Vulnerabilities of Multi-factor Authentication in Modern Computer Networks

A Major Qualifying Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

in

Computer Science

by

Matt Tolbert

Elie Hess

Mattheus Nascimento

May 6, 2021

APPROVED:

Professor Craig A. Shue, Project Advisor

Abstract

Multi-factor authentication (MFA) is a common method of securing private accounts in conjunction with a username and password, and helps protect against a variety of attacks. However, despite offering more security, it relies on the user to take certain actions to verify their identity. This study aims to test how secure MFA is against attacks that attempt to deceive the user into taking an incorrect action that compromises their account. We researched the types of attacks that could target MFA. From those we created two attacks that both focused on deceiving the user and had symptoms that could be noticed by the user. Then we ran a user study where we performed attacks on volunteers in which their actions controlled whether or not the account was compromised. We compared study results where users were and were not deceived into providing their credentials, as well as how users acted while logging in. We found that most users, contrary to our expectations, failed to carefully read the push notifications and treated MFA as an obstacle to logging in rather than a security measure to be carefully examined. Even when more context was added to hint at an attack, users still tended to rush through and accept an attacker's login attempt. We concluded that an important next step to improving MFA is researching how to counteract the fatigue users experience from frequent use of MFA.

Contents

List of Figures

List of Tables

1 Introduction

When attempting to gain unauthorized access to secure systems, attackers often attempt to compromise valid login credentials and bypass security measures. Recently, account login verification has been bolstered by the use of multi-factor authentication (MFA). MFA improves the authenticity of login requests by requiring users to present at least two forms of identity verification: what a user knows, has, and is. Sometimes the factor of a user's location or the time of day is also verified. While there are several benefits of MFA over traditional single-factor authentication systems, it has limitations as well. The effectiveness of many multi-factor authentication systems is reliant on their end users, who ultimately decide whether to approve or deny a login attempt. However, many current MFA implementations do not provide their end users with the context required to make informed decisions.

Businesses have become reliant upon computer networking, and security breaches can completely cripple a company's reputation and finances. The recent COVID-19 pandemic saw a worldwide shift towards remote work. In response, businesses were forced to adapt their security implementations to ensure secure transit between internal resources and employees working from home. A study conducted by password manager LastPass showed that 62% of businesses felt that MFA was the most effective way to secure a remote workforce [1].

Companies have put trust into these systems whilst also needing to deploy them rapidly to assure business continuity. While MFA is effective at protecting against certain kinds of attacks, the growing trust placed in it during an unprecedented time of global demand gives greater urgency to key questions: How much security does MFA actually add? How feasible it is to attack MFA? If attacks are feasible, how likely is it for an end user to notice them before it is too late?

One of the most pressing issues surrounding the success of a given MFA implementation is whether or not it provides end users with enough context. If a user is asked to verify that an authentication request is legitimate, the login system must provide them with enough information to make a fully-informed choice. Attackers can exploit the lack of context provided by many MFA implementations and effectively negate the additional security offered by MFA. Without the information necessary to make a decision as to the legitimacy of a login request, end users can unwittingly succumb to an attack.

In this paper, we demonstrate that many MFA implementations have exploitable weaknesses. Through the implementation of two different threat models - the packet pausing attack and malicious endpoint attack - we successfully defeat a wide variety of existing MFA implementations. The packet pausing attack, an on-path attack designed to temporarily halt the victim's network traffic, defeats MFA implementations involving a simple approval notification appearing on a user's phone and is very difficult to detect both during and after the attack. The malicious endpoint attack, on the other hand, compromises a broader range of MFA implementations, but also presents more symptoms that an end user can recognize. These attacks raise questions about whether the end user can or cannot be reasonably expected to identify or stop an attack without significant changes to MFA. Our paper makes the following contributions:

- We present the packet-pausing and malicious endpoint attacks, which compromise different MFA implementations while also presenting different symptoms to the end

user.

- We demonstrate the success of each of these attacks in both a lab setting and against end users.
- We present two potential solutions to decrease the likelihood of these attacks succeeding and demonstrate their effectiveness on end users.

The remainder of the report is organized in the following fashion. In Section 2, we discuss the necessary background information including details about MFA, its adoption, the variety of implementations, and their weaknesses. In Section 3, we discuss our design considerations and final implementations of the aforementioned packet-pausing and malicious endpoint attacks. Section 4 presents the results of our attacks in a lab setting. In Section 5, we describe the process of user study and analyze the results. We conclude in Section 6 and discuss the implications of this research.

2 Background

Attackers continue their efforts to compromise user accounts [2]. Cybersecurity experts have developed many systems to try to prevent attacks, from encryption to password standards to improvements of false website recognition. Despite all of these innovations, user carelessness can render them useless. Users frequently choose easy-to-remember passwords containing details about themselves or their interests, which attackers can guess after looking up the user's personal information online [3]. Some also use the same password on multiple sites, meaning that a breach to one site's password storage can cause a chain reaction of account compromises [4]. A crucial improvement to password security was the development of overlapping controls: applying distinct layers of security in order to cover the inherent weaknesses of each. This improvement came in the form of multi-factor authentication, otherwise known as MFA. Increasing the number of ways the user must prove their identity increases the difficulty of compromising all of them. Imagine that one is trying to secure a physical room instead of an online account. One could first lock the room with a key lock; an attacker without the key would need sophisticated tools to break the lock. Adding a second key lock would not significantly increase the difficulty of breaking into the room, since an attacker would not need any additional tools. If one instead added a combination lock, an attacker would need tools for picking both key locks and combination locks to gain entry. In the event that just one of a user's authentication factors is compromised, the user's data remains secure, and the provider of the now-partially-compromised account can alert the user to change that factor, and restore their account's full security.

2.1 Multi-Factor Authentication (MFA)

Multi-factor authentication (MFA) is an increasingly common method of confirming a user's identity via at least two methods – rather than the traditional method of requiring only a password – with the aim of protecting users from third-party attackers. On their own, passwords have many potential vulnerabilities: simple passwords can be guessed, passwords

used for multiple sites can be leaked from a data breach, and passwords written down can be found by nosy coworkers, among others [5]. To protect against these vulnerabilities, the industry began shifting to MFA. The earliest form of MFA was simple two-factor authentication (2FA), which was first made commercially available in 1986. At that time, the RSA company made a key fob with a small LCD screen that displayed a short numerical code to be appended to passwords. It was initially used by large enterprises and governments as extra protection for their internal systems. 2FA became more readily available when Google decided to release it as a security measure for some users. Google decided to implement it following suspicions of attacks by the Chinese government against human rights activists in China [6]. This release of 2FA was initially only for their business accounts, but in 2011, Google released it for all of their users [2]. Over time, 2FA evolved with improved security and active development of various methods of authentication. Currently, proper implementation of MFA requires each authentication factor to fall into a different category by type:

- Knowledge - something only you know, such as a password or PIN
- Possession - something only you have, such as a phone, bank card, or USB key
- Inherence - something only you are, such as a fingerprint or facial recognition scan

These categories are designed so that breaking the security of one will be of minimal use in breaking the others [7].

2.1.1 Importance of and Reliance upon MFA

Attackers have devised numerous ways to steal usernames and passwords. Therefore, it has become imperative for account providers to effectively implement MFA as a safety net to prevent account compromise. According to Microsoft's Director of Identity Security, Alex Weinert, companies that implement multi-factor authentication are 99.9% less likely to be compromised than companies that do not use MFA [8]. Given this statistic, it may be surprising how few users choose to enable MFA, and rely solely on a password to secure their account. A study conducted by Microsoft in 2020 revealed that only 11% of enterprise accounts had an MFA solution enabled. Microsoft hopes to increase this percentage to help mitigate account compromises and thus improve their security reputation, since accounts protected by MFA are less vulnerable to automated attacks [9]. However, therein lies a weakness in the security of MFA: stopping automated attacks that compromise accounts without user interaction does not necessarily indicate proficiency at stopping manual attacks.

Even with MFA enabled, a user's account is not perfectly defended. While users protected by MFA are less vulnerable to automated attacks, attackers can still bypass secondary forms of authentication by using targeted attacks in which they deceive victims into willingly granting attackers access to their account. Additionally, many websites allow users to mark devices as "trustworthy," requiring less information on future login attempts from the same device in order to streamline the user's experience. However, streamlining the login process in this way can leave a site vulnerable. In 2019, attackers found a weakness in a US-based banking website that allowed them to trick the site into thinking that the user had previously

marked the attacker's computer as "trusted", letting the attacker bypass MFA entirely and access users' accounts [10]. These types of attacks provide a warning: unless multi-factor authentication is properly implemented, it cannot provide fully effective security.

2.2 Relevant MFA Vendors and Products

End users have their choice of a wide assortment of MFA implementations, many related to the possession of a specific phone. These options include time-based one-time passwords (TOTPs), codes sent to the user via SMS, and application-based push notifications. Additionally, possession-based verifications unrelated to phones include physical items such as USB keys and keycards which are inserted into a certain port on a device for authentication. Forms of identification related to physical traits generally rely on specialized equipment such as a fingerprint scanner, facial recognition camera, or sophisticated voice recognition microphone. Examples of existing MFA products include third-party authentication applications open to use by the public, including Microsoft Authenticator [4], Google Authenticator [11], and Authy [12]. Furthermore, companies such as Yubico provide physical authentication keys used by a wide range of services [13]. Each of these implementations has a set of features intended to increase security with minimal effort from the user.

2.2.1 Application-Based Authenticators

In addition to possession-based forms of MFA that require the user to purchase an external device, multiple authentication applications available for download allow users to prove their identity via possession of their mobile phone. Users can link each application to multiple service providers, and are given a choice of which process to use when verifying a login attempt. These options include a one-time password login (copying a time-based one-time password from the user's mobile device to the service's website), a code-matching option (selecting a certain number in the application that matches one shown on their computer), and a simple approval login (selecting the "Approve" button from a push notification that appears on the user's device).

