Bypassing Push-based Second Factor and Passwordless Authentication with Human-Indistinguishable Notifications

Mohammed Jubur
University of Alabama at Birmingham
mjabour@uab.edu

Prakash Shrestha*
Equifax Inc.
prakash.shrestha@equifax.com

Nitesh Saxena
University of Alabama at Birmingham
saxena@uab.edu

Jay Prakash
Singapore University of Technology and Design
jay_prakash@mymail.sutd.edu.sg

ABSTRACT

Second factor (2FA) or passwordless authentication based on notifications pushed to a user's personal device (e.g., a phone) that the user can simply approve (or deny) has become widely popular due to its convenience. In this paper, we show that the effortlessness of this approach gives rise to a fundamental design vulnerability. The vulnerability stems from the fact that the notification, as shown to the user, is *not uniquely bound* to the user's login session running through the browser, and thus if two notifications are sent around the same time (one for the user's session and one for an attacker's session), the user may not be able to distinguish between the two, likely ending up accepting the notification of the attacker's session.

Exploiting this vulnerability, we present HIENA[1], a simple yet devastating attack against such "one-push" 2FA or passwordless schemes, which can allow a malicious actor to login soon after the victim user attempts to login triggering multiple near-concurrent notifications that seem indistinguishable to the user. To further deceive the user into accepting the attacker-triggered notification, HIENA can optionally spoof/mimic the victim's client machine information (e.g., the city from which the victim logs in, by being in the same city) and even issue other third-party notifications (e.g., email or social media) for obfuscation purposes. In case of 2FA schemes, we assume that the attacker knows the victim's password (e.g., obtained via breached password databases), a standard methodology to evaluate the security of any 2FA scheme. To evaluate the effectiveness of HIENA, we carefully designed and ran a human factors lab study where we tested benign and adversarial settings mimicking the user interface designs of well-known one-push 2FA and passwordless schemes. Our results show that users are prone to accepting attacker's notification in HIENA with high rates, about 83% overall and about 99% upon using spoofed information, which is almost similar to the rates of acceptance of benign login sessions. Even for the non-spoofed sessions (our primary attack), the attack success rates are about 68%, which go up to about 90-97% if the attack attempt is repeated 2-3 times. While we did not see a statistically significant effect of using third-party notifications on attack success rate, in real-life, the use of such obfuscation can be quite effective as users may only see one single 2FA notification (corresponding to attacker's session) on top of the notifications list which is most likely to be accepted.

We have verified that many widely deployed one-push 2FA schemes (e.g., *Duo Push*, *Authy OneTouch*, *LastPass*, *Facebook's* and *OpenOTP*) seem directly vulnerable to our attack.

ACM Reference Format:
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 (ASIA CCS '21), June 7–11, 2021, Virtual Event, Hong Kong*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3433210.3453084

1 INTRODUCTION

Given the rise in the Internet connectivity of mobile devices (such as smartphones), a new form of second factor authentication (2FA) has rapidly emerged. In this approach, referred to as *One-Push-2FA* or just *Push-2FA*, the possession of the second factor device during an authentication attempt to a remote website is verified by sending a push notification to the device running a pre-installed 2FA app and asking the user to approve (or deny) that they indeed intend to authenticate by simply tapping a button on the phone. The notification may also contain the information regarding the client machine the user is authenticating from (such as IP geolocation) (Figure 1). If the user does not see the notification, it will timeout and the authentication attempt is denied by default. This approach is also being widely deployed in a single factor mode, referred to as *passwordless* login, where authentication is solely based on approval of push notifications, without any need for a password.

Compared to the traditional, one-time PIN (OTP) based approach (e.g., Google 2SV [16]), Push-2FA provides a significant reduction in the amount of user effort and cognitive burden since there is no longer a need to copy a random passcode from the device to the login terminal in each login attempt. Thanks to this notable usability enhancement, Push-2FA has seen large scale deployments at both academic universities[2] and commercial online businesses, some of the current major Push-2FA services being: *Duo Push* [11], *LastPass Authenticator* [17], *Authy OneTouch* [2], *Facebook's* [13], *OpenOTP* [32], *RSA Secure ID* [34] and *PingMe* [20].[3] Duo Push is perhaps the most widely adopted scheme currently. Indeed, a recent

*Work done at University of Alabama at Birmingham
[1]HIENA denotes "Human-IndistinguishablE Notification Attack"

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ASIA CCS '21, June 7–11, 2021, Virtual Event, Hong Kong
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8287-8/21/06…$15.00
https://doi.org/10.1145/3433210.3453084

[2]For example, CMU, University of Texas (several campuses), UC Berkeley, Penn State, and University of Iowa, have deployed Duo Push.
[3]Alongside Push-2FA, most of these services provide the option of one-time passcode based 2FA.

study of the deployment of Duo Push at CMU [8] noted that users generally found it to be easy to use, helping them better adopt 2FA.

Besides usability, the Push-2FA approach is generally marketed and perceived to be highly secure. For example, Duo [11] advertises that their approach provides *"secure access"* and is *"resilient against man-in-the-middle attacks that allow attackers to steal your password and your second factor"*. Similarly, LastPass [17] claims that, when using their instance of Push-2FA, *"in the rare event that someone managed to steal or hack your password, you would still be protected."* PingMe and Authy suggest that their schemes can be used for "passwordless login" and "in place of a password" while still claiming to be *"highly secure"* and offering *"strongest authentication"*. Users of these systems also seem to perceive that they help improve the security of their accounts [8]. These security-related claims and perceptions are not unfounded. Indeed, a naive attacker, in possession of a victim user's username/password credentials, would not generally succeed in logging into the user's account at a *random point of time* since the login notification is most likely either going to be rejected by the user (as the *user did not intend to login*), or ignored by the user and rejected by default (as the user may not attend to that notification due to phone being out of reach, or in pocket or purse, etc.).

In this paper, we set out to pursue a deeper understanding of the security of Push-2FA and passwordless schemes. Unfortunately, we find that the effortlessness underlying this approach comes at a price as it hides a fundamental design vulnerability, which we refer to as *human-indistinguishable notifications*. The vulnerability stems from the fact that the Push-2FA generated notification, *as shown to the user*, is *not uniquely bound* to the user's login session, and thus if two or more notifications are sent around the same time (first corresponding to the user's login session and others corresponding to a session of the attacker attempting to login on behalf of the user), the user may get "lost", not being able to tell these notifications apart and would likely accept the notification corresponding to the attacker's session (attacker notifications would appear on top of the user notification on the interface). Once the attacker notification has been accepted, the attacker will be able to login and have full access to the user's online account. The user may not succeed in logging in during the process and attribute this to a minor glitch in the system (where the login somehow did not work) and would simply re-attempt to login, thereby the attack may remain oblivious to the user and the web service.

Based on the above vulnerability, we present HIENA (Figure 2), a potentially devastating attack against Push-2FA (including passwordless login), which can allow a malicious entity to login soon after the victim user triggering multiple near-concurrent human-indistinguishable notifications. Our **primary attack** *does not* require any spoofing capabilities. However, *optionally*, to further fool the user into accepting the notification corresponding to the attacker's login session, the attacker can spoof the victim's client machine information (e.g., IP address and location), and even generate other third-party notifications (e.g., email, social media or SMS) for obfuscation purposes such that the notification corresponding to the attacker's session is the only visible Push-2FA notification, most likely on top of the interface. For spoofing the client information, the attacker can just be in the same city as the victim since the login notifications usually show the city information. Moreover, unlike a typical phishing attack, HIENA ***does not require the attacker to impersonate the web service and fool the user into clicking a link via the phishing email [31].***

Overall, ***the attack works under the standard 2FA threat model, which is to provide security even when the username/password has been compromised,*** **and** ***without the need for learning passwords when the system is used in a passwordless manner***. It only additionally needs to know roughly when the user is attempting to login. This information can be readily obtained in a number of ways, such as by monitoring the user's browsing sessions even for the sites that use the SSL/TLS protocol [3] such as by deploying remote website fingerprinting techniques [38], using contextual information as to when the users are most likely to login (e.g., when the University reopens after a break, peak shopping times during holidays, paper/assignment submission deadlines, etc.), being in the physical proximity to the victim user (e.g., a disgruntled colleague or a roommate as the attacker) to observe when the user logs on to a particular site, or asking the user into logging in via social engineering trickeries such as technical support scams [9, 21], just to test if the login works during a troubleshooting session. While the security of other authentication schemes (e.g., passwords or one-time passcode based 2FA) ***does not breakdown*** with the attacker knowing the ***time*** at which the user logs in, in light of our attack, the Push-2FA schemes seem to have a unique weakness. In fact, we believe that the security of an authentication system *should not rely upon hiding such timing information from the attacker* (which is more of a privacy property).

To evaluate the effectiveness of our attack, i.e., to measure the level of users' susceptibility to human-indistinguishable notifications, we carefully designed and ran a human factors lab study where we tested benign and adversarial settings. Our results confirm our hypothesis and demonstrate that HIENA is highly successful in defeating the security of Push-2FA.

Our Contributions: Our work provides the following contributions to the field of web authentication:

(1) ***A New Vulnerability based on Human-Indistinguishable Notifications:*** We identify a fundamental vulnerability of Push-2FA and passwordless login systems arising from the human-indistinguishability of near-simultaneous notifications corresponding to a benign login session triggered by the user and malicious login sessions triggered by the attacker. It is our understanding that, to keep the login process near-effortless for the user, Push-2FA schemes cited above do not provide to the user a binding between the notification and the actual login session on the browser, and therefore the user may not be able to tell if he is accepting his own login session or the attacker's login session.

