) RECOVERY CODE

A 25-character Microsoft Windows recovery code is available upon request by authenticated users. Users can reset passwords with the recovery code alone when other techniques, like alternate email addresses and phones, are unable to do so. The code remains the same even after the user changes the credential.

The recovery code is optionally updated when the victim explicitly requests an update. The typical user is too clumsy to update the recovery code because the average user does not know how to update the code and cannot tell if the attacker has stolen the recovery code. Hence, the recovery code must be renewed automatically through remediation mechanisms. However, Microsoft did not renew the recovery under the remediation mechanism.

2) TRANSFER CODE IN MOBILE GAMES

A user can play games on different devices and platforms. User data and information in sequential sessions must be synchronized across devices. The transfer code is a random secret generated by the game application. Any user holding the valid transfer code can continue playing games on other devices. The transfer code is equivalent to a user credential and must be removed once recovery of the associated account begins.

The problem occurs in the game if the transfer code is irrevocable. For example, the transfer code for ''Fate Grand Order'' is a one-time use and remains the same until the user changes the device. The ''Pokémon Shuffle'' transfer code remains valid until the next generation and can only

be renewed after 30 days. These two game SPs provide indirect functions to void an existing transfer code. However, they do not offer the user history of transfer codes or the existence of an outstanding transfer code; therefore, it is almost impossible for the user to find the right time to revoke. The transfer code in the ''Evertale'' game lasts forever. The user cannot revoke a stolen transfer code because it does not offer any functionality. To be correct, the SP managing the session must bind the transfer code to an account and the remediation mechanism must offer a function to revoke all transfer codes associated with the compromised account or execute this function implicitly.

VII. DISCUSSION

In this section, we discuss the position our research occupies among existing studies, the challenges encountered during our research process, and the limitations of our study.

A. MOTIVATION

Broken authentication has grown increasingly prevalent in the modern web services. Users are usually the first to detect and attempt to reclaim control over their accounts. SPs have implemented various remediation mechanisms to enable users to recover their accounts securely.

A noticeable disconnection exists between the user's control over their accounts and the SPs' responsibility. The remediation mechanisms provided by SPs may need to align with user expectations, leading to confusion and potential missteps in the recovery process. Moreover, it is concerning that remediation efforts can fail. When this happens, users may be exposed to further risks, as their sense of security is falsely restored.

Our paper delves into this significant issue to bridge the gap. We strive to demystify available remediation mechanisms, enhancing their understanding and ability to navigate the recovery process effectively. We advocate for SPs to recognize the shortcomings of current remediation solutions and be proactive in implementing more robust, user-centric recovery options.

B. CONTRASTIVE STUDY

Our research addresses security concerns in the ''post-compromise'' phase. Our specific focus is on the recovery process of compromised accounts. Commonly, recovery is perceived as the process of regaining access to an account that has already been compromised. However, our research extends beyond simply regaining access. It encompasses terminating all access currently controlled by the attacker. Our study addresses a pivotal issue: the persistence of remaining sessions after the user has initiated a recovery process.

Our research points out the problems of recovery mechanisms provided by SPs. In most studies, recovery mechanisms are assumed to operate normally. However, we highlight these flaws of recovery mechanisms and define the remaining sessions as ''unreachable.'' This is a significant novelty of our research, and our measurements and case studies reveal how these unreachable sessions occur, what risks they pose, and how prevalent they are across various services.

Only a few studies have dealt with unreachable sessions. However, these studies do not focus on unreachable sessions but treat them as one method of extending attacks. Therefore, they only address unreachable sessions in limited scenarios where attacks occur and within limited recovery mechanisms. To the best of our knowledge, our paper is the first to mainly focus on unreachable sessions. Detailed differences and comparisons with related studies are extensively discussed in the Section VIII.

C. TECHNICAL CHALLENGES

Our measurements were conducted manually because automating the experimentation process for account compromise and recovery presents significant difficulties. The remediation mechanism requires user authentication, and many service providers apply various protection techniques. It includes detection and deterrence of automation, to prevent unauthorized access. Bypassing all these protective measures for automation is quite challenging.