The one-time password option works by synchronizing a pre-shared key between the application on the user's device and the authentication server, which ensures that the password displayed on the application will always match the password the server is expecting. This option requires more effort from the user, since they have to manually transfer the password from their mobile device to another device. On the other end of the spectrum, the simple approval push notification option just notifies the user that a login attempt is being made, and directly gives the user the option to approve or deny it via prompts on the push notification. The code-matching option combines traits of both, requiring users to match information between screens, but only requiring a button press.

Some authentication providers have developed additional security for the push notification option. For example, Microsoft includes a short string of characters that appears on both the page making the login attempt, and within the associated push notification. This tactic allows the user to verify that the push notification they are approving actually corresponds to their login attempt. We observed this behavior only when logging in directly

into Microsoft's website; it does not seem to be implemented into Microsoft's SSO system to allow other websites to benefit from it.

2.2.2 Yubikey

USB keys are a popular form of possession-based digital security, and a common series is the Yubikey line from Yubico [14]. About the size of a flash drive and designed to fit on a keyring, each Yubikey contains unique data that can be read by a variety of authentication services to corroborate a user's identity. One of the main benefits of Yubikeys is their compatibility with over 250 services, including popular Single Sign-On providers such as Google and Facebook, and a variety of password managers [15]. However, according to research presented in 2018 during an IEEE Symposium based on the Yubikey 4, Yubikey NEO, and Yubikey Nano, a significant number of users experienced difficulties configuring their Yubikey: nearly one-fifth of technologically competent university students studied were unable to configure a Yubikey 4 to work with a Google account, nearly 70% were unable to enable Yubikey-based 2FA on a Facebook account, and almost 75% were unable to configure it for use with Windows 10 [16]. Users in the study cited a lack of comprehensible documentation as their primary obstacle. At the conclusion of the study, only 13 out of 25 users preferred using Yubikeys to not using MFA at all (an additional four users preferred other MFA options). Users given a Yubikey Nano specifically noted that it was very small and easy to lose [16].

We tested the Yubikey 5 and Yubikey FIPS, the next generation of Yubikeys. Two testers independently navigated to a website in order to find instructions for using their Yubikey with specific services. The researcher testing the Yubikey 5 was able to easily locate a list of websites which permitted authentication via Yubikey 5 and subsequent instructions on how set up should be done for that specific service. The user testing the Yubikey FIPS arrived at the corresponding page, but it was empty instead of listing the usable services. After a bit of searching, the researcher found a list of websites accepting the Yubikey and registered their Yubikey successfully. For both researchers, registering the Yubikey when following the instructions and using the Yubikey to log into services were fairly smooth processes, showing an improvement from previous reports on ease of use.

2.2.3 FIDO and FIDO2

FIDO is an authentication standard in which a private key is stored on a device owned by the user and a matching public key is given to the service when registering two-factor authentication [17]. Each service has a different key paired with the user's device, and information for accessing the private key is stored only on the device itself. A user can then authenticate themselves using a variety of methods, including voice, fingerprint, facial recognition, or a security key [17]. FIDO then authenticates the user with the service they desire to use. The combination of the local authentication and FIDO's own authentication with the service works to cover the weaknesses of each authentication protocol if executed alone. The inclusion of biometrics in the system allows for authentication via all three categories assuming the user has a password alongside this system [17]. However, it allows the user to use simpler methods if they do not want the extra trouble of verifying in two or three different ways.

Figure 1: How People Use FIDO [17]

Part of FIDO's security is that the device's private key is both created and stored locally; only the public key is given to the service. This property prevents a single account compromise from creating a chain reaction. Since each key pair is unique, an attacker cannot use a key pair for one account to attack a different account [17]. However, if the user only uses FIDO to secure one "main" account and then uses that account as a single sign-on identity provider to log in to other accounts, this FIDO feature is eliminated.

FIDO2 (also known as webauthn) simplifies the process of FIDO authentication. It works by storing private keys locally on users' browsers, removing the need for a separate device [18]. In this case, a user's computer would require some way to perform the authentications outlined above, such as having either a built-in or peripheral fingerprint scanner. All authentication information must be stored locally to make account breaching more complex. This change results in FIDO2 being fairly unique among forms of possession verification: it relies on the user's browser (and, therefore, their computer) instead of an external device like a phone [18]. However, it can still use an external device if the user chooses. It also allows for further evolutions - such as fingerprint scans - in order to achieve three-factor authentication by adding a form of inherence verification. However, FIDO2 does not function with every web browser, which may result in users choosing a familiar browser over a more secure one [18]. Furthermore, FIDO2's choice of verifying a user's browser means that users will not be able to use FIDO2 to secure accounts that they use with different browsers or different computers.

2.2.4 Email and SMS

Email- and SMS-based authentication are fairly similar on the surface: after a user begins a login attempt to a given service, that service will send a short code (frequently 6 digits) to a predetermined email or phone number. Next, the user accesses their email or phone, and transfers the code into the service to log in. Finally, the service verifies that the code entered matches the one it is expecting, and grants the user access to the account if so. However, despite these similarities, email-based MFA does not meet the criteria of multi-factor authentication on its own. The key difference lies in the device(s) used to access the code: while SMS codes are only sent to a specific phone, email accounts are not device-specific. If an "MFA" code is sent to a user's email, an attacker can log on to that email account with just a username and password and obtain the code. Since no additional possession or inherence factor is required, MFA does not apply to email-based code verification unless the user's email account is independently protected by MFA. Furthermore, some services, such as Google Voice or iMessage, allow users to receive SMS messages on their computers.

Similar to email, these services do not qualify as MFA as they allow the user to receive codes on the same device making the login request.

2.2.5 Existing Security Features

An attacker placed on a network between a victim and the server they are trying to access grants that attacker a great amount of power to meddle with the victim's traffic. Because security experts are well aware of the power of an on-path adversary, any attempts by the attacker to meddle must be stealthy enough so that neither the victim nor the service notices anything wrong with the traffic. Furthermore, communication may be encrypted, preventing an attacker from gaining easy access to the actual information being shared, even if they can access the raw data being transmitted. Vulnerabilities that allow attackers to meddle while remaining undetected are considered very powerful, and as such, cybersecurity experts have developed security features to prevent such vulnerabilities. One example is known as HSTS.

HTTP Strict Transport Security (HSTS) is a web security policy that enforces HTTPS between a web server and its clients [19]. It is important for identity providers to ensure that all of their relying parties use HSTS: since attackers generally choose the path of least resistance, a relying party that does not use HSTS become vulnerable to theft of the identity provider account. This vulnerability is especially true if a successful login to the relying party causes the identity provider to automatically trust the browser; in such a case, a successful attack against a relying party would allow the attacker to immediately open the identity provider and compromise the main account. Such a compromise could be devastating: an attacker could change account information as they please in order to fully take over the account, potentially removing the original user in the process or merely disabling MFA on the account to enable them to easily and repeatedly re-access the account.

This mechanism makes on-path adversary attacks nearly infeasible on sites using HSTS. However, HSTS is only enforced on about 19% of sites [20]. This low rate of use means that most identity providers likely have a wide web of weak relying parties that can be used in order to compromise their accounts. Since attackers generally attempt to break the weakest link in a chain, the best steps to reinforce web security are not to increase the maximum possible amount of protection, but instead to ensure that existing, functional defenses are widely deployed.

3 Implementation

Before implementation of the lab and user studies, we first had to assess exactly how MFA could be broken. The search had to be narrowed to attacks where a user's actions would be the deciding factor for the account being compromised or not. Furthermore, we needed to consider attacks that could be reasonably implemented within a limited time frame yet be robust enough to fool users.

3.1 Attack Design Considerations

In order to ensure the design of robust attacks that could compromise a wide variety of multi-factor authentication styles, we devised a method for evaluating the threat models of

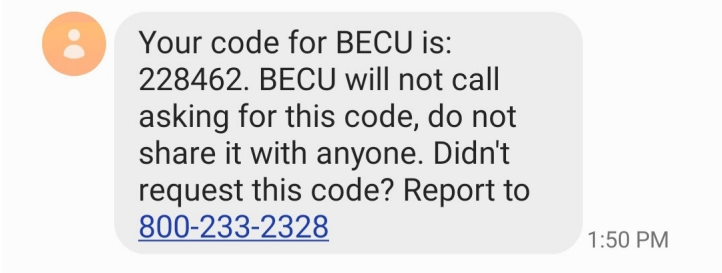

Figure 2: Sample MFA code with warning

several attacks. Sections 3.1.1, 3.1.2, and 3.1.3 discuss these considerations as well as the network built to support the implementation of these attacks. Our first step was to examine and discover the vulnerabilities that exist in popular MFA systems.

3.1.1 Notable Weaknesses of MFA

While MFA provides additional security over traditional single-factor authentication, it is an imperfect system with known vulnerabilities. One such vulnerability occurs through social engineering scams. Potential victims receive a phone call that informs them that one of their accounts may have been compromised, such as a suspicious purchase being made on an online shopping service or a login request coming from a foreign country. Victims are coerced into speaking with the attacker posing as a representative of the company holding the account, and are informed that the representative needs to log into their account in order to secure it [21]. Some users may willingly hand over their username, password, or even MFA code, believing those steps to be necessary to secure their account, unaware of the fact that the "representative" with whom they are speaking is actually an attacker.

Some MFA providers include a warning in their SMS message reminding the user not to share the code with anyone else, and explicitly mentioning that no representative from the provider will call and ask for the code. However, this practice is not universal.

Another of the most prominent threats to MFA are SIM-swapping attacks. In a SIM-swapping attack, an attacker fraudulently obtains a SIM card with the same number as their victim, enabling them to receive any SMS-based MFA tokens on their own phone instead of the victim's phone [22]. This type of attack also involves social engineering, but targeted at the SIM provider rather than the victim. The attacker obtains information about the user through social media, public records, or similar methods, then contacts the SIM provider's customer service line and pretends to be the victim [22]. They claim that they have lost their SIM card, need the old SIM cancelled, and want a new one with the same phone number delivered to them [22]. The SIM provider may then send a new SIM card to the attacker, and disable the authentic user's SIM card [22]. By the time the user notices that their phone service is no longer functioning, the attacker may have already fraudulently accessed the user's accounts. As these attacks target a phone provider rather than the user, there is little a potential victim can do to secure themselves. However, it is restricted to SMS-based MFA.

