Breaking Mobile Notification-based Authentication with Concurrent Attacks Outside of Mobile Devices

Ahmed Tanvir Mahdad
mahdad@tamu.edu
Texas A&M University
College Station, Texas, USA

Mohammed Jubur*
mjabour@jazanu.edu.sa
Jazan University
Jazan, Saudi Arabia

Nitesh Saxena
nsaxena@tamu.edu
Texas A&M University
College Station, Texas, USA

ABSTRACT

Notification-based authentication is an emerging Two-Factor Authentication (2FA) and passwordless solution that leverages interactive notifications on mobile devices to establish an additional layer of security beyond passwords. This method has gained popularity due to its convenience and ease of deployment in organizational settings. In this work, we aim to evaluate the effectiveness of notification-based authentication systems when a malicious entity is present on the user's computer, such as a keylogger or malicious extension, without compromising the mobile devices or communication channels. Furthermore, we investigate how the lack of information provided to users during the authentication workflow can lead to the approval of malicious authentication requests. Notably, we highlight the vulnerability of cross-service attacks, where an attacker authenticates to Service B while the user is attempting to authenticate to Service A. Our proof-of-concept attack program demonstrates the susceptibility of various notification-based authentication systems, and our user study reveals an alarming 82.2% cross-service attack success rate. These findings suggest a potential vulnerability in notification-based authentication systems, where the attacker compromise user account without compromising possession-factor device, such as smartphones.

CCS CONCEPTS

• **Security and privacy → Multi-factor authentication**.

*Work done as a PhD student in the SPIES Lab.

ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10.
https://doi.org/10.1145/3570361.3613273

KEYWORDS

2FA, MFA, Notification, Authentication, Security

ACM Reference Format:
Ahmed Tanvir Mahdad, Mohammed Jubur, and Nitesh Saxena. 2023. Breaking Mobile Notification-based Authentication with Concurrent Attacks Outside of Mobile Devices. In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3570361.3613273

1 INTRODUCTION

Authentication systems deployed in the wild rely primarily on passwords as the primary means of knowledge-based authentication. However, password-only systems are known to have vulnerabilities [17, 34] and usability issues [7, 33]. One drawback of these systems is the need for users to remember multiple passwords for various services, which can cause significant cognitive burden [25]. This burden may lead users to reuse the same password across multiple sites, increasing the risk of compromising high-value accounts, such as banking or credit card accounts. If a password is stolen from a low-value account that does not contain sensitive personal or financial information, high-value accounts where the same password is reused can also be at risk [11].

To counter this issue and other known vulnerabilities, Two-Factor Authentication (2FA) has been introduced which uses an additional authentication factor (such as possession or inherence) along with passwords to provide an extra layer of security in case of password theft or compromise. Among the deployed 2FA systems, One Time Pin (OTP) is a popular 2FA system that is also known for causing security and usability issues. More recently, notification-based 2FA systems are introduced as a more usable yet supposedly secure alternative to prove the smartphone's possession in 2FA.

In notification-based 2FA, service providers communicate with the users with interactive notifications (i.e., those that can be approved or denied) through smartphones which are assumed to be secure. The communication channel is believed to be protected, and a very low cognitive burden is imposed on the user. As a result, notification-based 2FA (e.g., Duo) has become increasingly popular and has already been trusted by many organizations. This method has also been

used in passwordless authentication systems, in conjunction with inherence factors such as fingerprints. According to prior research [29], push notification authentication has been found to achieve a high usability score when compared to other two-factor authentication (2FA) methods. Additionally, users tend to perceive push notifications as user-friendly and effective in bolstering account security [10].

Service providers collect users' consent for authentication through notifications using two methods. The first involves generating a phone call to the user's pre-registered number, which the user must answer and press a button to approve the notification. The second method involves sending a push notification to the user's registered smartphone, which they can approve or deny by tapping a button from the notification bar or within the notification details. We refer to the authentication system that uses notifications on smartphones or wearable devices in their workflow as *Push-2FA* in this paper. We also denote the *Push-2FA* variant that uses only approve or deny button in the notification or phone call, as *confirm*.

Terminal verification requires users to verify the binding between the login session on the terminal/browser and the notification on the 2FA device by matching unique identifiers (i.e., a randomly generated number) in *Push-2FA*. One such method is *compare-and-confirm*, where users compare the identifier in the terminal and confirm it from the phone. Another variant is *select-and-confirm*, which requires users to tap the button containing the displayed identifier. These variants are employed as a protection against fatigue attacks in the confirm method (discussed in detail in Section 7). We include all these variants of *Push-2FA* in our work.

According to our observation, the notification (both the phone call and the push notification) communicates minimum authentication information in the push notification body or in a voice call, which hinders users from making an informed decision while authorizing the notification. This scenario can be more significant when the notification is shown in wearable devices (e.g., smartwatch) where the display size is much smaller to render important authentication information (e.g., place, generation time).

Our Work: Security Analysis of Notification-based 2FA: In this work, we analyze information communicated with the user in notification-based 2FA and demonstrate how attackers can exploit it using a concurrent attack. Our proposed "Concurrent Login Attack" blocks the user session while simultaneously triggering a new malicious session from the terminal. This attack compromises the notification-based 2FA system without compromising the 2FA device itself, such as a smartphone. We discuss the threat model in detail in Section 4.1. Notably, we demonstrated the capability of our attack in cross-service scenarios where users try to authenticate to Service A, and the attacker collects the user's consent to login to Service B via a user study.

Previous works [6] commonly held the belief that attackers must compromise both the knowledge factor (such as a password) and possession factor (such as a smartphone) to successfully compromise 2FA systems. However, our attack framework and user study counter this belief by demonstrating how malware in the user's terminal can undermine the additional security provided by smartphones and wearables in different types of currently deployed notification-based authentication systems. It can also defeat proposed secure *Push-2FA* schemes from the literature[28] that offer protection against concurrent attack on *Push-2FA*.

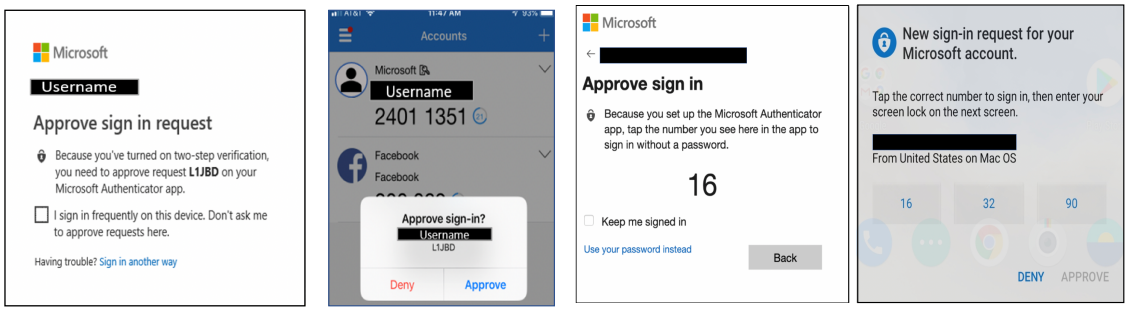
Our Contributions: Our contributions in this work are three-fold:

(1) **Analysis of information communicated in mobile devices by notification-based 2FA** : Our analysis focused on currently deployed *Push-2FA* systems (e.g., Google, Duo) and notifications communicated to users via smart devices, such as smartphones and smartwatches. We assessed how these notifications can aid in the decision-making process during authentication and found that insufficient information provided to users can result in significant vulnerabilities.

(2) **Design and implementation of concurrent login attack framework**: We developed an attack framework, "Concurrent Login Attack", to assess the security of *Push-2FA* systems when faced with malicious programs on user's terminal. Our evaluation of currently deployed *Push-2FA* and passwordless schemes, which utilize push notification authentication as part of their authentication method, revealed that nearly all of them are susceptible to this attack. We found that proposed academic systems claiming to protect against concurrent attacks (e.g., Prakash et al. [28]), are also vulnerable to it. Our proof-of-concept attack has lower detectability in the presence of anti-malware software and is stealthy, as it generates only a single notification on registered mobile devices.