Accurately gauging the duration for which an account has been compromised is the second challenge. Each service sets its session lifetime parameters, which may be maintained on the server-side and are distinct from the client-side lifetime. Moreover, sessions can be extended by user actions, necessitating the consideration of various interactions beyond merely waiting for a predetermined timeout period to elapse.

The third challenge is understanding the server-side implementation of session management and remediation mechanisms. Each SP has its unique infrastructure and protocols, often not publicly disclosed for security reasons. This lack of transparency makes it difficult for researchers and security professionals to assess the adequacy of the implemented mechanisms.

D. LIMITATION

A primary constraint of our research is its manual execution, which inherently challenges the scalability of our measurements. While we sought to mitigate this by focusing on widely used services and diverse authentication methods, our findings offer only a selective view of the entire service landscape. The manual analysis of our research might also introduce inconsistencies and potential measurement errors, particularly given that our data collection spanned two years. Though we endeavored to uphold the validity of our case studies by periodically reconducting experiments, discrepancies may persist.

Our methodology needs to measure the duration of attacker access via unreachable sessions precisely. The manual nature of our approach presents challenges in obtaining such specific data. Consequently, our study may categorize some sessions, which are controlled by timeouts, as persistently exposed. To mitigate this, we tailored our measurements according to the timeout periods publicly disclosed by

TABLE 6. A comparison table of our work with related studies.

Author (year)	Measurement of sessions remaining	Measurement across platforms	Includes non-SSO environments	Experiment on all mechanisms	Number of experiments set
Neil et al. (2021)	No	No	Yes	Yes	57 web sites
Ghasemisharif et al. (2018)	Yes	Yes	No	Yes	29 web sites 66 mobile apps
Sudhodanan and Paverd (2022)	Yes	No	Yes	No	150 web sites
Our Study (2023)	Yes	Yes	Yes	Yes	200 web sites 100 mobile apps 50 games

services. For services that did not disclose their timeout settings, we implemented a minimum delay of one hour in our measurements to ensure maximal accuracy within the constraints of our methodology.

The implementation of remediation by SPs remains undisclosed mainly. Consequently, our research is predicated on assumptions derived from client-side behavior and available information. This approach inherently limits our ability to pinpoint the exact root causes of unreachable sessions. Despite these constraints, our study adequately demonstrates the prevalence and nature of the potential risks across numerous services solely from a client-side perspective. Furthermore, we employed educated guesses based on various measurements and case studies. To elucidate the causes to the best of our ability, given the available data and observable trends.

VIII. RELATED WORK

User account security remains a pivotal concern in web security research. Extensive studies have addressed threats such as credential stuffing [33], [34] session hijacking [35], [36], [37], and password attacks [38], [39], [40], including responses to large-scale phishing [12], [41], spam [42] and targeted attacks [43], [44], [45], [46]. The availability of stolen credentials on underground forums [47] and the risks associated with them [48], [49], [50], [51] further underscore the critical need for robust account takeover prevention and remediation.

In post-compromise account recovery, the starting point is often the SP's detection and alert of a breach. ''Have I Been Pawned'' [20] pioneered public credential checking, spurring subsequent protocol improvement research [52], [53], [54], [55] and anomaly detection studies based on user behavior analytics [21], [56], [57]. Research has also delved into user responses to compromised account alerts [58], [59], examining security mental models [60], [61] and interpretations of warnings [62]. Findings indicate a general difficulty in prioritizing security advice, even among experts [22]. Neil et al. [63] focused on SPs' remediation advice and reception, revealing a stark deficiency in public remediation guidance. These studies assume that remediation is wholly carried out through the mechanisms provided by the SP.