Another weakness of MFA occurs as a result of a lack of context presented to the user. Some MFA providers include very little information in their messages that a user can use to determine whether or not the SMS or push notification was triggered by their login request.

Figure 3: Google SMS MFA challenge

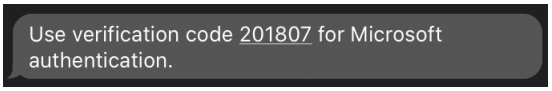

Figure 4: Microsoft SMS MFA challenge

Examples of these context-light messages can be seen in Figures 3 and 4. When a user receives a message from these providers, they are given only three pieces of context beyond the MFA code itself: the phone number from which the SMS was sent, a claim that it was sent from Google, and the time at which the message arrived.

Notably, the SMS message does not provide any context regarding the login request that triggered the code to be sent. A user can reasonably assume that the login request was made recently, but they are not provided with information about the location of the device that made the request or the service the requestor is attempting to log in to, among other potential forms of verification.

Push notification MFA has the potential to contain similar vulnerabilities, but many MFA providers offering app-based MFA include more context with their push notifications than they do with their SMS codes.

This notification provides the user with the email address associated with the account of the request, the operating system of the device making the request, the location of the device making the request, and the time the request was made. If any of these factors do not align with what the user expects, they can decline to authorize the request. However, none of these factors are foolproof since an attacker can make a fraudulent request from a device they know to have the same operating system as the user's computer, in the same general area, and at a similar time. Alternatively, an attacker can guess at these factors, performing attacks against so many potential victims that they can be satisfied even if a tiny percent of those attacks succeed. This attack model mirrors that of robocallers: sending automated phone calls can cost as little as $0.01 each, minimizing the costs of failure [23].

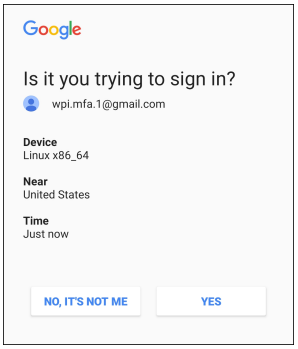

Figure 5: Example of Google MFA push notification

Threat Model	Requires on-path	Power of adversary	Difficulty to Implement	Can easily target individual users?	Symptoms
Typosquatting	No	Weak	Easy	No	User could identify that domain name is incorrect
Malicious Gateway (WiFi/Router)	Yes	Very Strong	Medium	No	Can passively view traffic without user awareness
Transparent Proxy	Yes	Moderate	Medium	Yes	Very difficult to detect
DNS Cache Poisoning	Yes	Strong	Hard	No	Hard for individual user to detect
BGP Hijacking	Yes	Very Strong	Very Hard	No	Hard for individual user to detect
Phishing	No	Weak	Very Easy	Yes	Varies

Table 1: Threat models evaluated by their feasibility of use for this study

3.1.2 Threat Models Evaluation

Taking HSTS into account, we evaluated multiple threat models based on their feasibility, control, difficulty of implementation, ability to target individual users, and symptoms, if any, visible to the end user.

When considering the threat models evaluated in Table 3.1.2, it became clear that combining these attacks would yield a more effective result than any one attack on its own. In order to best achieve the goals of the study, we needed to consider combining the threat models. We based our choices on implementation difficulty, on-path requirement, and the perceivable symptoms. In particular, we found that malicious gateways, transparent proxies, typosquatting, and phishing are all individually effective attacks whose potency can be greatly enhanced when combined. We also found that DNS cache poisoning and BGP hijacking attacks were too difficult to implement when compared to the others. Ultimately, we decided that the best combinations of these attacks were typosquatting with phishing and typosquatting with a transparent proxy.

Typosquatting refers to registering a domain that looks very similar to an existing website and, in this case, configuring the copied domain to resemble the original site. For example, an attacker might choose to register the domain *wwwexample.com* in an effort to emulate the real domain *www.example.com*. By itself, however, typosquatting is weakened by the fact that it requires users to make a typographical error in order to be directed to the

site. However, this weakness can be addressed by combining typosquatting with a phishing attack. By sending emails to a large number of potential victims, the attacker attempts to trick recipients into navigating to the typosquatted domain.

The combination of typosquatting and phishing is relatively easy to implement. An attacker would only need to register their top-level domain, create a UI that mimics the chosen site, and send emails en masse that might convince readers to click on the typosquatted link. This type of attack is likely to be successful as even experienced users are susceptible to typosquatting, and the potential consequences are dire. If the fake site poses as an e-commerce site, users may be convinced to give up not only their login credentials, but their credit card information as well [24]. If a large number of victims give up their login credentials to a typosquatted site, the public may lose confidence in the security of the original site, damaging its reputation and losing users and/or customers [25]. Furthermore, these attacks are difficult to detect, particularly if the typosquatted site is a perfect copy of the original. In this case, the only symptom a user can notice is in the incorrect domain name.

Additionally, we chose to combine the typosquatting attack with a transparent proxy. A transparent proxy is a server that is on-path in between a user and an end host that can view and in some cases, modify traffic unbeknownst to the user. By combining these two attacks, an attacker can direct users to their typosquatted domain without needing to send any phishing emails. For example, an attacker could configure their transparent proxy to alter requests from *www.example.com* to *wwwexample.com*. Unlike the combination of typosquatting and phishing, the combination of typosquatting and a transparent proxy requires the adversary to be on-path between the victim and their intended destination. However, the transparent proxy cannot be seen by the end user and does not rely on deceiving its victims with a specific phishing attempt. While a phishing attack requires its victims to make a mistake in order to be successful, the transparent proxy only requires that its victims not notice the symptoms of the attack. Similar to the typosquatting with phishing attacks, the potential for personal information theft is significant and could lead to consumer distrust of the website that the attacker emulated.

Despite the benefits offered by the transparent proxy over phishing, the typosquatting part of the attack has symptoms that the team could not mitigate. The victim's URL bar will always show a domain name with a typo, and while we can choose typos we think will be less noticeable, we cannot prevent them from showing up entirely. As such, we wanted to develop a stealthier attack that is not reliant on user error, and that even well-trained users may not notice until the attack is completed. This summary is explained it in detail in section 3.2.

The packet pausing attack, as we named it, requires the attacker to be on-path with the victim and to be able to read and modify the victim's traffic, but not necessarily to decrypt it. As such, an attacker may perform the packet pausing attack if they control the router through which the victim connects to the internet, but they do not need direct access to the victim's computer. An attacker can set up a public WiFi network and wait for victims to join it. The attack exploits the lack of context provided to users in many forms of simple approval push notification MFA. It involves selectively capturing the user's traffic, determining when they make an authentication request, and notifying an attacker to make a concurrent authentication request. To the user, it will appear as if the site is processing their authentication request when they receive a push notification. Unbeknownst to the

Attack	Detectability	Threat Posed	Description
Typosquatting + Phishing	Medium	Very High	Can compromise most kinds of MFA. Does not require HTTPS to be broken.
Packet Pausing	Very Hard	Low	Can only compromise simple approval push notification MFA, but is nearly undetectable.
Typosquatting + Transparent Proxy	Hard	Very High	Can compromise most kinds of MFA. Transparent proxy requires non-HSTS sites.

Table 2: Final attack design selections

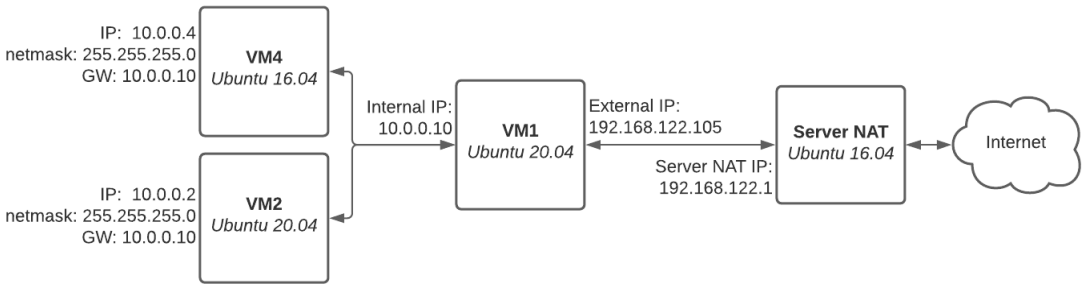

Figure 6: Isolated Network Design

user, this notification corresponds to the attacker's authentication request, and accepting it will allow the attacker access to their account. The attacker can then release the user's traffic, resulting in the user's authentication request reaching the relying party, which will then send a second push notification to the user. The only symptoms the user can notice are the long processing time and the duplicate push notification. The delay in packet arrival time is known as jitter. Table 2 summarizes these threat models.

3.1.3 Isolated Network Design

An isolated network is essential for safely and legally testing network attacks. In order to establish a network that we fully controlled, we created three virtual machines on a server. We configured the client machine (VM4) for testing using Ubuntu 16.04 Desktop, and the remaining two hosts (VM1 and VM2) using Ubuntu Server 20.04. We connected these devices to a local isolated network on the 10.0.0.0/24 subnet with a single host (VM1) acting as a gateway to the external network. The network infrastructure is shown in Figure 6.

Having all traffic pass through the router (VM1) allowed strict control over the outgoing connections from our isolated network. This configuration also laid the groundwork for the implementation of our final attack designs. Considering that malicious actors can position themselves on a network path in a variety of ways, we derived two distinct scenarios representing the network configurations of our attacks.