(2) ***Design of a Concrete Attack Instance:*** Based on the notion of human-indistinguishable notifications, we build HIENA, a concrete Push-2FA attack. HIENA is designed to maximize the chances of fooling the user into accepting the attacker's login notification amidst the user login notification. To do so, it can spoof the user's client machine information (*Spoof-Notif*) and trigger other third-party notifications (*Other-Notif*). The attacker can generate *Other-Notif* notifications on the victim

device in different ways, like sending spam emails or text messages, or performing specific actions to the victim social media account (e.g., posting comments, "liking", "following", etc.). The attacker can easily gain this information either through public profiles in the social media or leaked databases [7, 10, 28].

(3) ***Evaluation via a Human-Factors Study:*** We design and implement a small-scale lab-based user study to evaluate the effectiveness of our attack. A small scale experiment is sufficient here since we are assessing attack feasibility, not measuring usability. Each participant's task in the study is to perform an emulated login process to an online account (Gmail) through our implementation of the Push-2FA service. As part of accomplishing this task, they need to respond to the push notifications generated on the phone, in order to prove the possession of the second factor device (the phone). Our design of the push notifications mimicks the ones widely used by deployed Push-2FA systems. We also incorporate two UI design choices for the push notification – "Approve" option on the left side of the notification interface (*Left-UI*) (Figure 8c), or on the right side (*Right-UI*) (Figure 8d). Further, our study carefully integrates *real-time implementation* of benign and HIENA's adversarial login settings mentioned above.

Summary of Key Results: Our results suggest that users are highly prone to accepting the notifications triggered by the HIENA attacker with high rates. Specifically, the average success rate of the attack is 83% overall, and about 99% with *Spoof-Notif*, which is almost similar to the rates of acceptance of benign login sessions (where no HIENA attack takes place). Even ***without spoofing, which is our primary attack, the attack success rates are fairly high***, about 68%, and if this attack is executed twice or thrice, the attack success rate can reach as high as about 90% or 97%, respectively, assuming all attack trials are equally likely to succeed. While we did not observe a statistically significant effect of using third-party notifications in *Other-Notif* on the attack success rate, in real-life, the use of such notifications can be quite effective as users may only see one single login notification (corresponding to the attacker's session) on top of the interface which is more likely to be accepted by the users. We confirmed that our participants were not simply clicking-through, but were often responding from the UI on a large number of occasions and with due diligence (Section 5.3). In real-life, the attack success rates may be even higher since users are often habituated and rushed to accept the notifications when they are authenticating without diligently evaluating them. Some currently deployed services (e.g., LastPass, see below) do not even provide any login information to the user. Furthermore, a typical average user of Push-2FA or passwordless login may be more susceptible to the attack in contrast to our participants who were generally young, educated and technologically-oriented.

Many widely used Push-2FA schemes we tested (e.g., Duo Push, Authy OneTouch, LastPass, Facebook and OpenOTP) and any of their deployments in universities and personal/commercial settings are directly vulnerable to our attack. We tested our attack against Android phones, but since the iOS and Microsoft phones do not have noticeable differences from Android in the UIs of phone apps of these Push-2FA schemes, users of all of these phones seem vulnerable to our attack.

2 BACKGROUND AND MODELS

2.1 System Model

Push-2FA is a system for authenticating the user to a remote web (or online) service. For simplicity, we use the term *service* to denote a remote web service deploying Push-2FA to authenticate the user. Push-2FA requires the user to install a software app on his smartphone. The user links the installed app and his account associated with service. Multiple accounts with different services can be linked with a single software app such that the single app can be used with multiple services. In Push-2FA, the user's credentials (i.e., his username and password) serves as the first authentication factor while the software app installed on the user's phone serves as the second factor for authentication. The addition of the second authentication factor should improve the security compared to that of the password-only authentication as it may seem that the adversary now needs to compromise the second authentication factor along with the first factor to hack into the user's account.

In order to login to a Push-2FA enabled service, the user first provides his credentials to the service (unless the passwordless mode is used). The service then sends a login prompt, also called push notification, to the user's phone. The purpose of this push notification is to confirm from the user whether he has tried to login, or somebody else has tried to login on behalf of the user. To successfully login to the service, the user needs to approve the push notification by tapping the 'Approve' button. Figure 1 shows the flow diagram of Push-2FA login in the benign setting. Unlike traditional SMS or mobile passcode based 2FA, Push-2FA does not require the user to copy the passcode from the phone to the authentication terminal, mere tapping of a button on the phone is sufficient to login. This makes the Push-2FA scheme more usable compared to the traditional passcode based 2FA scheme. In this study, we investigate if this improved usability has any adverse effect on the security of the system.

To further improve the security of the Push-2FA system, the push notification may show various information about the current login attempt. For instance, the approximate location information (e.g., City, State, Country) from where login has been issued, and the IP address of the machine that is used for login. We refer to such information about the login as *login information*. Before approving the login prompt, the user can open the push notification to see these login details. If the login information shown on the notification is not valid (or seem fraudulent), the user can deny the login attempt and/or report the incident to the service provider. Push-2FA also defines a *"time-out"* period, the duration within which the user needs to respond to the push notification. If the user fails to do so for whatever reason, the login attempt is denied by default.

Some of the Push-2FA schemes, e.g., Microsoft Authenticator [30], Identity Automation [19], and Authy [42], do not require the user to enter passwords, mere approval on the login prompt is sufficient to login, and is referred as *passwordless Push-2FA*. Specifically, in passwordless Push-2FA, the user first provides his username to the service, which in turn sends a login prompt to the user's phone. The user then needs to approve the push notification by tapping the 'Approve' button.

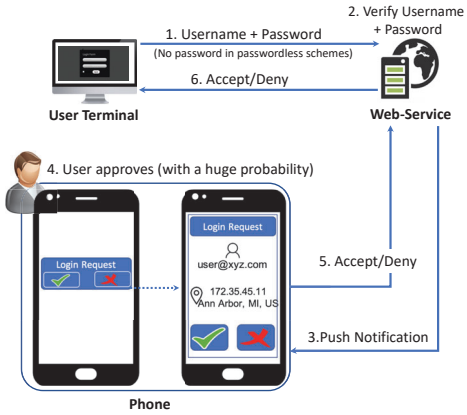


Figure 1: Normal operation of Push-2FA. At login, the web service sends a push notification to the user's phone. The user approves the push notification and succeeds to login to the web service.

2.2 Adversarial Model

Just like any other two factor authentication scheme, Push-2FA threat model assumes an adversary who has compromised the victim user's first authentication factor, i.e., his credentials (username and password), and attempts to log into the user's service. The primary goal of Push-2FA is to defeat such an adversary. The adversary can learn the information about victim's credentials, for example, via leaked password databases of the web-service, phishing attacks or other mechanisms [18, 25, 26, 39]. With this information, the adversary attempts to authenticate to the service on behalf of the victim. Obviously, to hack a passwordless Push-2FA scheme, the attacker does not need to learn the user's password. The adversary succeeds to hack into the victim's service if he can prove the possession of the second factor device, which is possible only when the push notification triggered by the adversary's login attempt is approved. Since the adversary-triggered push notification is usually generated at a random point in time when the *user has not tried to login*, it alerts the user that someone has tried to access his service. In such a situation, the user is likely to either deny the push notification and/or report the incident to the respective service provider, thereby preventing the attack (or the login attempt is denied by default after a time-out). The threat model of Push-2FA assumes that the adversary has not compromised, or gained physical access to, the user's second factor device, i.e., his phone. If the adversary gains control of the victim's device, the security of Push-2FA (including passwordless variant) obviously reduces to that of the password-only (or the single-factor) authentication .

In our threat model, we assume that the HIENA attacker can closely approximate the time instance at which the victim user is logging in. This can be accomplished in a number of ways, some of which are outlined below:

(1) ***Traffic Monitoring and Remote Website Fingerprinting***: The attacker can learn about the user logging time by monitoring the user's browsing session even through SSL/TLS [3]. In particular, a motivated attacker may use website fingerprinting approaches to learn when the user visits a given website. The recently proposed website fingerprinting attack [35] is highly relevant as the attacker can perform the website fingerprinting remotely. It is also possible to monitor the incoming traffic at the user's end to fingerprint push notifications [29]. Such a targeted attack can be launched, for example, by intelligence and law enforcement agencies and nation-state attackers. Governments and ISPs in the oppressive regimes already monitor users' Internet traffic on a routine basis and may use this approach to compromise their Push-2FA accounts[4].

(2) ***Utilizing Contextual Information and Login Patterns***: The attacker can launch the attack at a time when most of the users are most likely to login. For instance, users' login activity may be at peak when the organization opens after a break. As a concrete example, we evaluated the login data, i.e., time of login across respective dates for a year (August 14, 2019 to August 13, 2020), of 26,144 unique users in a university. Our analysis corroborates the exploitation of contextual information for estimating timeframes which would have high probabilities of success for our attack. Interestingly, January 13 is the start of the semester and many students and staff would return to the campus and login to university mails and services for initiating academic obligations, registering courses, accessing course information etc. Appendix Figures 5 reveal this phenomenon. There is a sudden surge in the number of login attempts in between January 10-13, while January 13 witnesses a clear peak. If the attacker launches the proposed attack at peak hours on January 13, there is a high chance of multiple accounts getting compromised. In the same vein, when grades are published at the University (at a known, set time), students are most likely to login as soon as grades are published to check on their grades. Similarly, many users are most likely to use and login to online shopping sites during the peak shopping season when a great amount of shopping deals are made available, e.g., the Amazon shopping site is highly used on Cyber Monday [43]. The users may login to and are mostly active on social media at a particular time in a day of the week, e.g., the users are mostly active on the Facebook on Wednesday at 11:00am and 1:00pm [1]. Also, many users are currently using various video conferencing tools like Zoom for online classes and virtual meetings, where they usually login to the service before the event starts at a set time. The attack can be launched against many users at such a peak time, with the hope that a large fraction of these users will indeed login at this time.