Our research aims to go beyond detection and user reaction to examine the safety of the remediation mechanism itself. The most widely known remediation mechanism is credential change, and research has been conducted on the safety of the recovery process targeting this. Prior work on account recovery mechanisms investigated different authentication schemes [64], [65] and password reset strategies [66]. Password recovery schemes may also be vulnerable to man-in-the-middle (MitM) attacks [67], [68], [69]. Liu et al. [70] focus on user information security after account deletion, but this falls outside the scope of remediation that we are addressing. These studies, like our research, focus on the safety of the remediation mechanism. While these studies focus on vulnerabilities in the remediation mechanism, they do not consider other accesses an attacker could generate from a compromised account. We focus on this, define them as unreachable sessions, and research them.

To our knowledge, only a few studies have addressed unreachable sessions after remediation. Ghasemisharif et al. [71] present a novel attack scenario where the attacker uses the victim's IdP account to preemptively create an account for the victim on an RP at which the victim does not yet have an account. They also conduct a large-scale measurement of authentication and session management flaws that lead to vulnerabilities in subsequent research [72]. In their studies, to measure the long-term access of the proposed attacks, they trigger remediation mechanisms for hijacked SSO sessions and examine unreachable sessions.

These findings motivated our research, and their methodologies and experimental approaches serve as references throughout our study. Ghasemisharif et al.'s research focuses only on the access gained through the attacks in the SSO scenarios they propose, among all attacker accesses that should expire correctly during remediation. To fill this gap, our study covers areas not addressed in their research, by investigating access in general authentication scenarios they did not consider.

Sudhodanan and Paverd [73] build upon the work of Ghasemisharif et al. by presenting the concept of account pre-hijacking, achievable by even less sophisticated attackers. Their introduction of the ''Unexpired Session Attack'' emphasizes the issue of unreachable sessions that fail to expire post-password reset. This attack is pivotal to our study

as it underscores the flaws in remediation implementations by service providers. Unlike their work, which only measured unreachable sessions following password resets, our study comprehensively assesses vulnerabilities across the full suite of remediation mechanisms offered by SPs. By doing so, we gain a more granular understanding of the security of remediation processes.

The key distinction between the studies by Ghasemisharif et al. and Sudhodanan and our own lies in our targeted investigation of unreachable sessions. Their research primarily extends the attack models, examining unreachable sessions within limited recovery scenarios. In contrast, by concentrating on unreachable sessions, we consider all potential conditions that could trigger such scenarios, allowing for a more comprehensive study.

Table 6 describes as a comparative table within the related work section, contrasting our research with other studies targeting services' remediation mechanisms. It provides a comparative analysis of research directly relevant to our own, offering insight into how our study stands about the existing body of work. We compare the extent to which each study, including ours, measures the persistence of sessions after attempted remediation and whether these measurements span multiple platforms, including both Single Sign-On (SSO) and non-SSO environments. Additionally, we examine the thoroughness of each study in terms of testing all remediation mechanisms provided by service providers.

Neil et al.'s study offers a structured approach to remediation advice, providing valuable measurement data on how web services present remediation mechanisms. However, it does not extend to measuring the effectiveness of the remediation mechanisms. The research by Ghasemisharif et al., while closely aligned with our own in focus, is limited to SSO environments, presenting a gap in their scope of study. Sudhodanan and Paverd expanded upon Ghasemisharif's work by including non-SSO environments. However, their investigation into unreachable sessions is confined to password reset scenarios and does not encompass a broader range of web services.

IX. CONCLUSION

This paper has shed light on the critical implications of broken authentication within SPs and has evaluated the effectiveness of three distinct remediation mechanisms. Our analysis has been underpinned by a thorough measurement of unauthorized access instances to user accounts across popular SPs. Focusing on a specific set of web services, our investigation aimed to delineate the pervasive insecurity and identify the primary challenges impeding effective remediation.

The findings of our research have brought to the forefront a concerning trend: a substantial number of SPs have incorrectly implemented remediation methods, leading to the inaccessibility of certain sessions and rendering attacker access irreversible. Three case studies, each illustrating unique problems within SPs, were presented, and a root cause analysis was conducted based on both our measurements and these illustrative cases. Central to these issues is the SP's struggle to maintain an up-to-date inventory of generated sessions, coupled with a policy that often prioritizes user experience at the expense of security. This inherent tension has perpetuated vulnerabilities, making it imperative for SPs to reassess their security protocols and strike a balance between user convenience and robust authentication practices.