(3) **A user study to evaluate the effectiveness of cross-service attack on *Push-2FA***: Using our proposed cross-service attack, an attacker can gain access to a high-value service while the user is attempting to authenticate to another service. To determine the effectiveness of this attack, we conducted a lab-based user study in which participants were presented with both benign and cross-service notifications during authentication. The results showed that users approved 82.2% of all cross-service notifications. Notably, we observed a lower detection rate of cross-service attacks in the more secure *Push-2FA* variants designed to prevent fatigue attacks (discussed in detail in Section 7).

Att demonstrations are shown at: https://sites.google.com/view/push2fademo/home.



(a) compare-and-confirm UIs from terminal and smart-phone **(b) select-and-confirm UIs from terminal and smartphone**

Figure 1: Snapshots of compare-and-confirm and select-and-confirm UIs

2 AUTHENTICATION NOTIFICATIONS

2.1 Notifications in Mobile Device

Push Notifications: Push notifications utilize application-specific secure channels to deliver alerts, general communications, and marketing campaigns to users. More recently, service providers have adopted push notifications as a means of collecting users' consent during authentication attempts. Users can approve or deny notifications from the notification bar or open the application to view more detailed information about the attempt. To use this authentication system, users have to install the service provider's app and pre-register.

Service providers use different variants of notifications for push notification authentication. We consider three such variants, which we named *confirm*, *compare-and-confirm*, and *select-and-confirm*. *confirm* is a simple *Push-2FA* method which shows the interactive "approve/deny" button in the notification body. This variant is popular among well-known service providers (e.g., Google prompt, LastPass [19], Duo [13]). In *compare-and-confirm* variant, a unique identifier is displayed in the authentication terminal screen and the push notification body. The users are expected to compare the identifiers and accept the notification if they match. Snapshots of the terminal and 2FA prompt of this scheme are shown in Figure 1a. In the *select-and-confirm* variant, the terminal displays a specific number, while the notification body presents a set of numbered buttons, including the correct one. The user has to select the correct button to establish the user's presence. This scheme is designed to avoid the problem of potential skip-through behavior (where the user accepts notification without comparing) inherent with *compare-and-confirm* and *confirm* variants. Microsoft uses *select-and-confirm* variant in their passwordless [37] scheme where users have to provide phone lock credentials (e.g., fingerprint) in addition to approve *select-and-confirm* push notification. Snapshots of the terminal view and the 2FA device prompt for *select-and-confirm* are shown in Figure 7c.

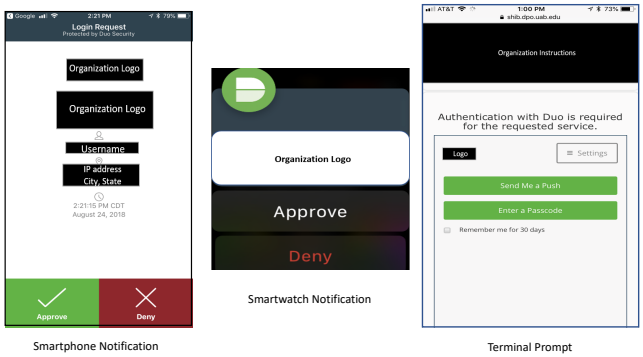


Figure 2: Snapshots of smartphones, smartwatch, and user terminal view of "confirm" *Push-2FA*

Opening the push notification in app provides users with additional details (e.g., location, time) to make an informed decision when approving the correct attempt. Wearable devices (e.g., smartwatch) shows less information due to their limited display size. Figure 2 displays push notification UI snapshots on smartphone, smartwatch, and terminal views.

Phone Call Verification: One approach to obtaining user consent for an authentication attempt is to place a phone call to a pre-registered number and ask the user to press a specific button to approve the request. With this method, users retain the ability to deny the request by pressing an alternative button and can report suspicious activity. However, phone calls only ask users to make decision by pressing a button, and no other information is communicated. Without communicating service name, location, request time to users, this method is vulnerable to cross-service attacks as it lacks necessary information to make informed decisions.

2.2 Notification Information

Service providers show information about authentication attempts in the notification body and the notification details. Some *Push-2FA* providers allow users to interact from

the notification body (e.g., Duo), while some others require users to open the app to approve the notification (e.g., id.me) where they can show detailed information. We noted our observations in Table 1 (described in Section 6).

Notification from Notification Bar: Most service providers allow users to approve/deny authentication notifications directly from the notification bar. Due to space limitations, service providers can not include much information in the notification body, which results in the skip-through behavior of users who approve the notification without carefully examining it. Most service providers use the name/logo along with the approve/deny button in the notification bar.

Notification Inside App: The notification app shows more authentication information inside the app compared to notification body. Most of the *Push-2FA* providers we studied show service, name, logo, location, and usernames. Some also communicate authentication terminal device names and IP addresses, which is useful during an authentication attempt.

Notification from the watch: Some service providers provide the feature to approve the notification from the smartwatch for the user's convenience. However, due to limited space on the watch screen, generally, it shows only the service provider logo as authentication information (e.g., Duo).

Phone Call: Some service providers (e.g., Duo) offer users the opportunity to verify an authentication attempt via a phone call. Here, after receiving the call, the user has to press a specific number from the keypad to verify the attempt. We observed that Duo does not communicate the service name or other authentication information with the user, making it susceptible to cross-service attacks.

Terminal Verification: Most of the *Push-2FA* service providers do not consider terminal verification (i.e., user verification if the notification is generated by the same authentication attempt) in their workflow. Only Microsoft uses it for two of their *Push-2FA* variants (compare-and-confirm, select-and-confirm). Terminal verification is a powerful tool (also demonstrated at [28]) to protect from concurrent attacks generated by adversaries (e.g., the attack shown at [16]). However, our designed Concurrent Login Attack can also defeat terminal verification, which we demonstrated and will elaborate more on Section 4.2.

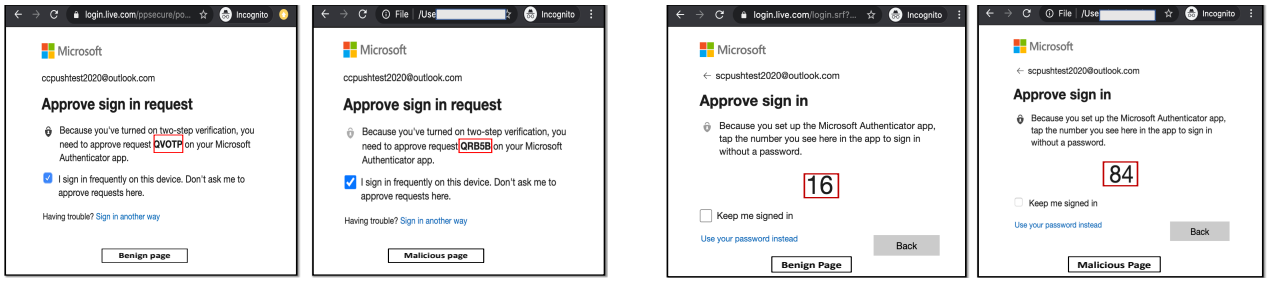
3 RELATED WORKS

Since authentication using push notification is a state-of-the-art authentication method, not much work has been done regarding its security issues. Zhi Xu and Sincun Zhu studied vulnerabilities of push notification itself [41] and figured out its possibility of malicious usages like phishing and spamming. Another similar work is Android push notification malware analysis by Hyun et al. [14] where malware can be executed with the help of push notifications. However, in our work, we did not compromise the push notification service or invoke any malware in the 2FA device (e.g., smartphone), and our attack methodology is different from phishing or spamming. Li et al. [18] worked on security issues of push notification cloud services and showed how an attacker can exploit them. Our proposed attack does not compromise the push notification service itself, and it is out of the scope of our work. Ding et al. [12] have done a combination of formal verification and testing on push notifications. All these works are related to the security analysis of push notification itself and the push notification service. On the other hand, we specifically work with real-world push notification authentication systems and evaluate how external malicious actors can compromise mobile-based push notifications without compromising the mobile device itself.