(3) ***Active Social Engineering***: The attacker can force the user to login by launching social engineering tricks, such as password reset scams, where users are informed that their accounts have been compromised, or their credentials have been expired, and asked to reset their passwords by logging into their accounts. For example, the attacker can employ the approach similar to that of the tech support scam calls[9, 21], where the attacker makes the call to victim users claiming to be a customer service agent in order to address login related issues, such as password reset, and ask the user to go through the login process while speaking to the victim in real-time and troubleshoot the login. The attacker can execute the HIENA attack at the time the user attempts the login during this scam call.

(4) ***Compromised End Points:*** While the threat model of two factor authentication does not assume that the user's phone has been compromised or malicious, there are other entities

[4] As an example, Chinese government has previously attempted to compromise the Gmail accounts of Chinese human-rights activists [22]

that could be compromised or are malicious in this model. For example, the attacker can compromise the victim user's login terminal (laptop or desktop PC) by deploying malware to learn (the login credentials and) the login time information of the victim user. Such threat models are already considered effective in the purview of 2FA schemes [6]. In another scenario, the website/webserver itself that the user is trying to access could be malicious or it could be infected with malware. The malware can be crafted cleverly that at the time of login, it stealthily extracts (the victim's credentials and) login time, and send them to the attacker. Once this information is received, the attacker can execute the HIENA at nearly the same time. Similarly, a password manager that employs Push-2FA (e.g., LastPass) for securing their passwords to web services could be another end point that is malicious itself or be infected with a malicious script. It important to note that traditional 2FA schemes (e.g., PIN-2FA) in these compromised end-point scenarios may still remain secure, or at least can not be compromised just by the added knowledge of the user's login time instance.

(5) ***Observation from Physical Proximity***: The attacker could reside in close physical proximity of the user (e.g., a disgruntled colleague, or a roommate, or a stranger peeping through in a public cafe) and can learn when the user logs on to a particular site using Push-2FA via physical observation. We assume that the attacker is nearby but does not still have physical access to the user's phone.

The attacker can optionally utilize IP address and physical address information to spoof the user's machine. Thus, the attacker would be able to generate push notification(s) that look similar to, or exact copy of, the one from the legitimate user's machine. It may actually not be necessary to spoof the IP addresses, merely spoofing the general location information shown on the notifications (e.g., the city) or being in the same city as the victim could be sufficient to succeed at the attack. Further, users do not pay much attention to such information (or know the IP addresses of their machines), so even without any spoofing, the attack could succeed. The attacker can also use phone numbers and/or email address information to trigger third-party, text and/or email based notifications. The attacker can easily gain this information either through public profiles in the social media or leaked databases [7, 10, 28]. Further, since many users often use the same usernames across multiple apps, the attacker can use the username information to trigger notifications using these apps that use the same username. For example, if a user has the same username in the leaked database server and on Twitter, the attacker can send a notification (e.g., a follow request, tweets/retweets) to user's Twitter account.

3 OUR ATTACK

The threat model of HIENA assumes that the attacker has already obtained the victim's password (not needed for passwordless login), and he knows, or makes a good estimate of, the time when the victim user would attempt to login. With this knowledge, the attacker tries to login soon after, or around the same time, the victim tries to login. To verify the possession of the second factor device (i.e., the victim's phone) for "complete" authentication, the web service sends two push notifications to the victim's phone, *UN* and *AN*, corresponding to the victim's and the attacker's login attempts, resp.

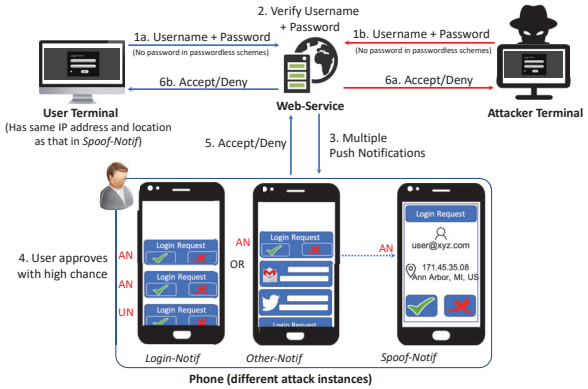


Figure 2: HIENA against Push-2FA. The attacker logs in around the same time, or soon after, the user logs in generating multiple push notifications, UN and ANs, on the user's phone. To fool the user into accepting attacker-triggered push notification, the attacker can spoof the user's login information (e.g, IP address and location) and produce other third-party notifications (e.g., email or social media) for obfuscation purposes. The user remains unaware of the attack as he can also succeed to login when he approves next push notification UN, or otherwise retries to logging in.

The victim, being unaware of the ongoing attack, would approve either *AN*, *UN*, or both as he intends to login and thinks that approving notifications would allow him to login. If the attacker login attempt is made after the victim login, typically *AN* would show up on the top of the notification bar and *UN* would reside below it. It is highly likely that the victim would review (or accept) notifications starting from the most recent one (i.e., following a top-down order, as also confirmed in our study later). The attacker succeeds if the victim approves *AN*. As approving *AN* does not help the victim to log into his service, the victim is likely to also approve the next notification (i.e., *UN*) that he received, or he may just ignore the second notification and rather retry logging into the service. Since approving *UN* or second login attempt allows the user to login into his service, he would think that it was a minor glitch in the system that causes this issue. Thus, the attacker succeeds at logging into the victim's service while the victim remains unaware of the attack. Figure 2 illustrates HIENA.

To further deceive the user into accepting the attacker-triggered push notification, the attacker can optionally spoof the victim's login information. Based on the level of matching of the attacker's and the user's login information, we categorize our attack into two types:

- ***Spoof-Notif Attack:*** Users often login using the same machine(s) (e.g, personal or particular machine assigned to him) from the same general location (e.g, home, office, etc.). Given this, the login information of the user that the attacker has learned and spoofed is highly likely to match with that of the user. We refer to such an attack scenario where the spoofed and actual login information match as *Spoof-Notif*. In such an attack setting, the user would get fooled into believing that the multiple push notifications are triggered due to his own login attempt and approve *AN*. The user may think that the multiple push notifications are generated potentially due to some glitch in the system. Such a glitch is often accepted by general user population.
- ***Non-Spoof-Notif Attack:*** We refer to the attack scenario where the attacker's and victim's login information do not at all match as *Non-Spoof-Notif*. However, many of the users do not pay much attention to the login information shown on the push notification during logins (as found in our study presented in Section 5).

Further, they are habituated to approving such notifications (click-through behavior and habituation is already well-highlighted in user-centered security literature, including brain studies, e.g., [14, 40, 44]). This may result in the user approving push notification triggered by the attacker and enables the attacker to break the Push-2FA system even with a completely different login information.

Intermediary scenarios also exist whereby the attacker is not able to fully spoof the victim's login information, but can successfully match up some crucial parameters. For example, spoofing IP addresses may be harder in some cases, while spoofing a higher level location information is not that hard (or the attacker can login from machines located in the same general location or the city). It may be hard for average users to understand the IP addresses and to detect whether the IP address shown on the notification is indeed their own machine's IP address and therefore if the higher level location information is matched, it is very likely users would accept such information. Thus, partial spoofing is very likely to succeed even if the users were paying attention to the login information. Further, some Push-2FA schemes do not show much of the login information to the user. For instance, LastPass does not show any login information (Appendix Figure 9f) to the user. To defeat Push-2FA schemes with such a design, HIENA does not need to spoof the login information since the notification triggered by the attacker login looks similar to that triggered by the user login.

The HIENA threat model assumes that the attacker has obtained information about the user's accounts associated with different phone apps. Given this knowledge, the attacker can trigger different third-party notifications on the user's phone during the attack. The notifications triggered by different apps may or may not reach the targetted device (or on time) while launching the HIENA attack. Based on this, we divide our attacks into the following two types.

- ***Other-Notif* Attack**: In this attack, the attacker utilizes the user's different accounts associated with different phone apps to trigger various notifications on the user's phone when launching the attack. These app notifications may obscure the receipt of multiple push notifications during the attack and further prevent the potential user suspicion raised from them.
- ***Login-Notif* Attack:** In this attack, either the leaked user's apps account information is not valid, or the triggered notifications do not reach the targeted phone device. In such a scenario, only multiple push notifications are triggered on the user's phone due to multiple login attempts to the same user's account.

Likelihood of Accepting Attacker's Push Notification: To summarize, with above-mentioned attack settings, when launching HIENA against Push-2FA, it is highly likely that the victim user will accept the attacker's push notification mixed with user notification (with varying level of probabilities for each attack setting) for several reasons. First, the user is intending to authenticate to the service and the user is likely to believe that the push notification(s) that he received are generated from his own login attempt. Second, since attacker's multiple attempts are made after the victim has tried to login, the attacker's notifications would likely reside at the top of the notification bar and the user is likely to start accepting from the top notification. Third, *Other-Notif* can potentially result in the user seeing only one push notification, likely from the attacker's attempt, due to the limited screen area. In such an attack setting, the third-party apps' notifications would make it harder for the user to detect the attack simply based on the presence of multiple push notifications. Lastly, the users are generally habituated to accept security prompts or warnings (e.g., [14, 40]) that may result in accepting the attacker's login attempt, especially when they are made around the time when the user is trying to login and when using the *Spoof-Notif* setting (including partial spoofing).

Real-World Push-2FA Systems Vulnerable to HIENA: We ran an experiment, besides the one in Section 4, to test if widely deployed Push-2FA systems are vulnerable to HIENA. We created a personal account using several Push-2FA systems, Duo-Push [11], Authy OneTouch [2], LastPass [17], Facebook [13], and OpenOTP [32], registered a smartphone to service, and observed if the system allows multiple near-concurrent logins and sends multiple notifications to the smartphone. Specifically, we tried to login from different browsers (e.g., Mozilla Firefox and Google Chrome) at a similar time from the same machine. We also logged in from two different machines around the same time. We observed all these Push-2FA systems allow near-concurrent login attempts and are therefore vulnerable to HIENA.