In essence, our work highlights the urgent need for Service Providers to reevaluate and strengthen their authentication mechanisms, emphasizing the importance of maintaining a vigilant and adaptive security posture. As we move forward, addressing the challenges outlined in this paper will be instrumental in fortifying online platforms against the ever-evolving landscape of cyber threats.

REFERENCES

[1] OWASP. (2021). *OWASP Top Ten*. Accessed: Sep. 18, 2023. [Online]. Available: https://owasp.org/www-project-top-ten/

[2] SecurityWeek. (2020). *Slack Vulnerability Allowed Hackers to Hijack Accounts*. Accessed: Sep. 18, 2023. [Online]. Available: https://www.securityweek.com/slack-vulnerability-allowed-hackers-hijack-accounts/

[3] Threatpost. (2021). *Exchange/Outlook Autodiscover Bug Spills 100K+ Email Passwords*. Accessed: Sep. 18, 2023. [Online]. Available: https://threatpost.com/exchange-outlook-autodiscover-bug-spills-100k-email-passwords/175004/

[4] The Record. (2023). *Norton LifeLock Says 925,000 Accounts Targeted By Credential-Stuffing Attacks*. Accessed: Sep. 18, 2023. [Online]. Available: https://therecord.media/norton-lifelock-says-925000-accounts-targeted-by-credential-stuffing-attacks

[5] Okta. (2022). *A Comparison of Cookies and Tokens for Secure Authentication*. Accessed: Sep. 18, 2023. [Online]. Available: https://developer.okta.com/blog/2022/02/08/cookies-vs-tokens

[6] C. Mainka, V. Mladenov, and J. Schwenk, "Do not trust me: Using malicious IdPs for analyzing and attacking single sign-on," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Saarbruecken, Germany, Mar. 2016, pp. 321–336, doi: 10.1109/EUROSP.2016.33.

[7] S. Sivakorn, I. Polakis, and A. D. Keromytis, "The cracked cookie jar: HTTP cookie hijacking and the exposure of private information," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2016, pp. 724–742, doi: 10.1109/SP.2016.49.

[8] E. Kim and H.-K. Choi, "Security analysis and bypass user authentication bound to device of windows hello in the wild," *Secur. Commun. Netw.*, vol. 2021, pp. 1–13, Jul. 2021, doi: 10.1155/2021/6245306.

[9] J. Navas and M. Beltrán, "Understanding and mitigating OpenID connect threats," *Comput. Secur.*, vol. 84, pp. 1–16, Jul. 2019, doi: 10.1016/j.cose.2019.03.003.

[10] Xenonstack. (2022). *Stateful and Stateless Applications and its Best Practices*. Accessed: Sep. 18, 2023. [Online]. Available: www.xenonstack.com/insights/stateful-and-stateless-applications

[11] M. Jones, J. Bradley, and N. Sakimura. *JSON Web Token (JWT)*, document RFC 7519, May 2015. [Online]. Available: http://www.rfceditor.org/rfc/rfc7519.txt

[12] M. Cova, C. Kruegel, and G. Vigna, "There is no free phish: An analysis of 'free' and live phishing kits," in *Proc. 2nd USENIX Workshop Off.*, vol. 8, San Jose, CA, USA, 2008, pp. 1–8.

[13] X. Shu, K. Tian, A. Ciambrone, and D. Yao, "Breaking the target: An analysis of target data breach and lessons learned," 2017, *arXiv:1701.04940*.

[14] F. Salahdine and N. Kaabouch, "Social engineering attacks: A survey," *Future Internet*, vol. 11, no. 4, p. 89, Apr. 2019, doi: 10.3390/fi11040089.