Jubur et al. show a way of bypassing push notification authentication in their work HIENA [16] by sending concurrent notifications to mobile devices. Through a user study, they also assess the risk of users being deceived by concurrent malicious notifications. In contrast, our designed attack methodology blocks user's request and generates single malicious notification to the user's mobile device. As only a single notification (with identical authentication information) is shown in the user's device during the active attack, our attack methodology is stealthier than their work [16] which generates multiple notifications in the mobile device that 38% of users can detect according to their study. Furthermore, our attack methodology can detect a user's authentication attempt, activate itself, and generate a concurrent attack, which is an added advantage compared to their work.

As discussed before, researchers are already working on incorporating more user engagement and preventing concurrent login attacks on push notification authentication. Prakash et al. [28] proposed six different methods to accomplish this purpose. They claimed their proposed authentication could prevent concurrent attacks using the randomness property of different challenges presented during the authentication process. However, our designed attack is capable of blocking legitimate users' requests and showing patterns sent to the attacker on the attacker's controlled page (similar to the example shown in Figure 3), which will persuade users to enter the attacker's pattern in their registered phone. Thus, our proposed Concurrent Login Attack can defeat all of the defensive schemes proposed in [28].

Mahdad et al. [20] designed an attack explicitly to defeat OTP (One Time PIN) 2FA. The key difference with our work is that, in notification-based 2FA, our attack convinces the user into accepting a wrong notification which is not applicable to OTP cases. Importantly, their work [20] is essentially



(a) Comparison of benign and malicious page during attack on compare-and-confirm *Push-2FA*

(b) Comparison of benign and malicious page during attack on select-and-confirm *Push-2FA*

Figure 3: An example of similarity of benign and malicious pages during attack on compare-and-confirm and select-and-confirm *Push-2FA*

a *passive same-service attack* with some optional active elements, whereas our attack is *fully active with cross-service attack* capability.

4 ATTACK DESIGN

4.1 Threat Model

We assume an adversary who can install a malicious program in the user's authentication terminal. The malicious program will not require any administrative privilege from the OS during installation. It will not compromise the OS core functionality after installation and will not access restricted areas of the OS. We also assume that the adversary will convince users to install a benign-looking browser extension containing malicious code. The extension will not compromise browser itself, rather, monitor all redirection from address bar, can block and redirect to another site.

According to the Sonicwall Cyberthreat report 2021 [31], 5.6 billion malware are reported along with 4.8 trillion intrusion attempts, including client application attacks and remote code execution. According to the NTT Global Threat Intelligence Report 2019 [24], about 33% authentication credential theft is done by malware, associated with keyloggers. Also, researchers discovered malicious browser extensions that affect millions of users [15, 38, 42] by redirecting them to a malicious site. Cybercriminals increasingly use it nowadays [36], and it is a well-known and common cybersecurity threat [39]. So, the assumption of malware and malicious browser extensions represents a practical threat.

We also assume targeted users will use laptop or desktop computers running on Windows operating systems and the Google Chrome browser to authenticate to a targeted service. Most importantly, the adversaries will not compromise any mobile device (e.g., smartphone, wearable devices) used as a possession factor device in *Push-2FA*.

Furthermore, the adversaries will not have the ability to steal session cookies from browsers or any other session hijacking capability. Our proposed attack demonstrates more effectiveness compared to traditional session hijacking techniques, particularly in the context of cross-service attacks. In Section 7 (Comparison with Session Hijacking Attack), we further elaborate on the additional advantages offered by our method over the conventional session hijacking attack.

4.2 Attack Workflow

Attack Components: We have used three primary attack components. Our designed Attack module, a keylogger program (KL), has the capability to learn usernames and passwords beforehand, enabling it to generate patterns based on the known credentials for use in the actual attack. When a user's input matches the pattern, it constitutes ***condition 1 (C1)***, which triggers the Background Browser Session (BBS) to activate. The BBS can perform user activity automation from the background using the Chrome driver [9].

Another attack component is a malicious browser extension (BE), which monitors internet traffic and looks for a match with the URL of the targeted authentication service. This match constitutes ***condition 2 (C2)***. When C2 is satisfied, the BE blocks the user's request and redirects them to an attacker-controlled page where attacker-specific altered information or instructions would be shown (e.g., instructions to approve the notification or provide a unique identifier). After some time (currently set to 15 seconds in our implementation), the user will be redirected to the authentication information entry page (i.e., login page).

The purpose of redirecting the user after a certain amount of time is to create the impression of an error, with the intention of avoiding arousing suspicion. Even if users detect an ongoing attack, they might be unable to stop substantial real-time harm, like emptying a bank account.

Malicious Program Installation: Malicious program component Keylogger (KL) is a standalone program and can run without installation, and the Background Browser Session (BBS) (which is a jar file) can be run with a portable Java Runtime Environment (JRE). Neither procedure requires administrative privileges during operation in the user terminal.

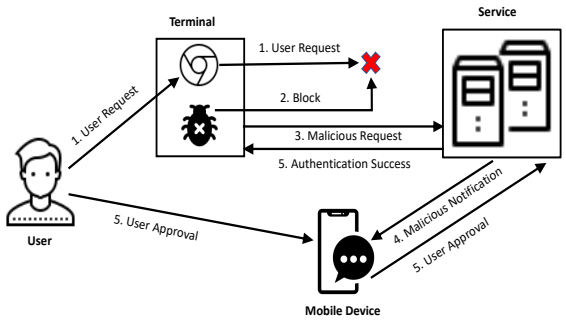


Figure 4: Overview of General Attack Workflow

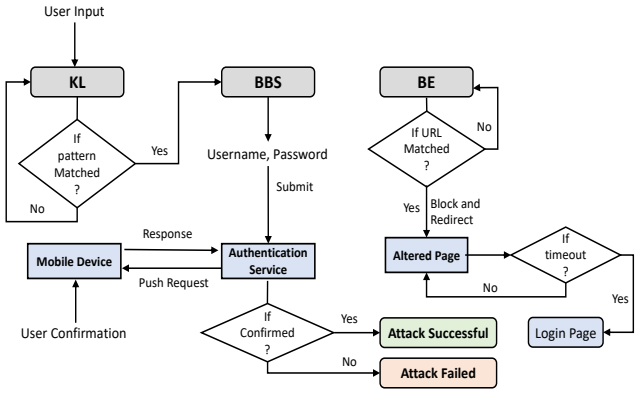


Figure 5: Block Diagram of General Attack

Cyber attackers frequently hide malicious browser extension code within seemingly benign extensions, a tactic that has been documented in recent reports (e.g., [8, 43]).

4.2.1 ***General Workflow.*** We present an active concurrent attack which is initiated when the user starts the authentication process. The General workflow of this attack is primarily applicable to the *confirm Push-2FA*. The overall attack workflow and block diagram are shown in Figure 4 and Figure 5, respectively. Further steps are elaborated below.

(1) **Step 1:** When the user starts the authentication process on the targeted service's site, *KL* starts monitoring keystrokes and launches *BBS* when condition *C1* is met.
(2) **Step 2:** *BE* blocks the initial request made by the user to the authentication service
(3) **Step 3:** *BE* redirects users to an attacker-controlled page which is a similar-looking web page to the targeted service's one (as shown in Figure 3). At the same time, *BBS* sends the request to the service.
(4) **Step 4**: Only the authentication request generated from *BBS* can reach the authentication service, and only a single notification is generated for the user.