LastPass Authenticator is used to authenticate LastPass users to the LastPass password management service, which stores users' passwords to access many of their other online accounts. By compromising LastPass Authenticator using HIENA, the attacker would, therefore, be able to gain access to the user's all online accounts locked with LastPass. To make the matters worse, Lastpass push notifications are very high level and do not contain any information related to login details, like where the login attempt was generated from (see Appendix Figure 9f). LastPass Authenticator [27] can also be used as a Push-2FA service for other online services/applications to authenticate their users similar to Duo Push. Appendix Figure 9 provides the snapshots of the (phone and browser) UIs of these schemes from our test runs.

4 ATTACK EVALUATION STUDY DESIGN

4.1 Implementation

We implemented an instance of Push-2FA that emulates many of the real-world Push-2FA (including passwordless login) systems. Specifically, we implemented the following two main components in our Push-2FA implementation:

(1) ***WebService and Browser Apps:*** We implemented two apps, *WebService-App* and *Browser-App*, using HTML, JavaScript, CSS, and PHP. WebService-App runs on a remote web server and Browser-App runs on a client machine. Browser-App is a simple web page with two forms – a sign-up form and a login form. The user can register to our test implementation of Push-2FA using the sign-up form. To store the user's account information, we employed MySQL as the backend database. Similarly, the user can use the login form to log into his account (created earlier) with our Push-2FA system. Thus, our implementation simulates the user registration and authentication processes of any standard 2FA scheme.

WebService-App verifies the validity of the user's credentials and triggers a push notification (embedding the login information received from WebService-App) on Phone-App for login approval. On receiving the user's response ("Approve" or "Deny"), WebService-App forwards it to the Browser-App. If the

Table 1: Brief description of various terms introduced in our study.

Terms	Description
UN	User Notification. Push notification triggered by the user login.
AN	Attacker Notification. Push notification triggered by the attacker login.
Input-PopUp	Input options (Approve/Deny) provided on the notification. User can also tap the notification to see login information and provide response.
Plain-PopUp	Input options not provided on the notification. The user must tap the notification to see login information from and provide response via UI.
Left-UI	Approve button is positioned on the left side of the UI layout.
Right-UI	Approve button is positioned on the right side of the UI layout.
Spoof-Notif	Attack scenario where login information, i.e., victim's client machine information such as IP address and location, is spoofed. ANs and UN show exact same login information.
Non-Spoof-Notif	Attack scenario where login information is not spoofed. ANs and UN show completely different login information.
Login-Notif	Attack scenario where the victim user receives multiple ANs + 1 UN.
Other-Notif	Attack scenario where the victim receives attacker triggered third-party apps' notifications along with multiple ANs + 1UN.

user approves the login request, Browser-App shows a success message ("Login Successful!"), and redirects it to a Gmail login page (to simulate a login attempt to Gmail via our Push-2FA service). In case the user denies the login request, *Browser-App* shows an error message ("Login Denied") and redirects back to the login form of our Push-2FA service. We employed Firebase Cloud Messaging (FCM) to establish a communication channel between the web-service and Android phone-app.

(2) ***Phone-App:*** We developed an Android app for the phone that remains idle in the background to emulate the Push-2FA app. To use our Push-2FA system, the user first links his account created earlier using WebService-App with Phone-App. When the user attempts to login through Browser-App, Phone-App receives the login information from WebService-App and generates a push notification asking the user for login confirmation.

Push Notification Design Choices: We implemented two types of designs – *Input-PopUp* and *Plain-PopUp*, that cover majority of the push notification designs used by widely deployed Push-2FA systems. In *Input-PopUp*, the user is given the option to provide his response (Approve or Deny) directly from the notification itself without looking into the login information, or he can tap on the notification, verify the login information, and then provide the confirmation. Various Push-2FA systems, such as Duo Push and PingMe, employ such design. In *Plain-PopUp*, the user has to tap on the notification to see the login information and provide his response. Unlike *Input-PopUp* (Appendix Figure 8b), *Plain-PopUp* (Appendix Figure 8a) does not feature the option for the user to provide his response from the notification. Many Push-2FA systems follow such design, such as OpenOTP and LastPass. Further, we implemented two different interface layouts of push notification based on the placement of Approve and Deny buttons, namely *Left-UI* (Appendix Figure 8c) and *Right-UI* (Appendix Figure 8d). *Left-UI* has "Approve" button on the left side of the notification interface and "Deny" button on its right side, while *Right-UI* has the Approve-Deny buttons in the reverse order. Push-2FA systems like Duo Push, Authy, Okta, and Gluu follow *Left-UI* layout while RSA, Gemini, PingMe, and Futurae follow *Right-UI*. Thus, our implementation of Push-2FA is inline with many of the widely used Push-2FA systems (as noted earlier) in terms of interface, notification design, position of approve/deny buttons, etc. This shows that our experiment simulates the real-world Push-2FA system well. Table 1 summarizes various terms used in our study.

We note that our study is not designed to evaluate the time-sensitivity of the attack; rather, we assume that the attacker logs in soon after the user (maybe within a few seconds, assuming push notifications themselves introduce some communication delay). Many scenarios exist where the attacker can successfully login soon after the user (Section 2.2). In case the attacker's login time is significantly off of the user's login time, the attack may fail but the user may still ignore the attacker's notification, and the attack can always be repeated after a sufficient period (to minimize user suspicion) and may succeed then.

4.2 Login Sessions

To evaluate the performance of our participants with respect to our Push-2FA implementation in the benign setting and that of HIENA in various attack settings (Section 3), we incorporated and implemented two types of login sessions in our study.

- **Benign Session:** Our Push-2FA triggers only one push notification, simulating a real-world login scenario where the user attempts to login using Push-2FA.
- **Attack Session:** Attack session in our implementation of Push-2FA incorporates various attack settings. To simulate *Login-Notif* attack scenario, WebService-App triggers 3 push notifications on the phone (Appendix Figure 8e). In HIENA, as the attacker logs in after the victim user has logged in, the first notification on the phone (at the bottom of the notification bar) corresponds to the victim's login session while remaining notifications (at the top of the notification bar) correspond to attacker's login sessions. To simulate the *Other-Notif* attack scenario, 2 third-party apps' notifications are generated by our Phone-App in addition to 3 push notifications (Appendix Figure 8f). The first app notification simulates a Twitter notification while the second notification simulates an email notification (i.e., Gmail notification). Phone-App positions these simulated app's notifications in between multiple push notifications. Further, to simulate the *Non-Spoof-Notif* attack scenario, we programmed Phone-App to show the actual login information of the victim triggered push notification and a random login information (i.e., a valid but completely different from victim's IP and location information) corresponding to all the attacker's push notifications. To simulate the *Spoof-Notif* attack scenario, Phone-App was programmed to show the actual login information of the victim's machine, simulating the spoofed attacker's notifications.

We programmed our Push-2FA system in such a way that each of our participants received a fixed set of benign and different attack login sessions. Specifically, each participant was subjected to a total of 32 login sessions, 16 of them were benign sessions and 16 were attack sessions. Out of 16 attack sessions, 8 correspond to the *Non-Spoof-Notif* attack setting and 8 correspond to the *Spoof-Notif* setting. Each set of these attack sessions further consist of 4 *Login-Notif* settings and 4 *Other-Notif* settings. Out of these 4 login sessions, 2 were *Input-PopUp* (both *Left-UI*) and 2 *Plain-PopUp* (1 *Left-UI*, and 1 *Right-UI*). Although all of participants performed an equal number of login sessions, the order of presentation was *randomized* to remove any learning or fatigue biases. Appendix Figure 6 shows the hierarchical flow of the various types of login sessions in our study. Our study therefore carefully integrates

real-time implementation of benign and HIENA's adversarial login sessions mentioned earlier.

4.3 Study Metrics and Hypothesis

We use following metrics to quantify the effectiveness of HIENA against Push-2FA system.

- **Benign Success Rate (*BSR*):** *BSR* measures how often the user approves the push notification in benign login sessions in our study leading to successful logins. We compute *BSR* as follows.

$$BSR = \frac{\#accepted\_benign\_sessions}{\#benign\_sessions} \quad (1)$$

- **Attack Success Rate (*ASR*):** *ASR* measures how often the user approves the push notification triggered by the attacker in an attack login session leading to successful attack. We compute *ASR* as follows.

$$ASR = \frac{\#accepted\_attack\_sessions}{\#attack\_sessions} \quad (2)$$

  When computing *ASR*, we do not consider the response of the user to victim triggered push notification in an attack session as such response does not lead to the success of the attack. Since multiple push notifications are generated during HIENA, the user may respond to (or approve) multiple push notifications with the intention to successfully log into the system. The attack succeeds if any of the notifications generated by the attacker is accepted by the user in a given attack session.

To evaluate the performance of HIENA in each of the attack settings, we compute *ASR* for each of these settings separately. When computing *ASR* of a particular attack setting, we consider only the users' responses to push notifications associated with that particular attack setting. If our attack were to succeed well against Push-2FA, which is our founding hypothesis, we would expect the *ASR* to be as close as possible to 100%.

To analyze the statistical significance of these results, we used Wilcoxon Signed-Rank Test (WSRT), at a confidence level of 95%, for measuring differences in the means of different groups underlying our analysis (e.g., *Login-Notif* and *Other-Notif*, *Non-Spoof-Notif* and *Spoof-Notif*). Bonferroni correction was used during post analysis to account for multiple comparisons. The effect size of WSRT was calculated as $r = Z/\sqrt{N}$, where Z is the value of the z-statistic and N is the number of observations on which Z is based.