[15] Z. Tu, O. Turel, Y. Yuan, and N. Archer, "Learning to cope with information security risks regarding mobile device loss or theft: An empirical examination," *Inf. Manag.*, vol. 52, no. 4, pp. 506–517, Jun. 2015, doi: 10.1016/j.im.2015.03.002.

[16] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, and A. Forget, "Let's go in for a closer look: Observing passwords in their natural habitat," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dellas, TX, USA, Oct. 2017, pp. 295–310, doi: 10.1145/3133956.3133973.

[17] I. Dacosta, S. Chakradeo, M. Ahamad, and P. Traynor, "One-time cookies: Preventing session hijacking attacks with stateless authentication tokens," *ACM Trans. Internet Technol.*, vol. 12, no. 1, pp. 1–24, Jul. 2012, doi: 10.1145/2220352.2220353.

[18] Y. Mundada, N. Feamster, and B. Krishnamurthy, "Half-baked cookies: Hardening cookie-based authentication for the modern web," in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur.*, Xi'an, China, May 2016, pp. 675–685, doi: 10.1145/2897845.2897889.

[19] C. Peeters, C. Patton, I. N. S. Munyaka, D. Olszewski, T. Shrimpton, and P. Traynor, "SMS OTP security (SOS)," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Nagasaki, Japan, May 2022, pp. 2–16, doi: 10.1145/3488932.3497756.

[20] Have I Been Pwned. *Check If Your Email Has Been Compromised in a Data Breach*. Accessed: Sep. 18, 2023. [Online]. Available: https://haveibeenpwned.com

[21] S. Shah, B. Shah, A. Amin, F. Al-Obeidat, F. Chow, F. J. L. Moreira, and S. Anwar, "Compromised user credentials detection in a digital enterprise using behavioral analytics," *Future Gener. Comput. Syst.*, vol. 93, pp. 407–417, Apr. 2019, doi: 10.1016/j.future.2018.09.064.

[22] D. Akhawe and A. P. Felt, "Alice in warningland: A large-scale field study of browser security warning effectiveness," in *Proc. 22nd USENIX Conf. Secur.*, Washington, DC, USA, 2013, pp. 257–272.

[23] K. Thomas, J. Pullman, K. Yeo, A. Raghunathan, P. G. Kelley, L. Invernizzi, B. Benko, T. Pietraszek, S. Patel, D. Boneh, and E. Bursztein, "Protecting accounts from credential stuffing with password breach alerting," in *Proc. 28th USENIX Conf. Secur.*, Santa Clara, CA, USA, 2019, pp. 1556–1571.

[24] D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, and D. Song, "Towards a formal foundation of web security," in *Proc. 23rd IEEE Comput. Secur. Found. Symp.*, Edinburgh, U.K., Jul. 2010, pp. 290–304, doi: 10.1109/CSF.2010.27.

[25] M. Woschek, "OWASP cheat sheets," *OWASP Found.*, pp. 132–135, Apr. 2015.

[26] Alexa. *Top 200 Sites on the Web*. Accessed: Mar. 1, 2022. [Online]. Available: https://www.alexa.com/topsites

[27] Data.ai. *Top Apps Ranking*. Accessed: Jun. 1, 2023. [Online]. Available: https://www.data.ai/intelligence/top-apps

[28] Amazon. *Web Apps vs. Native Apps vs. Hybrid Apps—Difference Between Types of Web and Mobile Applications*. Accessed: Sep. 18, 2023. [Online]. Available: https://www/aws.amazon.com/compare/the-difference-between-web-apps-native-apps-and-hybrid-apps

[29] DreamFactory. (2023). *Stateful vs. Stateless Web App Design*. Accessed: Sep. 18, 2023. [Online]. Available: https://blog.dreamfactory.com/stateful-vs-stateless-web-app-design

[30] Google. (2022). *Integrate Play Games Services With Existing Identity Solution*. Accessed: Sep. 18, 2023. [Online]. Available: https://developer.android.com/games/playgames/integrating-pgs-existing-id-solutions

[31] Microsoft. *Sign Me Out*. Accessed: Sep. 18, 2023. [Online]. Available: https://account.live.com/proofs/manage/additional

[32] M. Kumar. (2021). *Reissue JSON Web Token (JWT) With Sliding Expiration Using ASP.NET Core*. Accessed: Sep. 18, 2023. [Online]. Available: https://muneshkumar578.medium.com/reissue-json-web-token-jwt-with-sliding-expiration-using-asp-net-core-c-432119ceb647

[33] S. Sahin and F. Li, "Don't forget the stuffing! Revisiting the security impact of typo-tolerant password authentication," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 252–270, doi: 10.1145/3460120.3484791.