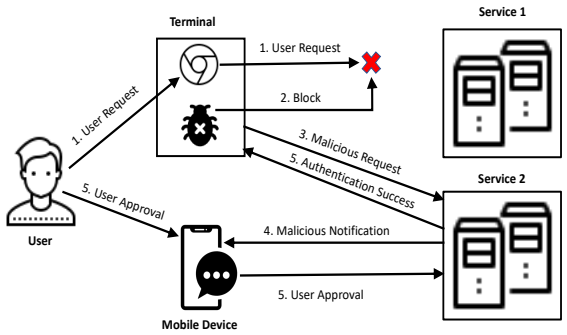


Figure 6: Overview of Cross-service Attack

(5) **Step 5**: As the user has the intent to authenticate and only a single request is generated with the same authentication information (e.g., location, browser), the user would approve it, which eventually approves the attacker's authentication request.

4.2.2 ***Cross-service Attack Workflow.*** In the cross-service attack, *KL* gathers authentication credentials for both S1 (the intended service for user authentication) and S2 (a different service utilizing the same *Push-2FA* service). In an active attack scenario, when the user intends to authenticate on S1, the malicious program blocks the user's effort and covertly initiates the authentication process on S2 in the background. Refer to Figure 6 for the high-level workflow of this cross-service attack.

(1) **Step 1**: *KL* collects username and password for both S1 and S2.
(2) **Step 2**: Same as the general workflow Step (1). Here *BBS* starts authentication for S2.
(3) **Step 3**: Same as the general workflow Step (2).
(4) **Step 4**: Same as the general workflow Step (3). It shows a similar-looking attacker-controlled page of S1 (the service user intended to authenticate).
(5) **Step 5**: At this point, users expect a notification (A push notification or phone call). For phone call, as no service name is communicated, users will approve the notification, assuming the call is generated for their request. For push notifications, the users are likely to accept the notification due to skip-through behavior.

4.2.3 ***Attack on compare-and-confirm Push-2FA.*** Upon receiving a user request, the authentication service displays a distinct identifier on the terminal screen (e.g., browser). This identical identifier appears on the subsequent push notification to the phone. The user's task is to compare these identifiers and validate the authentication attempt. The steps are outlined below.

(1) **Step 1-2**: Same as the general workflow.

(2) **Step 3**: *BE* redirects the user to an attacker-controlled page which will show the attacker's unique identifier generated from *BBS* authentication request.
(3) **Step 4**: Same as the general workflow.
(4) **Step 5**: As the notification is generated with the attacker's unique identifier, which would be shown in the browser window by malicious *BE*, the user would approve the notification thinking that it is generated from their request.

4.2.4 ***Attack on select-and-confirm Push-2FA.*** Unlike the *compare-and-confirm Push-2FA*, here, users have to select a button containing a specific identifier (e.g., a two-digit number) and tap it to approve the authentication request from the mobile device. *select-and-confirm* variant is designed to counter the skip-through behavior of the users (i.e., not paying much attention to the unique identifier) during compare-and-confirm. However, the Concurrent Login Attack can defeat this specific variant using the following steps.

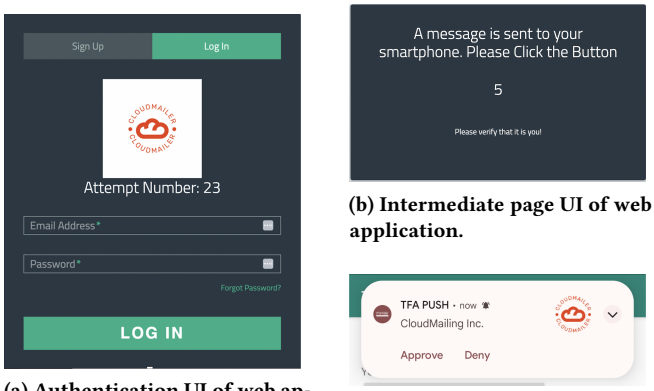
(1) **Step 1-2**: Same as the general workflow.
(2) **Step 3**: Same as the *compare-and-confirm*.
(3) **Step 4**: Same as the general workflow.
(4) **Step 5**: As the unique identifier (generated for *BBS* request) is shown in the user's browser window, and the legitimate user's request is blocked by *BE*, the notification arrives in users mobile device will show buttons with identifiers where one of them is the attacker's unique identifier.

4.3 Implementation Overview

Attack component *KL* is developed using libraries of Python 3.7 and *BBS* is developed using Selenium Webdriver [30] and PhantomJS [26]. Selenium web driver requires chromedriver [9] to be present to run the scripts. These automated browser tools can replicate any user's action and send requests to authentication services on behalf of them. They can also run in headless (i.e., running from the background and the user has no trace of it) mode, which helps the attack to be stealthy. Selenium web driver needs a java runtime environment to make it operational.

The attack components can be packaged as python executables which can call the executable jar files when *C1* is satisfied. Java runtime environment and chromedriver can be pushed during malware injection. None of the components needs an installation that requires administrative privilege.

Component *BE* used Google Chrome browser extension libraries to monitor URLs in the address bar and block any request when a pattern is matched (during submission of legitimate users' request). It can also redirect the user to the attacker-controlled web page and redirect back to the authentication services login page after a moment. These malicious browser extension codes can be placed inside any benign-looking necessary browser extension, which can act as *BE* later.



(a) Authentication UI of web application.

(b) Intermediate page UI of web application.

(c) Notification UI of smartphone application.

Figure 7: UI of web application and smartphone application used in the user study.

5 USER STUDY DESIGN

Concurrent login attacks on the same service are simple to execute, as the attacker can send a single notification that appears identical to a legitimate one. Since the attack is active, the user anticipates receiving a notification and may not realize that it is part of an attack. In contrast, cross-service attacks involve sending notifications that contain different information, such as the service name and logo, from the one the user intended to authenticate. To evaluate users' responses to cross-service attacks, we conducted a user study in which we implemented this attack for the confirm, compare-and-confirm, and select-and-confirm variants.

For phone-call-2FA, no service name is currently communicated with the user during the call, making a cross-service attack straightforward to execute without providing any distinguishable information to the user. Therefore, we did not include it in our user study.

5.1 Implementation

We have developed a secure authentication system for the user study that utilizes push notifications to provide an additional layer of security beyond traditional username and password credentials. Our authentication system is composed of two primary components:
Web Application: We have developed a web application for the user study using HTML, PHP, JavaScript, CSS, and MySQL that offers user registration and authentication features. We used Google Firebase to implement push notification authentication, which we integrated with both our web and smartphone applications. Our registration feature allows

users to sign up using their email addresses. After entering their username and password, participants are redirected to an intermediate page. Here, instructions are provided to approve a notification (for the confirm variant) or a unique identifier (for the compare-and-confirm and select-and-confirm variants). Our application automatically checks if the correct button is pressed in the smartphone notification and makes an authentication decision accordingly.

Smartphone Application: We have developed an Android application specifically for push notification authentication, which we integrated with Firebase. When a logged-in participant makes an authentication attempt, the app receives a push notification that displays the service name, logo, and a unique identifier to provide participants with the information they need to make an informed decision. During this study, participants were asked to approve or deny notifications only from the notification body. Once a participant responds, the information is sent to our web application to finalize the authentication decision.

Our primary objective was to assess whether participants could distinguish the service name/logo in the case of a cross-service attack, which is present in the notification body. The notification detailed page shown in the app displays other details about the notification attempt, such as location, IP, and device name (see Table 1). To keep the participants' task simple, we only implemented approval/denial from the notification body, which fully served our purpose.

Snapshots of the web application and smartphone application used in the user study are depicted in Figure 7.

5.2 Authentication Attempts

Throughout the study, we sent two primary types of notification during the authentication attempt and recorded user's response for them:

Benign Notification: Most of the notifications sent during the study are benign notifications where the same service, logo, and same unique identifier is shown in the notification body.