4.4 Study Protocol

We recruited 20 participants from diverse educational backgrounds (detailed demographic information is presented later in Section 5.4). One of the researchers serving the role of an examiner navigated each participant to a study desk where he/she was provided with a desktop PC and an Android phone. In the desktop PC, the Google Chrome browser was pointing to our Browser-App, and the Android phone was installed with our Phone-App for the study purpose. Participants were asked to consider the study desktop PC as their personal computer that they use for accessing their online accounts, and the Android phone as their personal phone where they have installed the app for 2FA authentication. They were informed that the login process in our study simulates the real-world login setting. We explicitly asked them to imagine that they were logging into their own online service.

Our study consists of three phases . The concrete steps followed in each phase of the study are outlined below.

Phase I: Study Primer: At the beginning of the study, we provided a brief introduction and instruction about the study, and then asked participants to create an account for the study.

- *Introduction and Instructions:* As participants may not be aware of 2FA and/or Push-2FA, we explained them about 2FA and Push-2FA, and their security purpose. In fact, we ensured that the participants clearly understand what Push-2FA is, its purpose and security implications. We told them that the purpose of our study is to test the user's behavior towards Push-2FA. We intentionally did not disclose the true purpose of the study (i.e., the attack study) as it might affect the user's behavior during the study. Participants were told to assume that they were logging into their own services (e.g., email account) via our proxy service, and their goal is to complete the login step successfully. We also informed the participants about the information shown on Push-2FA notification and advised they should verify it before deciding to accept or deny login. They were also explicitly told that if they saw anything wrong during the login process, they can either "Deny" the push notification or ignore it. Further, participants were told that they can use any of the two ways of approving notifications, i.e., via notification or UI.
- *Account Creation:* We asked participants to create a new account to register to our Push-2FA system. They were allowed to use any username and password they like that is not already registered with our system. We did not use the users' actual account in our study, instead we asked the participant to consider the account they are using for the experiment as their own account (this is a typical role play setting used in many security/usability studies[24, 33, 36]). We asked participants to use "remember me" feature that allows the browser to automatically fill the user's credentials when the user visits the same website again. With this, the user does not have to remember and retype his credentials repeatedly during the study. This is a common practice often used by the users for their comfort. The users can also choose not to use this feature. Nevertheless, either way, it does not affect our study since its goal was to evaluate the effectiveness of the second factor authentication, not the first factor. Since they did not have to input their password for subsequent logins, this setting also simulates the user experience of passwordless login. Participants were then asked to link Phone-App on the phone with their accounts by signing in using the same credentials as they had used to create the account earlier. Similar to any real-world 2FA system, this step was performed only once at the beginning of the study.

Phase II: Main Study: This phase includes a practice session and the main study tasks as explained below.

- *Practice Trials:* To familiarize the participants with the Push-2FA system, they were asked to login at least six times using our implementation of Push-2FA. In a benign scenario, as one push notification is sent to the user's phone for the login approval, only one notification is sent in each of these practice trials. After responding to a push notification of a login attempt, we asked the participants to wait for a few seconds (~5 seconds) for the phone's screen to get automatically locked. During this waiting

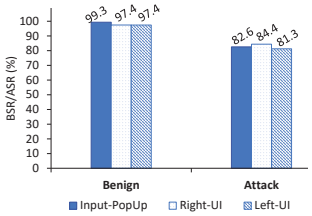


Figure 3: Performance of participants with Push-2FA in benign and attack sessions. Performance in attack sessions represents HIENA's performance.

period, Phone-App is also programmed to clear out all the notifications on the phone. The purpose of such auto-screen lock is to simulate the real-world scenario where the phone's screen generally remains locked during the login attempt, and when a new notification, potentially push notification, arrives, it alerts the user. All responses from these practice trials were discarded from our analysis as they were used only to familiarize the participants with Push-2FA. During practice trials, the participants were allowed to ask any queries they had about the study.

- *Study Task:* Before beginning the main study, we asked the participants for any queries they had, and informed that they cannot ask during the study. Then, they were asked to perform a total of 32 login sessions; each login process took 30 seconds on average. In total, each participant spent nearly 20 minutes = 32 * (30s login time) + 31 * (5s wait time). Appendix figure 6 shows the hierarchy of different login sessions that each participants performed.

Phase III: Post-Study Questionnaire: After completing the study, we asked participants to fill out a survey form about their general demographics, 2FA related questionnaire (e.g., type and frequency of 2FA usage), study-specific questionnaire (e.g., if participants experienced suspicious activity during the study).

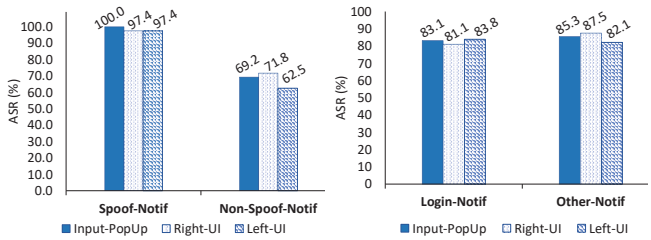
The study was approved by our University's IRB. Participation in the study was voluntary. Standard ethical procedures were fully followed. For instance, participants were well informed about the study, and they were given a choice to discontinue at any point of time during the study.

5 ANALYSIS AND RESULTS

5.1 The Benign Setting

We use responses from the participants in benign sessions to compute *BSR*, a metric to measure the login success rates of the participant with Push-2FA. The first block of the bar plot in Figure 3 shows the performance of our participants with Push-2FA in different design settings of push notification. We achieved overall *BSR* of over 98.4%. In the Push-2FA implementation with the *Input-PopUp* setting, we achieved a *BSR* of 99.3%. In case of *Right-UI* and *Left-UI* settings, we achieved *BSR* of 97.4% for each. These results show that our participants were pretty good at using our implementation of the Push-2FA scheme and they were accepting the benign login sessions with a very high probability irrespective of the push notification design (left-approve or right-approve).

Preference of Responding to Notifications: To find the participants' preferences towards the way they respond to the push notification, we use the participants' responses to the push notification triggered in the *Input-PopUp* setting. The reason behind using responses to *Input-PopUp* notifications was that the push notification in the *Input-PopUp* setting allows the users to respond



(a) *Non-Spoof-Notif* and *Spoof-Notif* **(b) *Login-Notif* and *Other-Notif***

Figure 4: HIENA with different notification settings.

to the notification in two different ways: (1) from the notification itself, or (2) from the UI. We recorded 165 responses with such an *Input-PopUp* setting for the benign sessions. We found that the participants responded majority of push notifications (60.6%) from the notification itself while they responded remaining (39.4%) of the notifications from the UI after viewing the login information for the benign sessions. Although the majority of users responded from notification, the WSRT test does not reveal a statistical significant difference between the responses from notification and UI. This analysis shows that our study participants were assessing the login information on a large fraction of the benign sessions and were engaged in the study diligently.

5.2 Attack Settings

We utilize participants' responses in the attack sessions to compute *ASR* to measure the performance of our HIENA system against Push-2FA. The overall *ASR* across all the attack settings was about 82.7%, as shown in the second block of the bar plot in Figure 3. Below we analyze the ASR for different attack settings:

- ***Non-Spoof-Notif* and *Spoof-Notif*:** Figure 4a shows the performance of HIENA with *Non-Spoof-Notif* and *Spoof-Notif* settings. We observed a reasonably high *ASR* of 68.2% with the *Non-Spoof-Notif* setting. With the *Spoof-Notif* setting, the *ASR* significantly increased to 98.7%, which is similar to *BSR* we obtained in benign trials. The WSRT test shows a statistically significant difference ($p = 0.006$) on *ASR*s between the *Non-Spoof-Notif* and *Spoof-Notif* settings with a medium-level effect size ($r = 0.44$). This result indicates HIENA with the *Spoof-Notif* setting is much more successful in deceiving the users into accepting attacker triggered push notifications compared to that with *Non-Spoof-Notif* setting. This is an expected result since if the login information embedded in the attacker's notification matches the one corresponding to the user's client machine, the attacker and user notifications would look the same to the user. Also, the WSRT test shows no statistically significant difference ($p = 0.285$) between *ASR* of the *Spoof-Notif* setting and *BSR* of the benign setting. Even when attacker's notification had a completely different login information, majority of the participants could not see the difference likely because they may have missed or not fully understood the information. In practice, the attackers might be able to mimic the user machine's information partially (e.g., not being able to spoof the IP address but being able to mimic the location), in which case the overall *ASR* will hover between 68-99%, i.e., between the *ASR* of the two extreme scenario tested in our study (*Spoof-Notif*, where the attacker fully spoofs, and *Non-Spoof-Notif*, where the

attacker does not at all spoof). Moreover, we found that 4/20 participants were never fooled while 9/20 participants were always fooled to accept the attacker's notification with *Non-Spoof-Notif*.

- ***Login-Notif* and *Other-Notif*:** Figure 4b shows the performance of HIENA with the *Login-Notif* and *Other-Notif* settings. We observed the *ASR* of 82.8% and 85.1% with the *Login-Notif* and *Other-Notif* settings, respectively. We did not find a statistically significant difference in *ASR*s between the *Login-Notif* and *Other-Notif* settings. This suggests that the injection of notifications triggered by third-party apps might not be helping to improve the performance of HIENA. However, the addition of third-party apps notification would have a significant impact in the real-world attack settings as the user would see only one notification (potentially from the attacker's session) and most likely on the top of UI. This would raise the probability of the user accepting the attacker's notification while reducing the user suspicion that could have been raised from the presence of multiple push notifications during the login process.

Preference of Responding to Notifications: We recorded 175 responses with the *Input-PopUp* setting for the attack sessions. Just like the benign sessions, participants responded slightly more from notifications (58.9%) than from UI (41.1%) after viewing the login information for the attack sessions. Our participants were assessing the login information on a large fraction of attack sessions.

Summary: In sum, our results show that HIENA can defeat Push-2FA system with a high success rate. HIENA achieved an *ASR* of 82.7% on average, and 98.7% with the *Spoof-Notif* setting. Even with the *Non-Spoof-Notif* setting, it resulted in a fairly high *ASR* of 68.2%. If the *Non-Spoof-Notif* attack attempt is repeated just twice or thrice, *ASR* will be 90% and 97%, respectively. The results also showed that users were responding after investigating the login information in the UI on a large number of occasions.