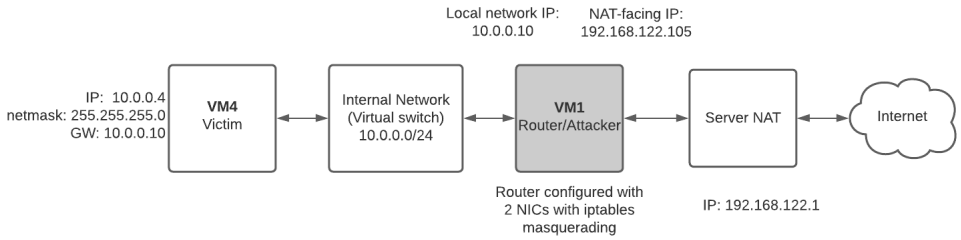

Figure 7: Packet Pausing - Attacker is on path at VM1

3.2 Packet Pausing Attack

When considering attacks that could effectively break MFA without being easily detected by the user, we considered a way to improve the time-based attack on the push notification verification. In the time-based attack, the attacker would attempt to access the account shortly before the user while at a nearby location. The attacker would need to time their attempt perfectly so their notification arrives before the user's. The user would then receive a second notification that corresponds to their authentic login request, and may dismiss the duplicate notification as a simple server error. Such an attack requires precise timing in order for the malicious notification to arrive before the one corresponding to the user's request. Thus, we formulated a way in which the malicious actor could ensure that their login attempt is processed first by strategically capturing the victim's internet traffic and delaying the victim's login attempt from reaching the server until after the attacker's attempt. We refer to this attack as the packet pausing attack.

3.2.1 Packet Pausing Attack Network Layout

In the packet pausing scenario above, we positioned ourselves to act as the router (VM1) for a local network that the victim (VM4) used to connect to the internet. The victim (VM4) was connected to the network with the IP addresses 10.0.0.4. This machine was connected to the Internet through a local subnet (10.0.0.0/24) through the router (VM1). The router acted as a network address translator (NAT) and forwarded the packets to and from the Internet.

By sitting on-path between the victim and the Internet, the attacker can manipulate the user's network traffic in any way they choose, oftentimes unbeknownst to the end user. In this scenario, the safeguard of multi-factor authentication can be nullified in several ways. For the purposes of this research, we restricted our approach to no more than basic traffic manipulation with no ability to decrypt traffic. From this vantage point, we inspected packets (though the packets remain encrypted), delayed them, dropped them, or passed them on untouched. For this attack, we focused only on pausing a login request with the intent to have an end user involuntarily accept our malicious login request through their 2FA push notification.

3.2.2 NetfilterQueue

To set up the attack, we used the `NetfilterQueue` Python library in conjunction with `iptables` to redirect all TCP traffic from the victim's IP address to a Python script we wrote. By configuring an `iptables` rule to send traffic to `NFQUEUE` with a particular queue number, we were able to use Python to define a function to apply to every packet matching the `iptables` rule. Any `iptables` rule could use `NFQUEUE` as its target, which gave us a large degree of control over how precisely packets must match pre-defined criteria in order to be placed in the queue.

3.2.3 Using NetfilterQueue and Scapy to Scan Packets

Once `NetfilterQueue` sent packets from the victim's computer to our Python script, we used the scapy Python library to dissect them and gain access to all of their fields (though we are unable to decrypt payloads). In particular, scapy made it trivial to determine the layers any given packet contained, as well as the size of the payload. Scapy also allowed us to examine a precise selection of attributes for a large number of packets, which was invaluable for collecting the preliminary data necessary to craft our attack. Finally, once we gathered all of the information we needed, we used `NetfilterQueue`'s "accept" function to send the packet to its original destination.

3.2.4 Identifying and Pausing Login Packets

We began crafting the packet pausing attack by collecting data on the information sent by the victim's computer whenever they made a login request. `NetfilterQueue` and scapy made this process easy: we configured `iptables` to send all TCP traffic from the victim's computer to a certain queue, then wrote a function to print out the length and (encrypted) payload of each packet received, then return those packets to the usual flow unmodified. With the script written, we then repeated the following data collection process several times:

First, we configured `iptables` on the router to *not* direct packets to `NetfilterQueue`, making it act like a normal router would. We then began a login sequence from the victim's computer, but paused after entering their password but before pressing the button to submit the password, as seen in Figure 8. At that point, we changed `iptables` on the router to direct traffic to our script, beginning the "attack". We then returned to the victim's computer, and pressed the "Next" button to send the password to the identity provider. Our now-active script printed the data for each forwarded packet to the console. After the password was authorized, and the user's browser proceeded to the next page, we disabled our script on the router, returned the network configuration to the same state it had before the attack, and recorded the packet information displayed on the console.

While examining the data gathered in this way, we noticed a pattern in the lengths of packets sent at certain points during the login process. For example, the second packet with a payload sent in the login sequence was always 202 bytes long, while the fifth was either 987 or 988 bytes. Following this pattern, we were able to use packets' lengths to determine what portion of the login sequence they belonged to. We hypothesized that we could use a certain sequence of packet lengths as a trigger to cause our script to begin pausing packets, commencing the user-visible part of the attack. When we tested this hypothesis when logging

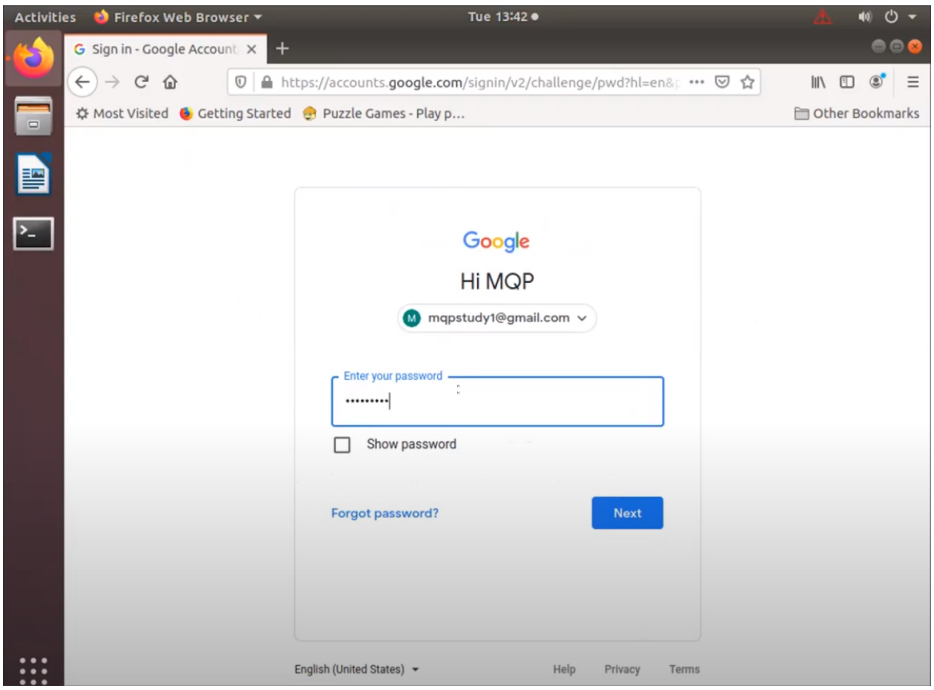

Figure 8: User's screen immediately before the packet pausing attack begins

in to other sites through Google, though, we discovered that the pattern did not always hold. However, we discovered that the attack worked if we set the trigger condition to be a packet that was 1400 bytes long.

The actual attack was fairly straightforward: our Python script began by scanning each packet sent to `NetfilterQueue`, and checked if it matched our trigger condition. Until a packet matched, the script simply accepted the packet and printed out a message containing basic information about the packet. Once a triggering packet arrived, the script toggled a global variable marking that the attack was underway, sent the triggering packet to a temporary Python queue, and printed a message to the console. All incoming packets were then sent to the same queue without any other actions performed on them until the attacker pressed a button on their keyboard. Once the attacker did so, the script accepted all queued packets in the same order they were received in, and the program returned to its initial state. As the packets were delayed but not modified the only effect that would be visible to the user was a slight delay, otherwise known as "jitter," in their login attempt being processed. We hoped victims would dismiss this delay as just their request taking a long time to process.

3.3 Malicious Endpoint Attack

When considering the extensive list of multi-factor authentication mechanisms in use, we determined that it was of utmost importance to build an attack that could compromise a wide array of these mechanisms. The following sections outline our construction of an typosquatted web server attack designed to compromise several styles of multi-factor authentication including push notifications, push notifications with number verification, one-time passcodes, text messages, email, and phone calls. For the purpose of a lab study, we elected to direct

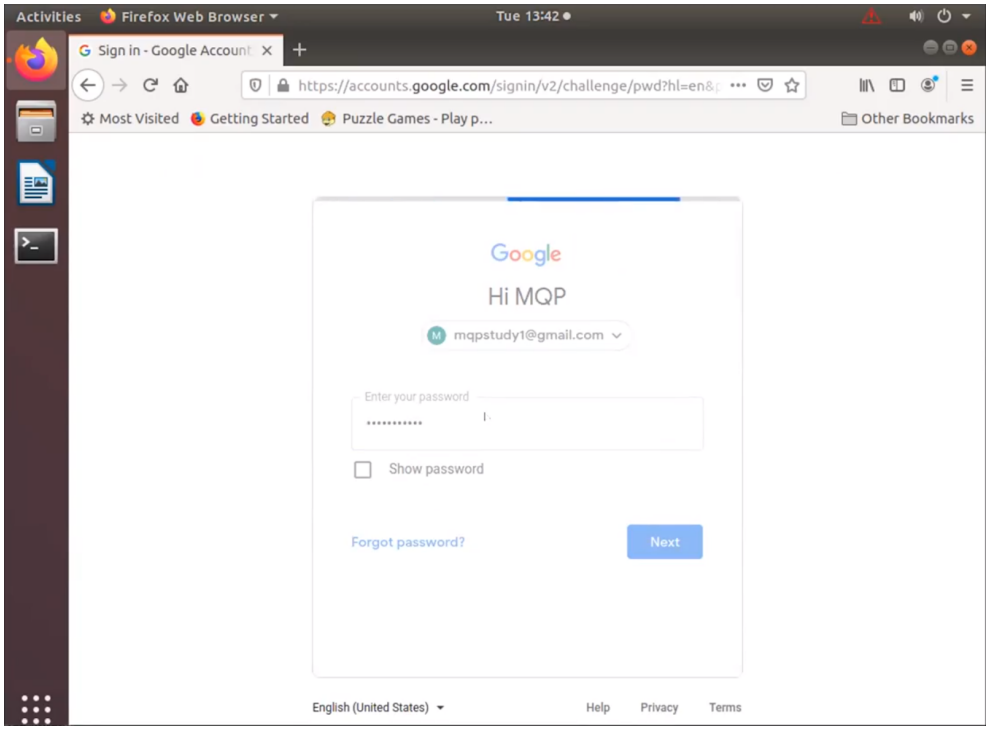

Figure 9: User's screen during traffic pause

users to our malicious domain by stripping away the secure socket layer (SSL) of HTTPS and subsequently injecting code causing a redirect upon login. It is important to note that the typosquatted web server outlined in Section 3.3.4 could also be used in a targeted attack when paired with a phishing attack.