Cross-service Notification: We also generated cross-service notifications (comprising 30% of the total notifications) for each variant, which displayed a different service name and logo.

5.3 Study Metrics

After a notification is sent to participants, they have the option to either approve, deny or not respond to it. We denote approved benign notifications as $BN_{approved}$, denied benign notifications as BN_{denied}, and benign notifications that have not been responded to as BN_{nr}. Similarly, we use $CS_{approved}$ to denote cross-service approved notifications, CS_{denied} to denote denied notifications, and CS_{nr} to denote notifications that have not been responded to.

To evaluate users' responses to benign and cross-service notifications, we used two metrics: *Benign Notification Success Rate (BNSR)* and *Cross-Service Notification Success Rate (CNSR)*. They are calculated using following formulas:

$$BNSR = \frac{\sum_i BN_{approved}}{\sum_i BN_{approved} + \sum_j BN_{denied} + \sum_k BN_{nr}} \quad (1)$$

$$CNSR = \frac{\sum_i CS_{approved}}{\sum_i CS_{approved} + \sum_j CS_{denied} + \sum_k CS_{nr}} \quad (2)$$

For an ideal *Push-2FA* system, we expect the *BNSR* to be 100% and *CNSR* to be 0%. Here, i, j, and k represent the total number of approved, denied, and pending requests, respectively.

5.4 Study Protocol

This study has been approved by our university's Institutional Review Board (IRB), and participation is entirely voluntary. In accordance with IRB guidelines, we have followed standard procedures to ensure participant privacy and confidentiality. Participants are free to withdraw from the study at any time, and we have taken care not to save any identifying information in our system.

We recruited a group of 20 participants of varying ages who were familiar with both two-factor authentication and push notification authentication. Each participant was provided with a laptop and an Android smartphone for authentication purposes. A researcher acted as the facilitator and explained the different variants of push notification authentication that were used in the study, including the information communicated in each variant such as the service name, logo, and unique identifier.

We structured our study into distinct stages to help organize the research process:

Orientation Phase: In this phase, the facilitator provided an overview of the different types of push notification authentication and the information that would be communicated to participants in the notification body. Participants were instructed to interact with the notification (i.e., approve or deny it) only from the notification body. The facilitator also showed the name and logo of the dummy email provider service that would be displayed on the login page. To address participant privacy concerns, we did not save their email/password in our system; therefore, the facilitator created a dummy account and password for each participant specifically for the study.

Pre-test Quesionnaire: As part of our data collection process, we administered a pre-test questionnaire that gathers

Table 1: Information communicated by notification-based authentication system

Service Provider(UI)	Service Name	Logo	Location	Time	IP	Username	Device Name	Approve/Deny	Terminal Verification
Duo (Notification Body)	✓	✓	✗	✗	✗	✗	✗	✓	✗
Duo (App)	✓	✓	✓	✓	✗	✓	✗	✓	✗
Duo (Phone Call)	✗	✗	✗	✓	✗	✗	✗	✓	✗
Google 2SV (Notification Body)	N/A	✓	✗	✗	✗	✓	✗	✗	✗
Google 2SV (Notification Details)	N/A	✓	✓	✓	✗	✓	✓	✓	✗
LastPass (Notification Body)	✗	✗	✗	✗	✗	✗	✗	✓	✗
LastPass (Notification Details)	✗	✗	✗	✗	✗	✓	✗	✓	✗
LastPass (Phone Call)	✗	✗	✗	✓	✗	✗	✗	✓	✗
Microsoft (Notification - confirm)	✓	✗	Country Only	✗	✗	✓	✗	✓	✗
Microsoft (Notification - compare-and-confirm)	✓	✗	✗	✗	✗	✓	✗	✓	✓
Microsoft (Notification - select-and-confirm)	✓	✗	Country Only	✗	✗	✗	✓	✗	✓
ID.me (Notification)	✓	✗	✗	✗	✗	✗	✗	✗	✗
ID.me (App)	✓	✓	✓	✓	✓	✓	✓	✓	✗

demographic information. This questionnaire included questions about the participant's age group, gender, and familiarity with using 2FA and push notification authentication systems.

Familiarization Phase: During this phase, participants were asked to authenticate in our system using the provided username and password and received notifications. The phase consists of 10 attempts, which cover three different variants of push notification authentication: confirm, compare-and-confirm, and select-and-confirm. This practice phase is designed to ensure that users understand the study workflow and become accustomed to using the system.

Data Collection Phase: During this phase, participants were required to authenticate themselves in our system 30 times. The three types of push notifications were presented in a random order, with 10 notifications of each type. 30% of these notifications are designed to display different service names and logos. We recorded each participant's response to the push notification in this phase.

Post-test Questionnaire: The questionnaire administered during this phase aimed to gather feedback from participants on any suspicious behavior they may have observed during the study. In addition, we asked specific questions to elicit further details on such behavior, including possible reasons. Finally, we inquired about the frequency with which participants noticed the URL of the intermediate page (i.e., the page that displays the unique identifier).

During both the familiarization and data collection phases, each attempt takes an average of 30 seconds. With a 5-second wait time between each attempt, the familiarization phase lasts approximately 6 minutes ($10 \times 35 = 350$ seconds), and the data collection phase lasts around 18 minutes as ($30 \times 35 = 1050$ seconds), for each participant. The attempts are presented back-to-back to the participants.

6 EVALUATION AND RESULTS

6.1 Mobile Notification Information

User possession is established in notification-based authentication systems through interactive communications (e.g., push notification, phone call) to the user where they are responsible for scrutinizing and approving the proper notification. Here, the service providers display some essential information beforehand to help in making decisions on authentication. We list down information that is generally communicated with the users during the authentication process.

Service Name: When the push notification authentication system supports multiple services (e.g., Duo), it is important to display the service provider's name (referred to as "Service Name" throughout the paper) in the notification body or communicate it during the phone call. Otherwise, the authentication system would be vulnerable to *cross-service attack*.

Service Logo: Sometimes, the logo is more eye-catching, and the user can easily distinguish between two different logos. Moreover, it is convenient to show logos instead of service names on devices with limited displays (e.g., smartwatch).

IP and Location: The IP and location information is vital to detect any remote or suspicious authentication attempt. Although the attacker can spoof the IP address easily, it is an important indicator to be checked carefully.

Generation Time: Sometimes, the attacker used to send random notifications to users hoping that the user would approve them mistakenly in order to clear the notifications [35]. Also, they can send multiple notifications as shown in [16] where examining sending time is crucial before approving any notifications.

Username: To ensure approval of the correct authentication request, it is important to know the request originator's username. Careful checking of username/email address will prevent the users from malicious authentication attempts.

Some *Push-2FA* designers provide the feature of approving/denying the notification from the notification body. Others require opening the notification detail page to enforce more user engagement in the authentication process and show more detailed information that would help them to make an informed decision. In the case of phone calls, users can approve/deny notifications by pressing specific buttons (instructions communicated with the user). Some of the *Push-2FA* authentication systems (e.g., compare-and-confirm, select-and-confirm) also provide unique identifier with each notification which is also communicated to the user terminal to protect users from fatigue attacks.

We analyzed notifications from 5 top service providers (Google, Duo, LastPass, Microsoft, ID.me) that use *Push-2FA* for authentication, and summarized our findings in Table 1. Our study shows that service providers limit the authentication details shown to users in the notification body, often only including an approve/deny button. Notably, *Duo* and *Id.me* display only the service name/logo, while *LastPass* does not show the service name at all. This lack of information during authentication is a significant vulnerability that can make the authentication system vulnerable to cross-service and random notification attacks.

Some service providers (e.g., Duo, LastPass) provide the "Call Me" option to provide more flexibility in the authentication. However, we observe that none of them communicates the service name (i.e., for which organization /provider the call is generated) during the phone call and just ask the user to press a specific button (e.g., button "3" for Duo) to approve the notification. By taking advantage of users' skip-through behavior, the adversary can easily introduce a cross-service attack in push notification service providers, which are used in multiple organizations (e.g., Duo).