5.3 Response Time and Engagement Analysis

To evaluate the user-engagement in the study, we computed the time our participants took to complete the authentication process using our implementation of the Push-2FA system, and is referred as *response time*. We considered the *starting point* of the authentication process to be the time when the system verify the participant's first-factor i.e., user credentials, and the *ending point* to be the time when the response to the login notification is received by the system.

We found that the participants took more time to complete the authentication process in the attack setting compared to that in the benign setting. In particular, the average (standard deviation) response time in the benign setting was 10.68 seconds (8.17 seconds) while the response time in the attack setting was 13.98 seconds (11.63 seconds). The WSRT test showed that the differences in the response times in the benign and attack settings are statistically significant ($p = 0.006$ when using notification for login approval, and $p = 0.000$ when using UI). Further, as expected, the participants completed the authentication process in less time when they responded using notification compared to when they used UI. Specifically, we found the average response time of 9.31 seconds (6.15 seconds) when using notification for login approval while we obtained the average response time of 14.75 seconds (12.92 seconds) when using UI. The WSRT test showed that the differences in the response times when using notification and UI are statistically significant ($p = 0.001$ for the benign setting and $p = 0.000$ for the attack setting). However, we did not find any statistical significant difference ($p = 0.529$) on the response times between the *Spoof-Notif* and *Non-Spoof-Notif* attack settings. This analysis confirms that our participants were engaged in the study tasks and were executing them as stipulated, and they were not rushing through and accepting the attack trials without due diligence.

5.4 User Survey Results

Demographics: Appendix Table 2 shows the demographics information of the participants in our study. A large majority (90%) of our participants were 25 - 34 years old. 75% of the participants were male and 25% were female. Participants were from diverse educational backgrounds, including engineering, health, physics, education, and computer science. 30% of the participants had or were working towards a Bachelor's degree, and 70% were working towards their graduate degree. 20% participants declared that they have "excellent" computer skills while 25% had "fair", 50% had "good", and 5% "poor" computer skills. 15% of the participants had excellent, 35% had fair, 40% had good, and 10% had poor security skills. Thus our participant sample is representative of young, educated and technologically-aligned individuals. If such a sample of users could not perform well against the HIENA threat (as our results show), it is likely that older, less educated and less technologically-savvy people will be more vulnerable.

2FA Related Questionnaire: We asked participants whether they have used (or are using) any of the 2FA schemes. 85% of them reported that they have used and are familiar with 2FA schemes. 76.5% of them have used SMS passcode based 2FA, 29.4% have used mobile push-based 2FA scheme, i.e., Push-2FA, 23.5% have used mobile passcode based 2FA, and none has used phone callback based 2FA. These values do not sum to 100% because many of the users have used multiple 2FA schemes. We also asked participants about the services where they have used these 2FA schemes. 82.4% participants mentioned that they use 2FA in banking services for secure account access and transactions, 35.3% uses in social media (e.g., Facebook, WhatsApp), 52.9% uses in Email, and 47.1% uses in work (e.g., University and company accounts). These values also do not sum to 100% because many of the users have used the 2FA scheme in multiple services. These numbers show that our participants had prior exposure to 2FA and Push-2FA systems, and hence they represented a good sample for our attack study.

We also asked the participants to rank the security of Push-2FA on a scale of 1 to 5, 1 being weak security while 5 being strong security. Participants ranked Push-2FA with an average score of 4.1. This shows that they feel highly secure when using Push-2FA systems. This confirms that people do perceive that Push-2FA systems provide high security (as also reflected in the study of [8]), one of the motivations of our study.

Study-Specific Questionnaire: We asked participants whether they detected any suspicious activity during the study tasks (login sessions). 50% reported that they detected suspicious activity. We also asked whether they noticed multiple push notification. 95% said that they noticed multiple notifications in a single login attempt while 5% did not notice at all. Then, we asked the participants

about the potential reason behind receiving multiple push notification when they had tried to login only once. Majority (42.1%) of them said they had no idea, 36.8% reported that the Push-2FA system had generated multiple push notifications while only 21.1% thought multiple notifications were generated due to malicious activity. Next, we asked participants explicitly about their preferences towards the way they respond to push notifications. 50% of the participants reported that they prefer to respond to the push notification from the notification itself, and 50% prefer by opening the notification (this is well-aligned with our quantitative data in the study). Finally, we asked how often they validated the login information before providing response to the push notification. 15% of the participants said they always validated the login information, 30% said they often validated, 40% said they validated sometime, while 15% said they never validated.

These answers suggest that users were engaged in our study tasks and may have sometimes noticed that there was something wrong (which we attribute mainly to the *Non-Spoof-Notif* attack where the login information was totally different from that of the client machine's). It seems most easily noticed that there were sessions with multiple notifications, however, only a small fraction considered those multiple notifications might be stemming from malicious activity (even though they may have still accepted the attacker's notification with a high chance as shown in our quantitative results). Many participants thought that the multiple notifications were somehow generated by the 2FA system which supports our hypothesis that such notifications may be attributed to a system glitch (not a security issue). Also, only a small fraction of users noted they truly verified the login information always.

6 POTENTIAL MITIGATIONS & CHALLENGES

Blocking Multiple Near-Concurrent Login Attempts: Blocking multiple login attempts to a web service could defeat our HIENA attack. However, such an approach would significantly lower the usability of Push-2FA as a whole. Users often have to make multiple login attempts because the user may forget the password, the phone data connection may be fragile, and push notification may get dropped or delayed to reach the designated phone. Further, the user may also share his accounts with others, for instance, with his spouse or colleague. They both may often access the account at a similar point of time. Moreover, many of the users own multiple computing devices [41], and they can try to login near-simultaneously from multiple devices, e.g., from the desktop PC and the laptop. Therefore, we believe that blocking multiple login attempts to defeat HIENA is not going to be a good idea as it would have a significant detrimental effect on system usability. Blocking such login attempts may frustrate users to the point they stop using the service, which would defeat the founding motivation behind Push-2FA schemes, i.e., improving usability and increasing 2FA adoption.

The user can also be asked to deny any/all push notifications if he notices multiple instances of such push notifications during a login attempt, and report the incident to the service provider for taking appropriate action. However, in HIENA with the *Other-Notif* setting, the user would likely see only one push notification and cannot detect the attack simply based on the presence of multiple push notifications. Further, denying all push notifications would lower the usability of the system as the user would not be able to login to his service during that login attempt.

Blocking Third-Party Notifications During Login: Disabling notifications from third-party apps during 2FA may allow the user to see multiple push notifications. This may make the user suspicious about the ongoing attack as he has attempted to login only once and was expecting only one push notification. In such a scenario, the user can deny and/or report the incident to corresponding service provider, thereby defeating the HIENA attack. However, blocking notifications from other apps significantly lowers the system usability, as it prevents the user from receiving important notifications (e.g., text messages from his spouse, emails from the work, etc.) during the login process. Perhaps even more seriously, it could enable a major vulnerability that could allow a malicious app to deliberately block notifications from other apps.

7 DISCUSSION AND FUTURE WORK

Study Limitations: In our study, we could not test all possible designs and interface layouts of a push notification, but we tested representative instances widely used by Push-2FA systems. Specifically, we tested *Input-PopUp* and *Plain-PopUp* design, and *Left-UI* and *Right-UI* interface layout of the Push-2FA system. Majority of widely deployed Push-2FA systems follow similar designs and interface layout. We could not test HIENA in the real-world setting where users login to their own online services, because doing such a test, especially to create attack sessions, would not be ethically-sound. In our study, users were subject to multiple login trials within a short span of time. In real life, the users do not login in this fashion. Multiple login trials in our study could have induced some, short-term, habituation effects on our participants. However, our study results show that the users were actually paying attention to push notifications during the login trials. We achieved a relatively low *ASR* of 68% in the *Non-Spoof-Notif* setting which suggests that users were looking/analyzing at push notifications and succeeded to block some of the *Spoof-Notif* login sessions. Further, the majority (85%) of our participants reported that they either always, often, or sometimes validated login information shown on the push notification. We deliberately selected young, educated and tech-savvy participants since if they are susceptible to the attack, average users will likely be more even susceptible. However, studies with average samples of users should be conducted to further test the attack with broader populations.

HIENA vs. Other Push-2FA Variants: HIENA can be applied to other variants of Push-2FA. For instance, the attack would be indirectly applicable to the Push-2FA scheme offered by Microsoft [12] where a random code is shown on the push notification and the same code is also shown on the browser (snapshot provided in Appendix Figure 7). The user is supposed to compare the two codes and accept the push notification only if the two codes match (snapshots appear in Appendix Figure 7). However, several studies in other security applications such as device pairing [23, 37] show that the users are ill-equipped to perform such simple looking "CompareCode–Confirm" tasks. They often accept without diligently comparing the codes as they are rushed and habituated to do Thus, this Push-2FA variant would also be prone to HIENA. Future work may test the effectiveness of HIENA against such variants.

The one-push notification approach is also gaining momentum in the context of payment transaction verification. For instance, Futurae [15], Secure TAC [4] and Secure2u [5] require the user to approve the notification prompt generated on the phone for approving a financial transaction. Our attack and study results would also be applicable to breaking such applications. Here, the attacker would have to make a fraudulent transaction on behalf of the user while the user is making a legitimate transaction, which will generate multiple approval notifications on the user's phone. Since the payment scenario is slightly different from a typical login scenario, it would be interesting to evaluate the feasibility of HIENA in the payment setting with a formal study in the future.