3.3.1 Malicious Endpoint Attack Network Layout

Before we could begin to implement our attack, we needed to design a network suitable for testing in a lab setting. The malicious endpoint network diagram, shown above in Figure 10, allowed us to perform these attacks in a controlled environment with the attacker placed on-path between the victim and the Internet. Two local virtual machines, the victim (VM4) and the malicious endpoint (VM2) were connected to the network with the IP addresses

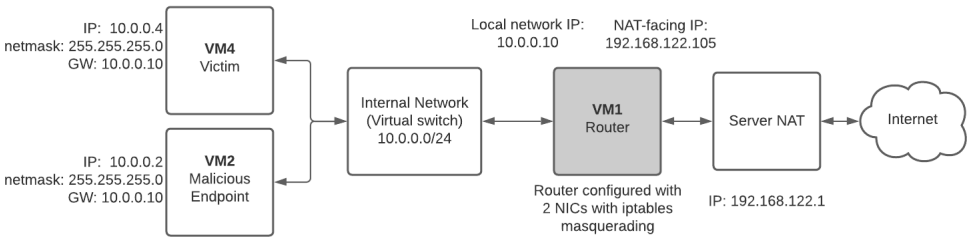

Figure 10: Malicious Endpoint - Attacker is on path controlling router and malicious web server

10.0.0.4 and 10.0.0.2 respectively. These machines were connected to the internet through a local subnet (10.0.0.0/24) through the router (VM1). The router acted as a network address translator (NAT) and forwarded the packets to and from the internet. With our on-path adversary (OPA) acting as the router, we gave the OPA more control over traffic passing through it than in the packet pausing attack. In order to begin this attack after we configured the network, we first needed to break the security between the end user and target site provided by SSL.

3.3.2 Stripping SSL

Following the release of RFC 6797 in 2012, major web service providers, such as Google, have promoted the use of HSTS as a means of preventing on-path adversary (OPA) circumvention of the secure socket layer (SSL) of HTTPS communications [26]. HSTS forces users to use HTTPS by blocking users who attempt to connect over HTTP. While HSTS provides enhanced security and makes OPA attacks much more difficult, it has not been fully implemented across the web. In fact, as of 2018, only 6.03% of the Alexa top one million sites implement HSTS [27]. While identity providers were generally not susceptible to OPA attacks due to HSTS, most other sites across the web were. This significant vulnerability at the relying party is where we chose to attack SSL in order to eventually defeat MFA.

In order to attack relying parties, we looked to existing methods to bypass SSL communications such as SSL*Split* and SSL*Strip*. SSLSplit is a tool that allows an on-path adversary to act as an intermediary between a client and server. This tool works by forming an HTTPS connection with both the server and client and as a result, decoding and modifying the traffic in real time [28]. Despite its promise as a sophisticated OPA attack, SSLSplit is difficult to implement due to the need to forge certificates. Certificate authorities have made forgery cumbersome and have rendered SSLSplit an insufficient means of attacking modern communications. For this reason, we chose to attack SSL using SSLStrip.

When dealing with sites that do not implement HSTS, SSLStrip proved to be a powerful tool. SSLStrip works by forcing HTTP communications to remain on HTTP rather than accepting a server's attempts to force a user to use HTTPS [29]. It accomplishes this by listening to all communications on port 80, the default port for HTTP communications, and actively rewriting all links on pages from HTTPS to HTTP. On VM1, our malicious router, we coupled SSLStrip with an `iptables` rule to block all traffic on port 443, the default port for HTTPS communications. This had the effect of preventing users from using secure connections to communicate with a relying party's web servers. With the SSL stripped, all communication between the end user and a relying party was forced to plaintext. Consequently, we were able to take full control over a user's traffic to that relying party. With HTTPS defeated, we were then free to begin modifying these communications.

3.3.3 Injecting custom code with a transparent HTTP proxy

In order to modify communication between the end user and the relying party, we created a transparent HTTP proxy to handle traffic at the router, VM1. The HTTP proxy listens on port 80 and redirects all traffic through itself, enabling this traffic to be read and modified using custom handlers written in Javascript. This proxy configuration was transparent be-

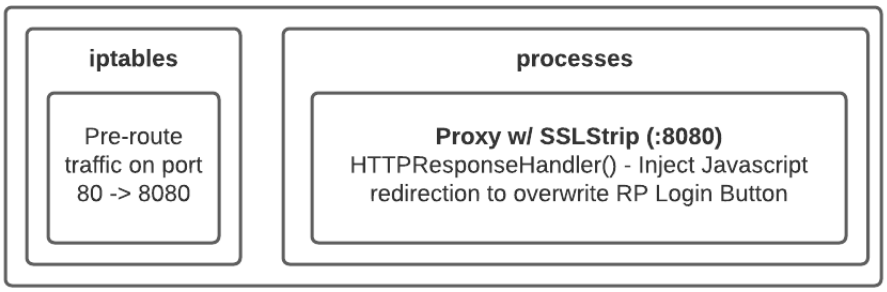

Figure 11: `iptables` redirect to proxy w/ SSLStrip

cause the end user was unable to determine that its traffic was processed through a proxy at the OPA. In order to run both this proxy and SSLStrip, which both listen on port 80, we utilized Bettercap, an open-source tool which allowed proxies to work alongside SSLStrip [30]. By combining these services into one package, we were able to greatly reduce the complexity of packet handling and minimize the need for multiple `iptables` rules.

After creating an HTTP proxy with SSLStrip, we identified where to inject a redirection on *agame.com*, a relying party that supported single sign-on (SSO). In order to create this redirection, the proxy read the HTTP body of all communications from the server to the client in order to scan for the packet containing the Javascript code of the sign-in button. When the proxy found this packet, it overwrote the code corresponding to the login button with a redirect to a domain we controlled. The proxy then rebuilt the HTTP packet and forwarded it to the user. Importantly, this injection occurred without the end user being able to identify or respond proactively to it, enabling us to proceed to the final stage of the attack: compromising the user's MFA at our malicious domain.

3.3.4 Compromising MFA with a Typosquatted Domain

After the user attempted to login, the injected code redirected them to our typosquatted domain, which we designed to mimic a legitimate identity provider login screen (Figure 12). In designing our malicious endpoint, we first created a simple web server using Node.js. Using HTML and Javascript, we were able to create login sequences that mimicked the login sequences of major identity providers, such as Google. At this stage, the user could detect the attack by noticing the typosquatted domain in their search bar or the crossed lock icon indicating an unsafe connection. However, previous research has shown that all users are at least somewhat susceptible to typosquatting attacks, regardless of education level or cybersecurity familiarity [24]. When coupled with a seamless redirect from a trusted relying party and a site design nearly identical to the real identity provider's, we suspected that the attack would be extremely difficult to detect.

Upon issuance of the legitimate login request, the server then forwarded the MFA challenge to the end user. When the user accepted this MFA challenge thinking they were using SSO to login to the relying party, the server finalized account login and full account takeover could then be completed. A majority of the common styles of MFA were susceptible to this attack. Upon successful account compromise, the malicious server redirected the user back

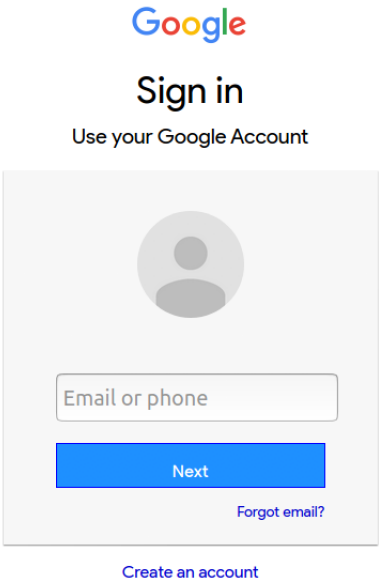

Figure 12: Malicious Server Login Sequence - Configured to mimic Google Login

to the relying party.

Figure 13: Malicious server typosquatted domain name with HTTP lock warning

4 Empirical Exploration: Adversary Capabilities in a Laboratory Environment

In our laboratory tests, we wanted to determine the precise capabilities of our attacks, considering the wide array of MFA implementations available as well as the significant differences in the threat models of the packet pausing and malicious endpoint attacks. To this end, we tested our attacks against a variety of MFA implementations including SMS codes, TOTP codes, e-mail codes, phone call codes, push notifications with simple approval, and push notifications with code matching. When testing the malicious endpoint, we created several page designs to mimic the actual layout of Google's login screen. Google implements a single-page design where the page dynamically changes as the user progresses through the authentication process (Figure 14). We emulated this behavior by creating a similar single-page design (Figure 15), serving it with a simple HTTP server, and typosquatting on the domain *accountsgoogle.com*.