[34] P. Doerfler, K. Thomas, M. Marincenko, J. Ranieri, Y. Jiang, A. Moscicki, and D. McCoy, "Evaluating login challenges as aDefense against account takeover," in *Proc. World Wide Web Conf.*, San Francisco, CA, USA, May 2019, pp. 372–382, doi: 10.1145/3308558.3313481.

[35] S. Calzavara, A. Rabitti, and M. Bugliesi, "Sub-session hijacking on the web: Root causes and prevention," *J. Comput. Secur.*, vol. 27, no. 2, pp. 233–257, Mar. 2019, doi: 10.3233/JCS-181149.

[36] K. Drakonakis, S. Ioannidis, and J. Polakis, "The cookie hunter: Automated black-box auditing for web authentication and authorization flaws," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 1953–1970, doi: 10.1145/3372297.3417869.

[37] S. Farooqi, F. Zaffar, N. Leontiadis, and Z. Shafiq, "Measuring and mitigating oauth access token abuse by collusion networks," in *Proc. Internet Meas. Conf.*, London, U.K., Nov. 2017, pp. 355–368, doi: 10.1145/3131365.3131404.

[38] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Vienna, Austria, Oct. 2016, pp. 1242–1254, doi: 10.1145/2976749.2978339.

[39] M. Islam, M. S. Bohuk, P. Chung, T. Ristenpart, and R. Chatterjee, "Araña: Discovering and characterizing password guessing attacks in practice," in *Proc. 32nd USENIX Conf. Secur.*, Anaheim, CA, USA, 2023, pp. 1019–1036.

[40] R. Verma, N. Dhanda, and V. Nagar, "Enhancing security with in-depth analysis of brute-force attack on secure hashing algorithms," in *Proc. Trends Electron. Health Inform.*, Kanpur, India, 2022, pp. 513–522, doi: 10.1007/978-981-16-8826-3_44.

[41] K. Thomas, F. Li, A. Zand, J. Barrett, J. Ranieri, L. Invernizzi, Y. Markov, O. Comanescu, V. Eranti, A. Moscicki, D. Margolis, V. Paxson, and E. Bursztein, "Data breaches, phishing, or malware: Understanding the risks of stolen credentials," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dallas, TX, USA, Oct. 2017, pp. 1421–1434, doi: 10.1145/3133956.3134067.

[42] R. Kaur, S. Singh, and H. Kumar, "Rise of spam and compromised accounts in online social networks: A state-of-the-art review of different combating approaches," *J. Netw. Comput. Appl.*, vol. 112, pp. 53–88, Jun. 2018.

[43] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: Analysis of a botnet takeover," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, Chicago, IL, USA, Nov. 2009, pp. 635–647, doi: 10.1145/1653662.1653738.

[44] R. Hiesgen, M. Nawrocki, T. C. Schmidt, and M. Wählisch, "The race to the vulnerable: Measuring the Log4j shell incident," 2022, *arXiv:2205.02544*.

[45] B. Jabiyev, S. Sprecher, K. Onarlioglu, and E. Kirda, "T-REQs: HTTP request smuggling with differential fuzzing," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 1805–1820, doi: 10.1145/3460120.3485384.

[46] W. R. Marczak, J. Scott-Railton, M. Marquis-Boire, and V. Paxson, "When governments hack opponents: A look at actors and technology," in *Proc. 23rd USENIX Conf. Secur.*, San Diego, CA, USA, 2014, pp. 511–525.