Attacker Login to a Different Service: The HIENA attack is generally applicable even when the victim and the attacker login to different services that support Push-2FA (i.e., the attacker trying to hack into user's account at service *B* at the time the user logs into service *A*). The push notification does show the name of the service but users may not pay attention and would still likely accept the attacker's notification with a high chance. More importantly, such an attack cannot be prevented by simply looking at the presence of multiple login attempts in a short timespan, since the login attempts happen to different services. Such an attack seems ideal against password management services that use Push-2FA (e.g., LastPass).

8 CONCLUDING REMARKS

We introduced and studied a vulnerability in the design of widely deployed Push-2FA schemes (e.g., Duo Push or LastPass) that crucially rely on the user to approve a login attempt with the tap of a button on the smartphone. An attacker exploiting this vulnerability attempts to log in to the user account around the same time (or just after) the user attempts to log in thereby generating multiple login notifications on the user's phone. Since these Push-2FA schemes do not explicitly show to the user the binding between the notification and the browser's login session, the multiple login notifications all look the same to the user and there is a high likelihood that the user would just accept the attacker-triggered notification. Our lab-based study designed to test the feasibility of the attack indeed showed that the users are fallible to this attack with a very high probability (about 99%) especially when the attacker logs in from a machine spoofing the user machine's information. Even when logging in from a totally different, unspoofed machine, the attack succeeds with a reasonably high probability, which can be further increased to above 90% by repeating the attack a couple of times. Moreover, it is possible for the attacker to hide the notification corresponding to the user's login session by obfuscating the screen interface with many third-party notifications issued to the user. Since users often receive multiple notifications from the same service and attribute them to minor glitches in communication channels or phone's connectivity, the attack activity may remain oblivious to the user or the web service. We argued that defeating the HIENA attack without disrupting the normal operation of the scheme, which usually includes allowing multiple near-concurrent login attempts for the user, may be a challenging task. Explicitly binding the notification with the browser's login session by asking the user to compare the same random code shown on both the notification and the browser is also very likely to fail since most users would just click-through/approve without comparing (such effects have been seen in previous device pairing research).

ACKNOWLEDGMENTS

This work is partially supported by National Science Foundation (NSF) under the grants: CNS-1547350, CNS-1526524 and CNS-1714807, and Jazan University. We thank the reviewers for their helpful feedback.

REFERENCES

[1] Elizabeth Arens. 2019. Best times to post on social media for 2019. https://bit.ly/2J7cfLr. (2019). Last accessed Feb 28, 2020.
[2] Authy. 2019. Authy 2FA – OneTouch. https://www.twilio.com/authy/features/push. (2019). Accessed: April 22, 2019.
[3] Ben Ku. 2018. Does HTTPS Protect Your Privacy? shorturl.at/efA26. (2018). Accessed: Sep 2, 2019.
[4] CIMB Bank Berhad. 2019. Secure TAC | CIMB Clicks Malayasia. https://bit.ly/2LCXroC. (2019). Last accessed May 13, 2019.
[5] Malayan Banking Berhad. 2019. Secure 2u | Digital Products and Services. https://bit.ly/2LCX51g. (2019). Last accessed May 13, 2019.
[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 *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 553–567.
[7] Colin Lecher. 2019. Facebook app developers leaked millions of user records on cloud servers, researchers say. https://bit.ly/2RlQBE7. (2019). Accessed: Sep 2, 2019.
[8] 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*. ACM, 456.
[9] Dave Albaugh. 2019. Common tech support scams: How to identify and avoid them. shorturl.at/pyQY2. (2019). Accessed: Sep 13, 2019.
[10] Davey Winder. 2019. Unsecured Facebook Databases Leak Data Of 419 Million Users. https://bit.ly/2lR2nd5. (2019). Accessed: Sep 5, 2019.
[11] DUO. 2019. Duo Push: Duo Authentication. shorturl.at/pwE16. (2019). Accessed: April 21, 2019.
[12] EwanD. 2017. Enabling 2FA for MSA. (2017). https://bit.ly/2HihL9p Accessed; Last accessed 11 May, 2019.
[13] Facebook. 2019. Two-factor authentication for Facebook now easier to set up. https://bit.ly/2MpF3vP. (2019). Accessed: May 10, 2019.
[14] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. 2012. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the eighth symposium on usable privacy and security*. ACM, 3.
[15] Futurae. 2019. One-Touch. (2019). https://futurae.com/platform/one-touch/ Accessed; Last accessed 11 May, 2019.
[16] Google Inc. 2017. Google 2-Step Verification. https://www.google.com/landing/2step/. (2017). Accessed: May 13, 2017.
[17] Amber Gott. 2019. LastPass Authenticator Makes Two-Factor Easy. url-https://bit.ly/2HsKbh5. (2019). Accessed; Last accessed 11 May, 2019.
[18] Matt Gutman. 2015. Snapchat hacked: 4.6 million user names, partial phone numbers leaked - ABC15 Arizona. https://bit.ly/2vSSdKZ. (2015). Accessed: May 5, 2019.
[19] Identity Automation. 2018. Two-Factor Authentication (2FA) Explained: Push Notifications. https://bit.ly/3hsxDpO. (2018). Accessed: Mar 01, 2020.
[20] Identity Automation. 2019. Push Authentication using RapidIdentity PingMe. https://bit.ly/2KWvfNj. (2019). Accessed: April 22, 2019.
[21] International Association of Better Business Bureaus. 2019. BBB Tip: Tech Support Scams. shorturl.at/dsuBJ. (2019). Accessed: Sep 13, 2019.
[22] Jason Dean Jessica E. Vascellaro and Siobhan Gorman. 2010. Google Warns of China Exit Over Hacking. https://www.wsj.com/articles/SB126333757451026659. (2010). Last accessed Mar 04, 2020.
[23] Ronald Kainda, Ivan Flechais, and AW Roscoe. 2009. Usability and security of out-of-band channels in secure device pairing protocols. In *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM, 11.
[24] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. 2015. Sound-proof: usable two-factor authentication based on ambient sound. In *USENIX Security Symposium*.
[25] Mohit Kumar. 2012. Anonymous leaks database from Israeli Musical Act Magazine site #OpIsrael. https://bit.ly/2Y84yqo. (2012). Accessed: May 5, 2019.
[26] Mohit Kumar. 2012. Bulgarian torrent tracker forum hacked and accused of collecting user IP. https://bit.ly/2VSqLvf. (2012). Accessed: May 5, 2019.

[27] LastPass. 2019. The only authenticator app you need. https://lastpass.com/auth/. (2019). Accessed: Mar 3, 2020.
[28] Laura Hautala. 2019. Instagram website leaked phone numbers and emails for months, researcher says. https://cnet.co/2WjilNz. (2019). Accessed: Sep 2, 2019.
[29] Pierpaolo Loreti, Lorenzo Bracciale, and Alberto Caponi. 2018. Push Attack: Binding Virtual and Real Identities Using Mobile Push Notifications. *Future Internet* 10, 2 (2018), 13.
[30] Microsoft. 2019. Enable passwordless sign-in with the Microsoft Authenticator app. https://docs.microsoft.com/en-us/azure/active-directory/authentication/howto-authentication-passwordless-phone. (2019). Accessed: Mar 01, 2020.
[31] Ariana Mirian, Joe DeBlasio, Stefan Savage, Geoffrey M Voelker, and Kurt Thomas. 2019. Hack for hire: Exploring the emerging market for account hijacking. In *The World Wide Web Conference*. 1279–1289.
[32] RCDevs. 2019. Multi-Factor with OTP and FIDO-U2F. https://www.rcdevs.com/products/openotp/. (2019). Accessed: May 10, 2019.
[33] Ken Reese, Trevor Smith, Jonathan Dutson, Jonathan Armknecht, Jacob Cameron, and Kent Seamons. 2019. A Usability Study of Five Two-Factor Authentication Methods. In *Fifteenth Symposium on Usable Privacy and Security ({SOUPS} 2019)*.
[34] RSA Security. 2019. Mobile Authentication | Push Notification. https://bit.ly/2UA6wxC. (2019). Accessed: April 22, 2019.
[35] Muhammad Shahzad, Alex X Liu, and Arjmand Samuel. 2013. Secure unlocking of mobile touch screen devices by simple gestures: you can see it but you can not do it. In *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 39–50.
[36] Maliheh Shirvanian, Stanislaw Jareckiy, Hugo Krawczykz, and Nitesh Saxena. 2017. SPHINX: A password store that perfectly hides passwords from itself. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1094–1104.
[37] Maliheh Shirvanian and Nitesh Saxena. 2015. On the security and usability of crypto phones. In *Proceedings of the 31st Annual Computer Security Applications Conference*. ACM, 21–30.
[38] Anatoly Shusterman, Lachlan Kang, Yarden Haskal, Yosef Meltser, Prateek Mittal, Yossi Oren, and Yuval Yarom. 2019. Robust website fingerprinting through the cache occupancy channel. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 639–656.
[39] Nikhil Sonnad. 2015. What's in the Ashley Madison database that hackers released online - Quartz. https://bit.ly/1WFcrP6. (2015). Accessed: May 5, 2019.
[40] Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor. 2009. Crying Wolf: An Empirical Study of SSL Warning Effectiveness.. In *USENIX security symposium*. 399–416.
[41] Viktoriya Trifonova. 2018. How Device Usage Changed in 2018 and What it Means for 2019. shorturl.at/gmKV8. (2018). Accessed: May 5, 2019.
[42] Twilio. 2019. Authy: 2FA and Passwordless Login. https://www.twilio.com/docs/authy. (2019). Accessed: Mar 01, 2020.
[43] Jordan Valinsky. 2019. Cyber Monday was the biggest shopping day in Amazon's history. shorturl.at/grLNZ. (2019). Last accessed Feb 28, 2020.
[44] Anthony Vance, Brock Kirwan, Daniel Bjornn, Jeffrey Jenkins, and Bonnie Brinton Anderson. 2017. What do we really know about how habituation to warnings occurs over time?: A longitudinal fMRI study of habituation and polymorphic warnings. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2215–2227.