Users can only see detailed information when they open notification details. However, some providers (e.g., LastPass) show only the username as detailed information. Furthermore, device name and location/IP are also critical for informed decisions. However, some providers (e.g., Microsoft) only display country names as location information, leaving them vulnerable to remote attacks.

Careful users can identify their notification source using the device name displayed in the notification details. However, our analysis indicates that only *Google* and *Id.me* provide this device information. Unfortunately, authentication request time is not shown in interactive notifications (e.g., Duo, LastPass), creating an opportunity for Random Notification attackers.

6.2 User Study Result Analysis

Demographic Information: Among the participants, 50% were between the ages of 31-40, while 45% were between the ages of 21-30. The remaining 5% were in the age group of 41-50. All participants are university graduate students and 50% of them are male and the other 50% are female. All participants were familiar with two-factor authentication and had previously used push notification authentication. Among the participants, 80% reported using two-factor authentication for banking and email, while 70% reported using it for social media.

Benign Notification: Throughout the study, a total of 420 benign notifications across three push notification variants were sent. Using Equation 1, we calculated a benign notification success rate (BNSR) and achieved an overall success rate of 92.38%. Users denied 5% of the benign notifications and did not respond to 2.62% of them. These results suggest that users were able to efficiently manage the system and follow instructions. The overall *BNSR* and breakdown of notification success rates across all variants are presented in Table 2.

Cross-service Notification: We have sent 180 cross-service notifications simulating our proposed attack throughout this study to all participants. To evaluate the effectiveness of our approach, we calculated the Cross-service Notification Success Rate (CNSR) using Equation 2. Our results show a high success rate of 82.22%, indicating that our proposed attack was successful in most cases. However, we also observed a denial rate of 16.11%, which suggests that some participants were able to detect and deny the cross-service notifications. We have shown overall *CNSR* and breakdown of approval, denial, and no response rate for all three variants of push notification authentication in Table 2.

Among the variants, participants denied 40% of cross-service notifications for the confirm variant, 6.67% cross-service notifications from the compare-and-confirm variant, and only 1.67% notifications from the select-and-confirm variant. Interestingly, we noticed that when participants were assigned an additional cognitive task, such as comparing unique identifiers, they were more likely to miss changes in service names or logos. As a result, the cross-service attack detection rate was lower in the protective scheme designed to prevent fatigue attacks [35] by engaging users to compare/select a number in the notification approval process.

Attacker-controlled Intermediate Page: The attack program blocks the user's request and redirects them to an attacker-controlled intermediate page, which looks similar to the original page (as shown in Figure 3). This allows the attacker to display their unique identifier, as in the case of compare-and-confirm and select-and-confirm. Our user study revealed that 40% of participants never checked the intermediate page URL, while 20% checked it only a few times (less than 40% of attempts). Only 20% reported checking it every time. This survey result reinstates the attack opportunity using an intermediate page URL by the attackers.

Post-test Questionnaire: We have asked questions about participants' experiences during the study. Of those surveyed, 40% reported not noticing any suspicious behavior, such as changes in service names within notifications. A significant number of users (35%) believed that cross-service notifications were generated due to a system error, while only a small number (15%) suggested that such notifications were caused by malicious activity on their devices. Throughout the study, 45% of participants did not deny any notifications. Among all participants, 50% reported denying notification because they noticed a change in the service name, while only 20% reported noticing a change in logo. Among the individuals who believed there was a system error, they approved 68.25% of the malicious attempts and denied the remaining 31.75%.

Table 2: Summary of User Study Findings.

Scenario	Variant	Response			BNSR	CNSR
		Approve	Deny	NR		
Benign	confirm	85.71%	11.43%	2.86%	**92.38%**	–
	compare-and-confirm	95.71%	1.43%	2.86%		
	select-and-confirm	95.71%	2.14%	2.14%		
Attack	confirm	56.67%	40.00%	3.33%	–	**82.22%**
	compare-and-confirm	91.67%	6.67%	1.67%		
	select-and-confirm	98.67%	1.67%	0.00%		

NR – No Response

6.3 Summary of Attack Detectability

We list down attack detectability from the user terminal and 2FA device in Table 3. In the same-service attack, the user will see no difference in the notification information (i.e., information listed in Table 2) and the number of notifications (the user receives a single notification that they expected) at the 2FA device compared to the legitimate scenario. So, there is no way to detect same-service attacks from the 2FA notification. Similarly, for phone-call-2FA, no service name is communicated with the user during a phone call, hence, there is no way to detect cross-service attacks from the 2FA device. As a result, we marked these cases as **No** in terms of detectability in Table 3.

From the user study we observed that, participants only approved 56.67% cross-service notification for the confirm variant. So, we marked the detectability as **Moderate** in that case. We also noticed that participants approved more than 90% notifications from compare-and-confirm and select-and-confirm variants. As such, we mark the detectability as **Very Low** here. Detection from the user terminal largely depends on whether users noticed the change in URL for the altered intermediate page. According to user study outcome, only 20% of participants checked the URL always, and 40% never checked them. So, we mark the detectability from the user terminal as **Low** for all cases.

Table 3: Summary of Attack

Notification Variant	Attack Variant	Detectable in 2FA Device	Detectable in Terminal
Confirm	Same-service	No	Low
	Cross-service	Moderate	Low
Compare-and-Confirm	Same-service	No	Low
	Cross-service	Very Low	Low
Select-and-Confirm	Same-service	No	Low
	Cross-service	Very Low	Low
Phone Call	Same-service	No	Low
	Cross-service	No	Low

Table 4: Scan Result of Desktop and Web-based Anti-malware Engines

Name	Detected?
Windows Defender [23]	✗
Avast [2]	✗
MalwareBytes [21]	✗
Kaspersky Security Cloud [1]	✗
Sophos Home [32]	✗
Avira [4]	✗
AVG [3]	✓[1]
Virustotal [40]	✓[2]
Mcafee Total Protection (Free Trial) [22]	✗

✓- Detected, ✗- Not Detected

6.4 Detection by Anti-Malware Programs

Anti-malware programs detect malware through signature-based analysis and behavior-based analysis. Signature-based analysis matches the code of a malicious program with known malware signatures. However, it's possible to evade detection by refactoring suspicious code. We have designed our program to avoid overloading the user's terminal with multiple requests, consuming an excessive amount of memory, or accessing restricted files in the operating system. Table 5 shows a comparison of resource consumption between our program and three other benign applications. Additionally, we have refactored our code and incorporated custom libraries to circumvent signature-based analysis performed by anti-malware programs. These characteristics help our program avoid detection during behavior analysis by desktop-based anti-malware programs.

We tested our proof-of-concept program with eight desktop anti-malware programs and the online tool VirusTotal [40]. It remains undetected by all of the desktop-based anti-malware programs and most anti-malware engines in virustotal, as shown in Table 4. Only two out of sixty engines detected it as a threat. However, during further testing, we evaluated these two engines using other benign and simple python executables. Interestingly, we found that they classified each of these executables as malware. Therefore, the detection turned out to be a false positive.

[1]After showing the warning and performing an initial scan, the program is allowed to run.

[2]Only 2 engines detected it which turns out to be a false alarm

Table 5: Comparative resource consumption of our attack program with Google Chrome, Microsoft Word and Skype.