[47] A. Mirian, J. DeBlasio, S. Savage, G. M. Voelker, and K. Thomas, "Hack for hire: Exploring the emerging market for account hijacking," in *Proc. World Wide Web Conf.*, San Francisco, CA, USA, 2019, pp. 1279–1289, doi: 10.1145/3308558.3313489.

[48] C. VanDam, J. Tang, and P. N. Tan, "Understanding compromised accounts on Twitter," in *Proc. Int. Conf. Web Intell.*, Leipzig, Germany, 2017, pp. 737–744, doi: 10.1145/3106426.3106543.

[49] J. Onaolapo, E. Mariconti, and G. Stringhini, "What happens after you are PWND: Understanding the use of leaked webmail credentials in the wild," in *Proc. Internet Meas. Conf.*, Santa Monica, CA, USA, 2016, pp. 65–79, doi: 10.1145/2987443.2987475.

[50] P. Peng, C. Xu, L. Quinn, H. Hu, B. Viswanath, and G. Wang, "What happens after you leak your password: Understanding credential sharing on phishing sites," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Auckland, New Zealand, Jul. 2019, pp. 181–192, doi: 10.1145/3321705.3329818.

[51] M. H. N. Ba, J. Bennett, M. Gallagher, and S. Bhunia, "A case study of credential stuffing attack: Canva data breach," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Las Vega, CA, USA, Dec. 2021, pp. 735–740.

[52] K. C. Wang and M. K. Reiter, "Detecting stuffing of a user's credentials at her own accounts," in *Proc. 29th USENIX Conf. Secur.*, 2020, pp. 2201–2218.

[53] L. Li, B. Pal, J. Ali, N. Sullivan, R. Chatterjee, and T. Ristenpart, "Protocols for checking compromised credentials," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, London, U.K., Nov. 2019, pp. 1387–1403, doi: 10.1145/3319535.3354229.

[54] B. Pal, T. Daniel, R. Chatterjee, and T. Ristenpart, "Beyond credential stuffing: Password similarity models using neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2019, pp. 417–434, doi: 10.1109/SP.2019.00056.