APPENDIX

A. Login Time Analysis

Figure 5 shows an analysis of the login data from a large public university.

B. Study Sessions

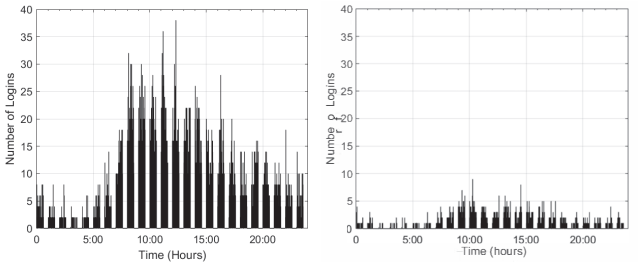
Figure 6 shows the hierarchical flow of the various types of login sessions in our study.

C. Demographics

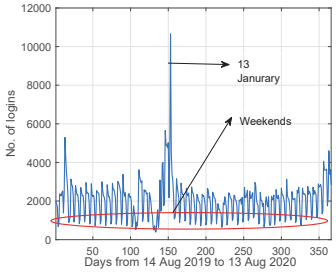
Table 2 shows the demographics of the participants of our study.

D. Various Snapshots

Figures 7 , 8, and 9 show various snapshots of push-based 2FA and variants



(a) January 13 Login **(b) Normal Day Login**



(c) The Logins for One Year

Figure 5: Analysis of the login data

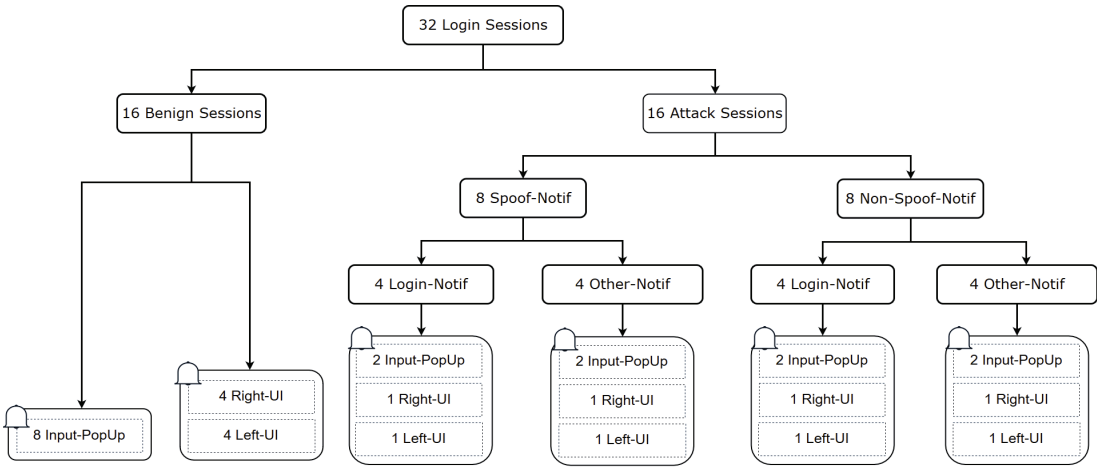
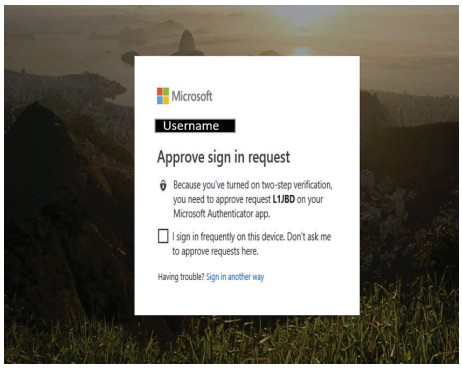


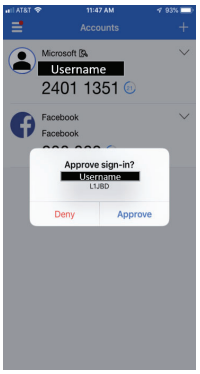
Figure 6: Hierarchy of different login (attack and benign) sessions used in our study.

Table 2: Demographics of our participants (N=20).

Gender		Age		Education		General Computer Skills	
Male	75%	18-24	10%	Bachelor's degree	30%	Poor	5%
Female	25%	25-34	90%	Master's degree	60%	Good	50%
				Doctorate degree	10%	Fair	25%
						Excellent	20%

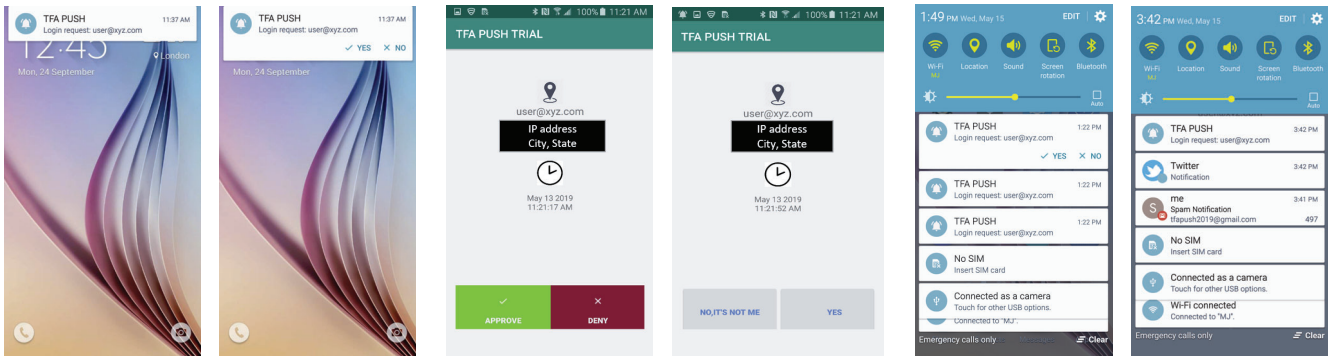


(a) Microsoft Authenticator browser UI



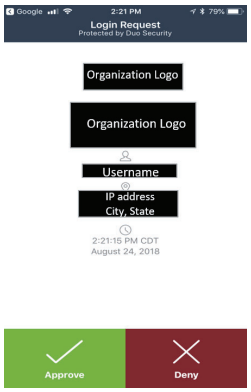
(b) Microsoft Authenticator phone UI

Figure 7: Push-2FA implementation offered by Microsoft (ComapareCode-and-Tap) (some parts redacted for paper anonymity).

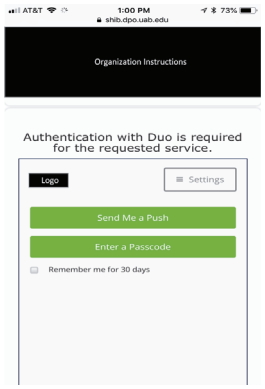


(a) Plain-PopUp **(b) Input-PopUp** **(c) Left-UI** **(d) Right-UI** **(e) Login-Notif** **(f) Other-Notif**

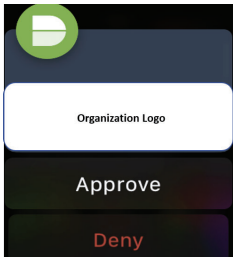
Figure 8: The interface and attack designs implemented and tested in our study (some parts redacted for paper anonymity).



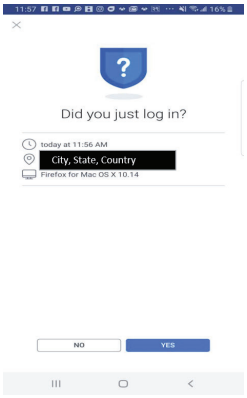
(a) Duo Push phone UI



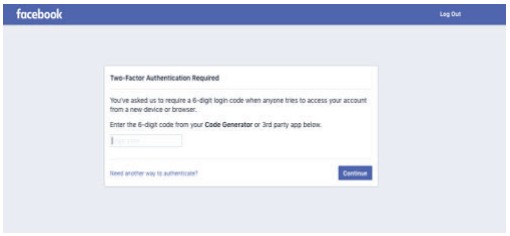
(b) Duo Push browser UI



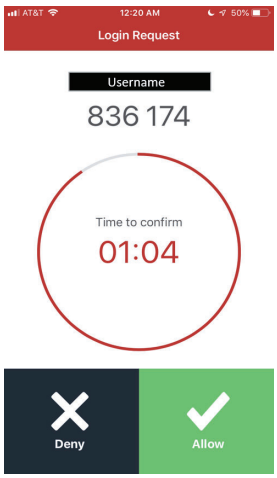
(c) Duo Push watch UI



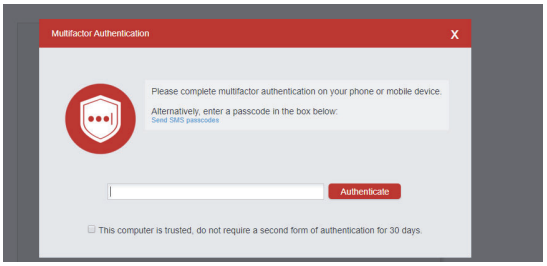
(d) Facebook phone UI



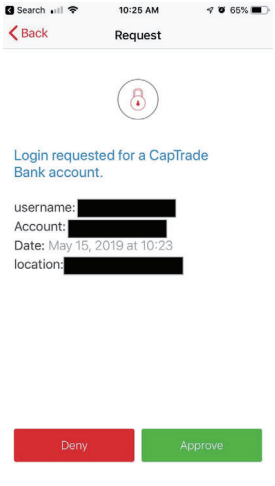
(e) Facebook browser UI



(f) LastPass phone UI



(g) LastPass browser UI



(h) Authy OneTouch phone UI

Figure 9: Push-2FA applications that we tested and are vulnerable (some parts redacted for paper anonymity).