In order to thoroughly evaluate the malicious endpoint, we tested it against SMS codes, TOTP codes, e-mail codes, phone call codes, push notifications with simple approval, and push notifications with code matching. We deemed the adversary to be successful in tests using this attack whenever the malicious endpoint could retrieve the MFA code or be authen-

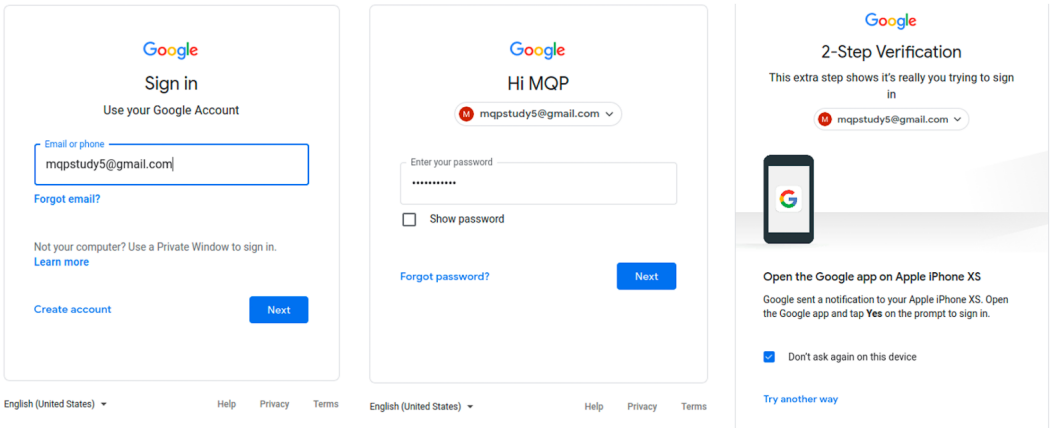

Figure 14: Google login procedure

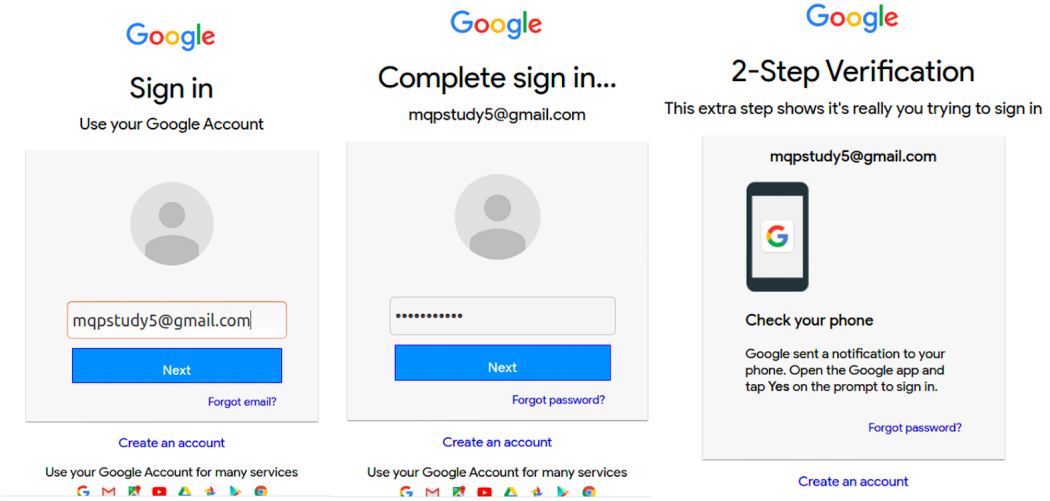

Figure 15: Emulated Google login procedure

MFA implementation	Malicious Endpoint Success?	Packet Pausing Success?
SMS	Yes	Ineffective
Phone call	Yes	Ineffective
E-mail	Yes	Ineffective
TOTP	Yes	Ineffective
push notification with simple approval	Yes	Yes
push notification with code matching	Yes	Ineffective
Yubikey	Ineffective	Ineffective

Table 3: Final attack design selections

ticated via a phone application, as appropriate. While we only tested our attacks against Google, the malicious endpoint attack is not site-specific, and could be expanded to mimic other sites by creating additional frontend pages. We only tested the packet pausing attack against push notifications with simple approval since it has very limited control over the end user's traffic. We deemed the adversary to be successful whenever they paused traffic successfully and the first push notification that arrived on the user's phone corresponded to the attacker's authentication request. As with the malicious endpoint, the packet-pausing attack is not site-specific, and could likely be expanded to work with other sites by analyzing their traffic and adapting the conditions used to begin pausing packets.

The results presented in Table 3 show that our attacks successfully compromise a wide variety of MFA implementations. These results also disprove the common belief that MFA prevents typosquatting attacks. Ameen Chowdhary, in an article entitled "Typosquatting and Mitigating its Consequences," writes, "[MFA] would mean that a hacker has no plausible means of accessing your account unless they have access to your phone or email address" [31]. Our results prove that an attacker can not only access accounts without access to a victim's phone or email address, but that they can do so using a typosquatting attack. Clearly, typosquatting attacks are still a very effective means of compromising accounts, even with MFA enabled.

Furthermore, the results for our packet pausing attack show that attackers can compromise MFA even against observant users who are less susceptible to typosquatting . The packet pausing attack successfully compromises simple approval push notifications, and does so with fewer noticeable symptoms than the malicious endpoint attack. The only symptom visible to the victim during the attack is that the website will appear to hang on the password-entry page when the user receives the push notification. After the victim's packets are unpaused, their browser will proceed to the screen telling them to expect a push notification, and they will receive a second notification. However, by this point, the victim will have already logged the attacker in and the attack will have succeeded. In this case, the victim's potential actions are limited to remediation.

Despite the promising initial results of the packet pausing attack, its use is limited to simple approval-based push notifications. As a result, providers can prevent this attack if they do not offer simple approval push notifications. For example, Microsoft accounts allow users to configure their accounts with code matching push notifications, which display a code on the user's screen that must be matched to one displayed on their phone. This implementation makes it less likely for users to confuse their push notifications with those of an attacker since the approvals require the codes to match. Code-matching push notifications may also reveal a packet pausing attack. The user will not have any codes displayed on their screen when the push notification arrives, and will not know which one to press on their phone. Some users may be suspicious of this and abort the login process, and those who do not will have to guess the correct code. Many code-matching push notifications, including Microsoft's, use three choices, so the odds of guessing correctly is 1 in 3. However, when the packet pausing attack is applicable, it is very effective.

During our lab tests, we also discovered that some identity providers, including Google and Microsoft, have heightened security on their own website's MFA, while other sites that rely on them do not always follow suit. For example, when logging in to Microsoft and Google accounts directly, HSTS prevents an attacker from downgrading the connection to HTTP. This security feature makes it significantly more difficult to compromise accounts directly through these identity providers. On the other hand, some relying parties, such as agame.com, do not protect themselves using HSTS and are vulnerable to both packet pausing and the malicious endpoint. As a result, we conclude that Google and Microsoft accounts are easier to compromise when accessed from a relying party rather than attacking the main account.

While we were unable to compromise YubiKey given time constraints and the complexity of its implementation, we believe that at least a subset of its functionality could be compromised. We believe that this would be the case for the simplest form of YubiKey authentication, where pressing a button on the key causes the key to input a specific sequence of characters as keyboard input. This form of YubiKey authentication could be compromised by a malicious site in a similar fashion to how MFA codes are captured by the malicious endpoint. Since the attacker can read the sequence in plaintext, they can simply submit this stolen key in their own login request to gain access to a victim's account.

Our laboratory study established that the attacks worked in an isolated environment, but we wanted to determine whether the attacks would work against real users. While we were familiar with what symptoms of the attacks manifest during their execution, we wanted to know what symptoms, if any, end users would notice. To answer these questions, we designed a user study to be run on volunteer students at WPI.

5 User Study

To gather data on how well our attacks worked against real users, we conducted independent user studies with 13 volunteers. In order to run the study we first had to get approval from the IRB to ensure we were following the proper guidelines for studies involving human subjects. We had to present to them information such as the methodology of our study, the benefits and drawbacks to the participants, and what we hoped to gain from it. Once we had revised our

study and received IRB approval we recruited from the pool of students currently attending WPI, who we then compensated with 5-dollar gift cards to a coffee shop for participation. We were expecting very few people to be interested, but were surprised to receive many responses indicating interest in participating. We had to limit our sample size due to a combination of limited participants and limited budget for rewarding said participants for their time. Subjects met remotely with researchers through Zoom to conduct the study. Just prior to the meeting, one researcher connected to one of our VMs through SSH, and loaded the VM's GUI on their screen through TigerVNC. Then, in the meeting, that researcher shared their screen through Zoom and allowed the user to remotely control it.

We told our participants that we were studying UI design in order to simultaneously hide the true intentions of our study while putting them in a mindset to pay close attention to the websites. When the study began, we informed users they would be judging an online puzzle game website, and instructed them to log in using a researcher-provided account to track their progress in the games. We used a google account for this purpose as it was available as a SSO option and we had successfully compromised it in the lab study. At no point did we attack any user's personal account(s). When performing the malicious endpoint attack, we first directed the user to the puzzle website, and then asked them to navigate to the Google login page through a link on the site. On the other hand, when performing the packet pausing attack, we instructed the user to log in to Google first, then directed them to the puzzle website once they were authenticated. If at any point the user recognized that they were under attack, we immediately ended the study. As they had to be debriefed to alleviate concerns, and there was no further need to deceive them. If they completed the login procedure without recognizing the attack, we then instructed them to play an online game for a few minutes, while asking them to comment on design aspects that they noticed. This extra step beyond the section we were examining was to ensure the user still believed we were running a UI study when being asked the post-experiment questions. The user's answers to our questions may have changed if they were aware of the true nature of the study.

We ran three different types of attacks in our user study: 1) malicious endpoint with SMS verification, 2) malicious endpoint with push notification verification, and 3) packet pausing on push notification. Due to the similarities between different forms of code-based MFA and our limited access to users, we used SMS as the only form of code-based MFA in our user study.We also considered and tested two potential solutions to help users detect our attacks. These solutions involved presenting users with additional context that we theorized might alert them to the possibility they were under attack.