[55] B. Pal, M. Islam, M. S. Bohuk, N. Sullivan, L. Valenta, T. Whalen, C. Wood, T. Ristenpart, and R. Chatterjee, "Might I get Pwned: A second generation compromised credential checking service," in *Proc. 31th USENIX Conf. Secur.*, Boston, MA, USA, 2022, pp. 1831–1848.
[56] D. Freeman, S. Jain, M. Dürmuth, B. Biggio, and G. Giacinto, "Who are you? A statistical approach to measuring user authenticity," in *Proc. NDSS*, San Diego, CA, USA, 2016, pp. 21–24, doi: 10.14722/ndss.2016.23240.
[57] E. K. Boahen, W. Changda, and B. M. B. Elvire, "Detection of compromised online social network account with an enhanced KNN," *Appl. Artif. Intell.*, vol. 34, no. 11, pp. 777–791, 2020, doi: 10.1080/08839514.2020.1782002.
[58] E. M. Redmiles, "'Should I worry?' A cross-cultural examination of account security incident response," in *Proc. IEEE Symp. Secur. Priv.*, San Francisco, CA, USA, 2019, pp. 920–934.
[59] Y. Huang, B. Obada-Obieh, and K. Beznosov, "Users' perceptions of Chrome compromised credential notification," in *Proc. 18th USENIX Conf. Usable Priv. Secur.*, Boston, MA, USA, 2022, pp. 155–174.
[60] C. Bravo-Lillo, L. F. Cranor, J. Downs, and S. Komanduri, "Bridging the gap in computer security warnings: A mental model approach," *IEEE Secur. Privacy*, vol. 9, no. 2, pp. 18–26, Mar. 2011.
[61] E. M. Redmiles, A. R. Malone, and M. L. Mazurek, "I think they're trying to tell me something: Advice sources and selection for digital security," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2016, pp. 272–288.
[62] E. M. Redmiles, N. Warford, A. Jayanti, A. Koneru, S. Kross, M. Morales, R. Stevens, and M. L. Mazurek, "A comprehensive quality evaluation of security and privacy advice on the web," in *Proc. 29th USENIX Conf. Secur.*, 2020, pp. 89–108.
[63] L. Neil, E. Bouma-Sims, E. Lafontaine, Y. Acar, and B. Reaves, "Investigating web service account remediation advice," in *Proc. 17th USENIX Conf. Usable Priv. Secur.*, 2021, pp. 359–376.
[64] J. Bonneau, C. Herley, P. C. V. Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc. IEEE Symp. Secur. Privacy*, San Francisco, CA, USA, May 2012, pp. 553–567.
[65] E. Gerlitz, M. Häring, C. T. Mädler, M. Smith, and C. Tiefenau, "Adventures in recovery land: Testing the account recovery of popular websites when the second factor is lost," in *Proc. 19th USENIX Conf. Usable Priv. Secur.*, Anaheim, CA, USA, 2023, pp. 227–243.
[66] J. H. Huh, H. Kim, S. S. V. P. Rayala, R. B. Bobba, and K. Beznosov, "I'm too busy to reset my linked in password: On the effectiveness of password reset emails," in *Proc. CHI Conf. Human Factors Comput. Syst.*, Denver, CO, USA, May 2017, pp. 387–391, doi: 10.1145/3025453.3025788.
[67] N. Gelernter, S. Kalma, B. Magnezi, and H. Porcilan, "The password reset MitM attack," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 251–267.
[68] M. Guri, E. Shemer, D. Shirtz, and Y. Elovici, "Personal information leakage during password recovery of internet services," in *Proc. Eur. Intell. Secur. Informat. Conf. (EISIC)*, Uppsala, Sweden, Aug. 2016, pp. 136–139.
[69] T. Innocenti, S. Mirheidari, A. Kharraz, B. Crispo, and E. Kirda, "You've got (a reset) mail: A security analysis of email-based password reset procedures," in *Proc. Detection Intrusions Malware, Vulnerability Assessment, 18th Int. Conf.*, 2021, pp. 1–20, doi: 10.1007/978-3-030-80825-9_1.
[70] Y. Liu, Y. Jia, Q. Tan, Z. Liu, and L. Xing, "How are your zombie accounts? Understanding users' practices and expectations on mobile app account deletion," in *Proc. 31th USENIX Conf. Secur.*, Boston, MA, USA, 2022, pp. 863–880.
[71] M. Ghasemisharif, A. Ramesh, S. Checkoway, C. Kanich, and J. Polakis, "O single sign-off where art thou? An empirical analysis of single sign-on account hijacking and session management on the web," in *Proc. 27th USENIX Conf. Secur.*, Baltimore, MD, USA, 2018, pp. 1475–1492.
[72] M. Ghasemisharif, C. Kanich, and J. Polakis, "Towards automated auditing for account and session management flaws in single sign-on deployments," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2022, pp. 1774–1790.
[73] A. Sudhodanan and A. Paverd, "Pre-hijacked accounts: An empirical study of security failures in user account creation on the web," in *Proc. 31th USENIX Conf. Secur.*, Boston, MA, USA, 2022, pp. 1795–1812.

JEONGHO LEE received the B.S. degree in science and computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2018, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

His research interests include network security, system security, and user authentication.

HYOUNG-KEE CHOI received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanata, GA, USA, in 2001.

From 2001 to 2004, he was with Lancope, where he guided and contributed to research in internet security. He is currently a Professor with the College of Software, Sungkyunkwan University, South Korea. His research interests include network security and vulnerability assessments.

JIN HEE YOON is currently pursuing the joint bachelor's and master's degree in computer science and engineering with Sungkyunkwan University, Suwon, South Korea.

Her research interests include network security and digital forensics.

SEONGJUNE KIM received the B.S. degree in computer science and engineering from Sungkyunkwan University, Suwon, South Korea, in 2023.

His research interests include network security and information security.