Usage	Computational Range			
	Attack Program	Chrome	Word	Skype
CPU (%)	0.01 – 20.1	7.5 – 51.6	0 – 31.5	0.5 – 50.1
RAM (MB)	26 – 127	760 – 1150	68 –110	131–320
Power[3]	VL – L	L – H	L – VH	L– M
Network (Mbps)	0	0 – 11.3	0	0 – 4.6

VL - Very Low, L - Low, M - Moderate, H - High, VH – Very High

7 DISCUSSION & FUTURE WORK

Limitation of User Study: In our user study, we used custom web and mobile applications rather than participants' everyday apps, due to ethical concerns. Participants used our provided computer and smartphone. We did not implement all user interfaces found in real-world push 2FA systems. Instead, we used a standard push notification body on Android that conveyed all necessary information. These factors could have affected participant behavior and response. The study required participants to authenticate 30 times, potentially leading to habitual bias. However, we achieved a 92.38% benign application approval rate, indicating that participants were capable of completing the task. Overall, 16.11% of notifications were denied, with 40% of "confirm" variant notifications being denied, indicating that participants noticed changed information during the cross-service attack. We have selected participants from a pool of graduate students who are habituated to using push notification authentication daily in their university authentication procedure. This may introduce a selection bias in the study. However, it can be inferred that if graduate students, who are likely to be technologically sound, cannot detect the cross-service attack, it might be even more challenging for average users to identify.

Comparison with Random Attack on *Push-2FA*: In the MFA-fatigue attack, the attacker can send multiple push notifications on the user's mobile device randomly. In that case, the user has a possibility of accidentally accepting the notification to clear multiple annoying notifications thinking that it shows up due to a bug [35]. However, a vigilant user can defeat this attack by carefully reviewing the information in the notification body (e.g., location, IP, device) or discarding unexpected notifications that they are not expecting. On the other hand, "Concurrent Login Attack" is an active attack generated from the same user terminal which sends only a single notification to the user's phone. As a result, it would be very difficult to distinguish these notifications as users would expect it at that time and all other information, such as location, and device name would be the same.

Comparison with Active Phishing Attack on *Push-2FA*: The adversary may trick users into providing authentication credentials to an attacker-controlled site and generate notifications on their phones by requesting the credentials simultaneously. However, since the request is sent from a remote machine, the IP address, location, and machine name may differ, allowing a vigilant user to notice. Although spoofing IP addresses and locations are easy for attackers, doing so for each user can be challenging. Our attack design avoids the need for IP or location spoofing, as the initial request is generated from the same device. This limits detection opportunities, even for the most careful users in our case.

Comparison with Session Hijacking Attack: Our designed concurrent login attack creates independent sessions, which differs from session hijacking attacks. This approach offers more advantages, such as increased flexibility for attackers, as the user cannot terminate the attacker's session or prevent their activities. Notably, our attack enables more damaging cross-service attacks, unlike session hijacking. However, it is important to note that session hijacking is not an attack against the 2FA system per se. Instead, it leverages an existing user session to carry out malicious activities. According to Ballare-Rogaway [5], relay attacks such as session hijacking is not considered as a valid attack against a secure protocol. They argue that a valid attack on a secure protocol occurs when a malicious entity successfully masquerades as a trusted entity, effectively bypassing the security measures in place. Our proposed attack method establishes a new, independent session specifically for attackers, separate from the user's session, and behaves like a trusted entity to compromise the 2FA system, which constitutes a valid attack as per the definition provided by Ballare-Rogaway [5]. Our focus is to explore vulnerabilities in the authentication workflow, rather than stealing session cookies.

Concurrent Login Attack Vs. Passwordless Authentication: Passwordless authentication employs possession factor verification methods (e.g., security key, push notification) alongside device verification techniques (e.g., fingerprint, PIN) for user authentication. Concurrent login attacks can compromise passwordless schemes that rely on notifications to establish device possession, effectively reducing the attacker's challenges. Additionally, since the attack is active and users are expecting a notification during the attack, they are likely to provide a screen lock or biometric verification afterward, making it easier for the attacker to access the user's device. Notably, in passwordless authentication, the attacker does not need to capture passwords, which reduces the overall complexity of the attack. However, Keylogger (KL) module is still needed to capture the username.

Concurrent Login Attack Vs. Password Managers: Using a password manager can provide an additional layer of security by eliminating the need for users to manually enter their login credentials into the terminal, thereby reducing the risk of a keylogger program, such as our KL module, intercepting the keystrokes. However, it's important to note

[3] Power Metrics used in Windows 10

that password managers can also be vulnerable to concurrent login attacks, which can compromise the manager itself and allow attackers to retrieve the stored passwords. In the attack demonstration, we have shown how the LastPass password vault can be compromised, as a representative example.

Other 2FA Deployments: Two-Factor Authentication (2FA) deployments, particularly in the financial services sector, prioritize strengthening security in the customer authentication process. These services commonly rely on third-party ID providers like Duo [13] and Ping ID [27] for employee authentication, while customer authentication utilizes methods such as One Time PIN (OTP) or biometrics. Ping ID, for instance, employs SMS-OTP, the "confirm" variant of push notification authentication, or biometric verification (e.g., fingerprint) as Multi-Factor Authentication (MFA) methods.

However, a vulnerability arises in Ping ID due to the absence of service names, such as the organization's name, in the notification and SMS bodies. This exposes Ping ID to a cross-service attack that we have devised. Furthermore, SMS OTP verification, widely employed by financial services as a 2FA method for customer-facing services, is susceptible to the general attack flow depicted in Figure 5. Additionally, some services utilize phone fingerprint verification as a 2FA method to confirm user presence.

In both scenarios, adversaries can obstruct the user's request and simultaneously send a fraudulent request to the service, resulting in an SMS containing the adversary's OTP and a fingerprint verification request. Since the targeted users are anticipating a notification at that moment, and there are no apparent distinctions between the genuine SMS and fingerprint request, users may be deceived into approving them by providing their fingerprint and entering the OTP on their device. Adversaries can then capture this information and exploit it to authenticate their own session. So, here, SMS-OTP does not effectively mitigate the risk of compromise posed by our designed attack.

Potential Mitigations & Challenges: For phone-call 2FA, service providers must add service information during the call to circumvent cross-service attacks. Perhaps using a different audio sound for different classes of services may help users better detect potential attacks.

To ensure the security of push notification systems, it is crucial to display important identifying information, such as location, IP, time, and device name, when the approve/deny button is presented to the user. Failing to provide this information makes push schemes vulnerable to random and concurrent attacks, including our proposed attack. It is essential for identifying information and security measures to become standard practice across all service providers.

It is recommended that users should deploy phishing site detection tools and carefully examine URLs of intermediate pages during authentication attempts, not only the login page. While this approach can improve security to some extent, its effectiveness depends on the efficiency of the detector or the user's attention to the browser address bar.

Authentication system designers should make the logo larger and more visible to the user, especially with devices with a limited interface, such as smartwatches. In our threat model, we assume that adversaries will not compromise the mobile device (e.g., smartphone) used as a possession factor, nor intercept the push notification service registration process. Consequently, push service providers like Duo will store and display the original logo. While an enhanced logo visibility can aid in detecting the attack within our adversary model, it cannot guarantee absolute prevention.

Implementing certain mitigation measures can reduce the risk of the proposed concurrent login attack to some degree. However, since this is a fundamental attack, it cannot be completely eliminated, and some level of risk may still remain. Moreover, it can achieve security at the cost of lowering the system's usability and more reliance on user diligence, which may undermine the founding motivation behind these schemes as they were designed to make the login processes less cumbersome for the user.

Future Work: The researchers have the opportunity to develop similar attacks on more OS-browser combinations (e.g., smartphone OS and browser) to design more general attacks. They also have the scope to design a more secure *Push-2FA* authentication system that can prevent similar attacks while keeping usability as a priority.

8 CONCLUSION

With the emerging popularity of notification-based authentication systems, it also becoming target of cybercriminals aiming to exploit its shortcomings. In this work, we focused on such vulnerabilities and designed a concurrent login attack from the user terminal, which can compromise notification-based authentication system without compromising the possession-factor device. The demonstration of our proof-of-concept attack and user study exhibits this critical vulnerability in action. The vulnerability disclosed by our work can provide opportunities for researchers to work on more secure yet usable notification-based authentication systems in the future. It will also help the service providers to streamline their authentication process by carefully evaluating our observations and recommendations.