The primary goal of our user study was to determine if users notice the symptoms of our attack, and if they do, if those symptoms concern them enough for them to cause them to abort the authentication process. We also wanted to determine if our proposed solutions were effective. However, since the study relies on users being unaware that the true purpose of the study is the login procedure and not what they do after the login, we could not perform any tests with users who already knew the purpose of the study.

Since it would be difficult for a solo researcher to run an attack while a user was controlling their screen and mouse, we conducted all of our user studies with at least two researchers present. One researcher acted as the host, guiding the participant through the study, and allowing them to control their screen. The other researcher was thus able to devote their

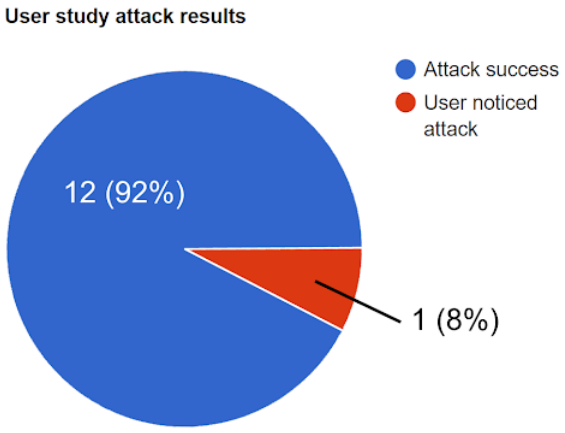

Figure 16: Overall success of attacks during user study

full attention to performing the actual attack. The attacker would assist the main presenter if necessary, but would otherwise just silently run the attack. From the user's perspective, the silent researcher may have seemed like they were simply observing and taking notes. We ended each study when one of two conditions occurred. One such event was if the study participant realized they were under attack. In this situation, so that they knew they were not actually under attack, we immediately informed them of the true purpose of the study. We then proceeded to ask them brief follow-up questions, including whether they preferred using MFA over just a username and password during the study, if they thought MFA is effective for the average user's security needs, and whether or not they noticed anything unusual during the login procedure. See Appendix A for a full list of questions asked in the user study.

Alternatively, if the participant completed the login procedure while unaware they were under attack, we ended the study after they played online games for a few minutes. In such a scenario, we asked users the same follow-up questions, but only revealed the true purpose of the study after they had answered all of the questions. Then, we answered any questions the users had for us, and offered them advice as to how they could make themselves less vulnerable to attack in the future.

We expected to see a significant difference in the results between the malicious endpoint and packet pausing attacks because we believed the packet pausing attack is much harder to detect. The symptoms of packet pausing are fairly common occurrences in daily internet use and can occur for legitimate reasons such as server side issues, dropped packets, and connectivity problems. Furthermore, by the time the second notification arrives, the malicious actor has already compromised the account. Packet pausing is also difficult to detect after the attack has occured. When viewing a network trace of the attack, an end user would need to look for the high jitter caused by pausing the packets. This jitter occurs as a result of the packets arriving at an atypical speed due to the pausing when compared to their normal speed of transmission. Consequently, we hypothesized that the packet pausing attack would be nearly impossible for end users to detect during our user study.

Figure 16 summarizes the results of all attacks during our user study.

5.1 Packet Pausing Attack

The packet pausing attack can only be performed if the attacker is on-path with the victim, the victim has push notification MFA enabled, and the victim's username and password is already compromised. However, if all of these requirements are met, we hypothesized that the attack is difficult to detect. There are only two symptoms that a victim can notice: The webpage will hang on the password-entry screen rather than proceeding to a screen telling the user to expect a push notification. Additionally, the victim will receive a second push notification when their initial request is unpaused. We wanted to verify if our hypothesis about the difficulty of detecting this attack were true.

For both variants of the packet pausing attack, one researcher ran our attack script while the users were entering their password into the appropriate Google login field. At the same time, they began making a login request to Google for the same account, but did not hit "Enter" on the password entry section until the attack script paused the victim's traffic. At that point, the attacker submitted their login request. When the corresponding push notification appeared on their phone screen, the researcher showed it to the victim and asked them what button to press. If the victim told them to accept the login request (which actually corresponded to the attacker's login), the attacker would cease pausing their traffic, causing the victim's login request to be transmitted to Google and resulting in a second push notification being sent. The attacker then presented the second notification to the victim and asked them how to proceed.

5.1.1 Packet Pausing Attack with no additional context

The packet pausing attack without solution attempted to make the attack as hard to notice as possible. The attacker logged in from a system using the same operating system and in the same location as the user. As a result, the only perceivable symptoms would be the webpage hanging on the screen preceding MFA and that a second notification would arrive. We wanted to see whether users would be suspicious of these symptoms. We tested this subsection of our user study on three participants, as summarized in Figure 17.

Users 12, 7, and 4 participated in this part of our study. None of these three users noticed they were under attack nor were suspicious of the second notification. Two of these participants told us in their post-experiment interview that they had received multiple notifications during authentication attempts in the past, indicating to us that receiving a second notification is a common occurrence. Furthermore, User 7 told us that receiving a second notification "seemed pretty standard for Google." We also noticed an interesting difference in the reactions between Users 12 and 7. User 12 considered both notifications closely before telling the researcher to hit "Yes", while User 7 simply said "Good, it's me" almost immediately upon being presented with a second notification, signaling the researcher to proceed. The action User 7 took may indicate that they may view MFA verification as a formality and that they just wanted to login as easily as possible. The second notification is also the point at which we expected users to slow down and become suspicious rather than speed up; this was not confirmed in practice.

When asked about their preference in regards to utilizing MFA versus simply using a username and password, Users 12 and 7 said they prefer having MFA. On the other hand,

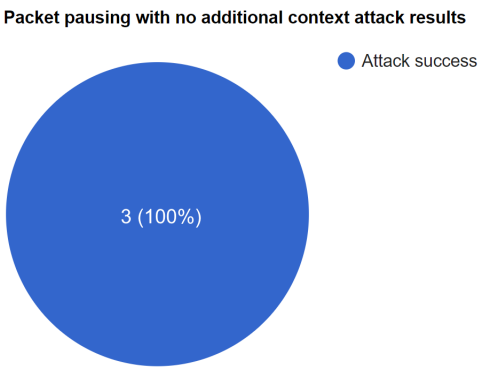

Figure 17: Results of packet pausing attack with no additional context

User 4 preferred to skip MFA, and told us "I would prefer something that is easier, but I recognize the benefit MFA has because of security." This response demonstrates that convenience is a temptation even for users cognizant of the benefits of security. Participants 7 and 12 stated that they would use MFA as we did in the study since they believe Google accounts require extra security, but they would not log into a game website as we did in this study. Both Users 12 and 7 noted that the process of logging in with MFA was not useful for a website whose sole purpose was to quickly hop in and play a game. The combination of these three answers confirms the idea that both in MFA and account use in general, users prefer ease of use whenever possible. All three agreed on the added security benefits of using MFA.

When asked about their beliefs about MFA being useful for the average person to secure their account, all three participants agreed on the usefulness of MFA for situations where the user's password has already been compromised. These users noted that MFA is useful for easily blocking unknown login attempts on their accounts.

5.1.2 Packet Pausing Attack with Additional Context

The packet pausing attacks with additional context began in the same manner as the packet pausing attacks without solution. The main difference between the two was that in the attack with solution, we presented users with our modified push notification images rather than the actual push notification the attacker received (Figures 18, 19, 20). The researcher performing the attack had access to two phones: one phone was configured to receive push notification for the appropriate Google account while the second displayed our solution image. The attacking researcher only presented the second phone to the user, pretending that the second phone had received a push notification, when appropriate. If the user told the researcher to press a button, the researcher acted as if they were pressing a button on the second phone, but actually pressed the corresponding button on the authentic prompt. We aimed to determine if the additional context provided by these messages would serve to alert users to the fact that they were under attack.

In two of the four packet pausing attacks with solution, a researcher showed the participant their phone with the information solution shown in Figure 19 occupying the full screen.

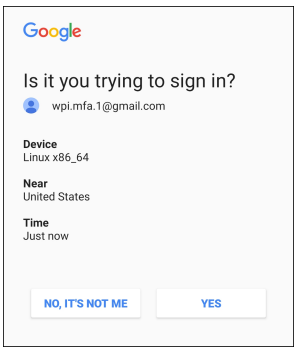

Figure 18: Example of Google MFA push notification

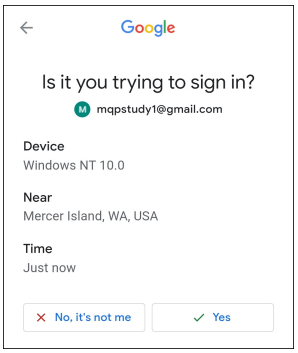

Figure 19: Google MFA push notification with additional information

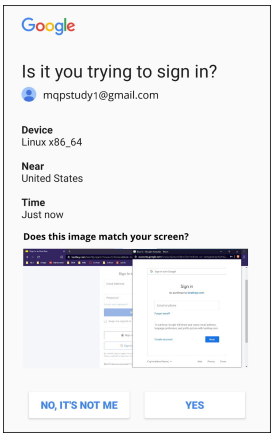

Figure 20: Google MFA push notification with screenshot

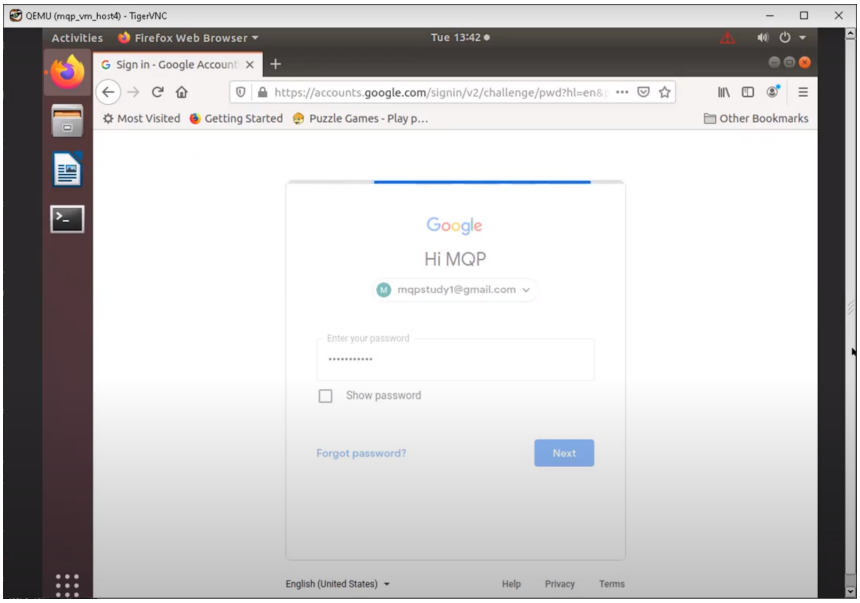

Figure 21: Actual user's view

The image states that the login request was made by a computer running Windows 10 and located in Mercer Island, Washington, whereas the user should expect it to say that it was from a computer running Linux and located in Worcester, Massachusetts.