ACKNOWLEDGMENTS

This work is funded in part by NSF grants: OAC-2139358, CNS-2201465 and CNS-2152669.

REFERENCES

[1] AO Kaspersky Lab. 2021. Kaspersky Security Cloud - Free. https://www.kaspersky.com/free-cloud-antivirus.

[2] Avast Foundation. 2021. Avast Free Antivirus. https://www.avast.com/en-us.

[3] Avast Software s.r.o. 2021. AVG Free Antivirus. https://www.avg.com/en-us/.

[4] Avira Operations GmbH & Co. KG. 2021. Avira Antivirus. https://www.avira.com/.

[5] Mihir Bellare and Phillip Rogaway. 1993. Entity authentication and key distribution. In *Annual international cryptology conference*. Springer, 232–249.

[6] Joseph Bonneau, Cormac Herley, Paul C Van Oorschot, and Frank Stajano. 2012. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *2012 IEEE Symposium on Security and Privacy*. IEEE, 553–567.

[7] Kay Bryant, John Campbell, et al. 2006. User behaviours associated with password security and management. *Australasian Journal of Information Systems* 14, 1 (2006).

[8] Chrome Unboxed. 2022. These 30 malicious Chrome extensions just showed their true colors, affecting millions. https://chromeunboxed.com/color-changing-malicious-chrome-extensions.

[9] Chromium Project. 2021. ChromeDriver- WebDriver For Chrome. https://chromedriver.chromium.org.

[10] Jessica Colnago, Summer Devlin, Maggie Oates, Chelse Swoopes, Lujo Bauer, Lorrie Cranor, and Nicolas Christin. 2018. "It's not actually that horrible" Exploring Adoption of Two-Factor Authentication at a University. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–11.

[11] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. 2014. The tangled web of password reuse.. In *NDSS*, Vol. 14. 23–26.

[12] Junhua Ding, Wei Song, and Dongmei Zhang. 2014. An approach for modeling and analyzing mobile push notification services. In *2014 IEEE International Conference on Services Computing*. IEEE, 725–732.

[13] Duo. 2021. Duo Two Factor Authentication and Endpoint Security. https://duo.com.

[14] Sangwon Hyun, Junsung Cho, Geumhwan Cho, and Hyoungshick Kim. 2018. Design and analysis of push notification-based malware on android. *Security and Communication Networks* 2018 (2018).

[15] Informa PLC Informa UK Limited. 2021. Researchers Discover Two Dozen Malicious Chrome Extensions. https://www.darkreading.com/vulnerabilities-threats/researchers-discover-two-dozen-malicious-chrome-extensions.

[16] Mohammed Jubur, Prakash Shrestha, Nitesh Saxena, and Jay Prakash. 2021. Bypassing push-based second factor and passwordless authentication with human-indistinguishable notifications. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 447–461.

[17] Daniel V Klein. 1990. Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the 2nd USENIX Security Workshop*. 5–14.

[18] Tongxin Li, Xiaoyong Zhou, Luyi Xing, Yeonjoon Lee, Muhammad Naveed, XiaoFeng Wang, and Xinhui Han. 2014. Mayhem in the push clouds: Understanding and mitigating security hazards in mobile push-messaging services. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 978–989.

[19] Logmeln Inc. 2021. Lastpass - Password Manager & Vault App. https://www.lastpass.com/.

[20] Ahmed Tanvir Mahdad, Mohammed Jubur, and Nitesh Saxena. 2021. Analyzing the Security of OTP 2FA in the Face of Malicious Terminals. In *Information and Communications Security: 23rd International Conference, ICICS 2021, Chongqing, China, November 19-21, 2021, Proceedings, Part I 23*. Springer, 97–115.

[21] MalwareBytes. 2021. MalwareBytes Cybersecurity for Home and Business. https://www.malwarebytes.com/.

[22] McAfee. 2021. McAfee Total Protection. https://www.mcafee.com/en-us/antivirus/free.html.

[23] Microsoft. 2021. Windows 10 Security, Windows Defender Antivirus, Windows Defender Security Centre. https://www.microsoft.com/en-us/windows/comprehensive-security.

[24] NTT Security. 2019. Global Threat Intelligence Report. https://us.nttdata.com/en/-/media/assets/reports/digital-global-threat-intelligence-report-2019.pdf.

[25] Borke Obada-Obieh, Yue Huang, and Konstantin Beznosov. 2020. The burden of ending online account sharing. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[26] PhantomJS Contributors. 2010-2018. PhantomJS- Scriptable Headless Browser. https://phantomjs.org.

[27] PingIdentity. 2023. Identity Security for digital Enterprise | Ping ID. https://www.pingidentity.com/en.html.

[28] Jay Prakash, Clarice Chua Qing Yu, Tanvi Ravindra Thombre, Andrei Bytes, Mohammed Jubur, Nitesh Saxena, Lucienne Blessing, Jianying Zhou, and Tony QS Quek. 2021. Countering Concurrent Login Attacks in "Just Tap" Push-based Authentication: A Redesign and Usability Evaluations. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 21–36.

[29] Ken Reese, Trevor Smith, Jonathan Dutson, Jonathan Armknecht, Jacob Cameron, and Kent Seamons. 2019. A usability study of five two-factor authentication methods. In *Proceedings of the Fifteenth Symposium on Usable Privacy and Security*.

[30] Software Freedom Conservancy- Selenium Project. 2021. Selenium WebDriver. https://www.selenium.dev/projects/.

[31] Sonicwall.com. 2021. Sonicwall Cyber Threat Report. https://www.sonicwall.com/medialibrary/en/white-paper/2021-cyber-threat-report.pdf.

[32] Sophos Ltd. 2021. Sophos Home - Cybersecurity made simple. https://home.sophos.com/en-us.aspx.

[33] Elizabeth Stobert and Robert Biddle. 2014. The password life cycle: user behaviour in managing passwords. In *10th Symposium On Usable Privacy and Security ({SOUPS} 2014)*. 243–255.

[34] Viktor Taneski, Marjan Heričko, and Boštjan Brumen. 2014. Password security—No change in 35 years?. In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 1360–1365.

[35] The Daily Swig. 2022. MFA fatigue attacks: Users tricked into allowing device access due to overload of push notifications. https://portswigger.net/daily-swig/mfa-fatigue-attacks-users-tricked-into-allowing-device-access-due-to-overload-of-push-notifications.

[36] The Hacker News. 2022. Hackers increasingly using Browser Automation Frameworks For Malicious Activities. https://thehackernews.com/2022/05/hackers-increasingly-using-browser.html.

[37] The Secret Security Wiki - Secret Double Octopus. 2021. How does Passwordless Authentication works? https://doubleoctopus.com/security-wiki/authentication/passwordless-authentication/.

[38] ThreatPost. 2021. 500 Malicious Chrome Extensions Impact Millions of Users. https://threatpost.com/500-malicious-chrome-extensions-millions/152918/.

[39] Tom Olzak. 2021. Malicious Browser Extensions: Why They Could Be the Next Big Cybersecurity Headache. https://www.spiceworks.com/it-security/vulnerability-management/articles/malicious-browser-extensions/.
[40] Virustotal. 2022. Virustoal-Home. https://www.virustotal.com/gui/home/upload.
[41] Zhi Xu and Sencun Zhu. 2012. Abusing Notification Services on Smartphones for Phishing and Spamming.. In *WOOT*. 1–11.
[42] Zdnet, a Red Ventures Company. 2020. Three million users installed 28 malicious Chrome or Edge extensions. https://www.zdnet.com/article/beef-up-microsoft-edge-with-my-favorite-add-ons/.
[43] Zdnet, a Red Ventures Company. 2022. Malicious Google Chrome extensions affect 1.4 million users. https://www.zdnet.com/article/malicious-google-chrome-extensions-affect-1-4-million-users/.