For the other two packet pausing attacks with solution, the researcher showed the participant the screenshot solution pictured in Figure 20. The screenshot solution contains an image of a computer desktop that does not match the screen the user is looking at (see Figure 21). While the screenshot solution shows an attempt to log in to Google, it is using a different browser, operating system, and screen size. Furthermore, the study participant was on the password-entry page, whereas the screenshot is of the preceding email-entry page, and the screenshot says that they are attempting to log in to an ecommerce website.

Although we had hoped that our proposed solutions would alert users to the fact that they were under attack, we nonetheless successfully attacked all four of the relevant participants, as seen in Figure 22. Regardless of any concerns the users may have had, all four of them ultimately told the researcher to accept the login request. However, we believe the technical restrictions of conducting the user studies remotely may have affected the data.

We performed the packet pausing attack using the information solution with users 11 and 13. After following the study procedure through the point where the researcher presented the user with the push notification, User 11 paused for three seconds before telling the researcher to hit "Yes." When presented with the second push notification, they immediately told the researcher to accept it. While this user did not have their video on, we infer that they used this time to consider the image presented to them. At the same point in the procedure, User 13 told the researcher to press "Yes" without appearing to read the screenshot in detail. Both participants either did not consider the available context to be potentially important enough to read, or read it and decided to proceed anyway.

We performed the packet pausing attack using the additional context with Users 1 and 2. When presented with the image containing the screenshot solution, User 1 spent over a

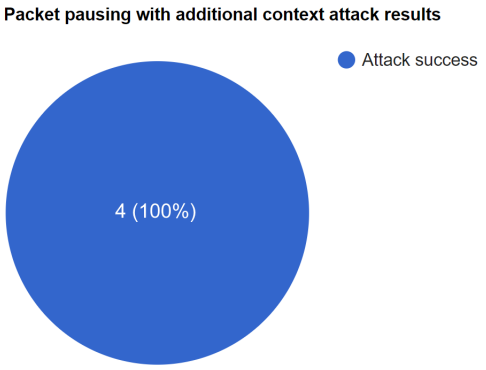

Figure 22: Results of packet pausing attack with additional context

minute examining and, and read "'Does this image match your screen?"' aloud, but still told the researcher to press "Yes." The length of time they spent reading the image caused their paused request to Google to time out, and they were required to restart the login process after their request was unpaused. User 2 similarly spent over a minute trying to read the image, and like User 1, they also eventually told the researcher to press the "Yes" button.

While both of the screenshot solution attacks were successful, we believe that the technical limitations of conducting the experiments through Zoom affected the results. As such, we do not think that either user would have been deceived had they been able to see the phone with the push notification in person. Both participants attempted to analyze the prompt in detail, but were unable to clearly see the small and blurry Zoom window containing the researcher holding up their phone to the camera, as seen in Figure 23. User 1 noted difficulties viewing the picture clearly through Zoom, saying that they "can hardly see the image." While examining the push notification, User 2 said "I can't see the screen inside of it, but I think you can press 'yes'. Wait. Oh! Wait a minute. Yeah, I can't really see the screen inside of it." Furthermore, after the true purpose of the study had been revealed, User 2 noted that the image in the prompt "looks like [their] desktop, like on [their] actual system," which may have contributed to their confusion. They also explicitly stated that if they saw the prompt in person, they would have noticed that it looked completely different from their actual screen.

We considered the possibility that the results for Users 11 and 13 were also affected by the same factor. If so, both of them may have attempted to read the prompt thoroughly, found themselves unable to due to the image's blurriness, and for some reason chose not to voice their concerns to the researchers. However, we do not believe this to be the case. User 13 told the researcher to press "Yes" less than one second after being presented with the image, and did not appear to look closely at the researcher's phone at all. While User 11 took a couple of seconds before responding to the first push notification, the fact that they immediately told the researcher to accept the second push notification indicates that they likewise did not attempt to closely examine the image. These factors suggest that both users experience some form of MFA fatigue, in which they are so accustomed to taking the same action every time they are presented with an MFA prompt that they do not fully consider the consequences of their actions.

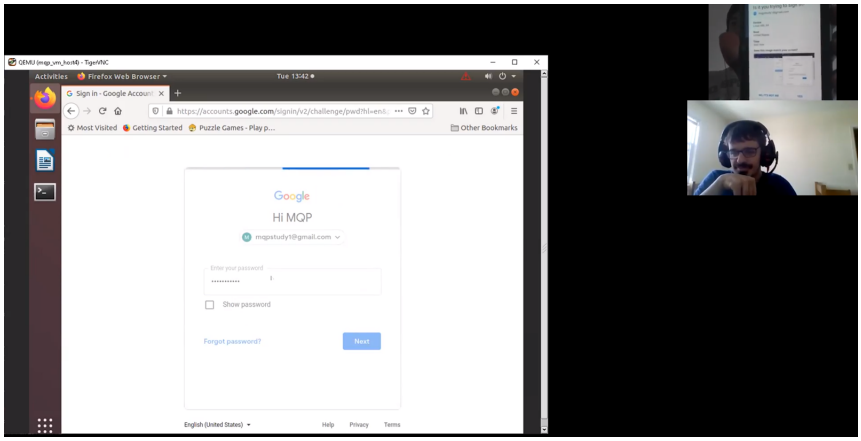

Figure 23: Zoom window as seen during the study (participant's video removed)

When asked if they preferred using MFA over just a username and password for this study, User 1 noted that their request timed out, requiring them to restart the login process, and questioned whether or not the trouble was worth it. They did not describe MFA itself as annoying, however, and would likely have continued logging in if they were using their own account. Users 2, 11, and 13 all said that for this study, they would have preferred just a username and password for expediency's sake. However, both Users 2 and 11 additionally noted that they would have preferred MFA to be enabled if they were logging in to the relying party with their own account and not a researcher-provided one.

Users 1, 2, and 11 believe MFA to be a useful tool for an average person to secure their accounts against attackers. Users 1 and 11 both believe that MFA is "definitely" effective at protecting their accounts. Furthermore, both use MFA with their personal accounts - User 1 protects their password manager with MFA, and User 11 uses MFA to secure their Google and Discord accounts, among others. User 2 was recently convinced of MFA's usefulness by multiple recent incidents in which attackers had attempted to gain access to one or more of their accounts with their username and password, but were stopped by MFA. However, User 2 also noted that having to pull their phone out of their pocket for every login quickly becomes tiresome.

In contrast, User 13 believes that MFA is effective at alerting users to attacks, but was unsure whether or not MFA can effectively prevent attackers from gaining access to their accounts in the first place. They said that they "hate" MFA systems because, while they "understand that it's good that you can monitor if someone's signing into your account," they find the process of "constantly having to sign in, and having to wait for a text message or an email" to be "really annoying." This explanation supports our hypothesis that Users 11 and 13 were simply repeating familiar actions without considering their potential consequences.

5.2 Malicious Endpoint Attack

The malicious endpoint attack differs only slightly from its description in Section 3.3. While we tested the attack against several implementations of MFA in our lab study, we only tested SMS and push notifications during our user study. We did not test phone calls,

Figure 24: Screenshot of typosquatted domain with HTTP lock warning on participant VM running Mozilla Firefox

Figure 25: Malicious endpoint prompting attacker about MFA, then displaying SMS

email, or TOTP codes because all of them have nearly identical user-noticeable symptoms. In particular, we wanted to determine if users notice that the site is served over HTTP, which causes a lock icon to appear in the browser, or that the domain name of the site is a typosquatted version of the real site (Figure 24). These symptoms are the primary methods of noticing that the site is illegitimate and occur regardless of how well the malicious endpoint mimics the user interface of the real login site.

When testing this attack, one researcher controlled the malicious endpoint while the users entered their information into the typosquatted site. The researcher controlling the endpoint began making a login request to Google for the same account, but waited to hit "Enter" on the password entry section. When the user completed the password section of the typosquatted site, the researcher pressed "Enter" on their login request to trigger Google to send a real MFA challenge. At the same time, they examined the appropriate console output to identify the type of MFA used on the account, then displayed the appropriate screen to the user on the typosquatted site (Figure 25). When the MFA challenge arrived, the other researcher showed it to the user, who determined what action to take. For the push notification tests, the user had to either accept or decline the request. If they accepted it, the attacker would automatically be logged in. For the SMS tests, the user had to either enter the code into the website or actively decide not to, aborting the login process. If they entered the code into the site, the malicious endpoint directly reported it to the researcher (Figure 26), who manually entered it into their fraudulent login request. In both cases, the malicious endpoint then redirected the user back to the relying party.

We ran this attack as an on-path adversary, but being on-path is not a requirement of the attack. A hypothetical attacker could register a typosquatted domain, then direct users

Figure 26: Malicious endpoint displaying stolen code and redirecting user

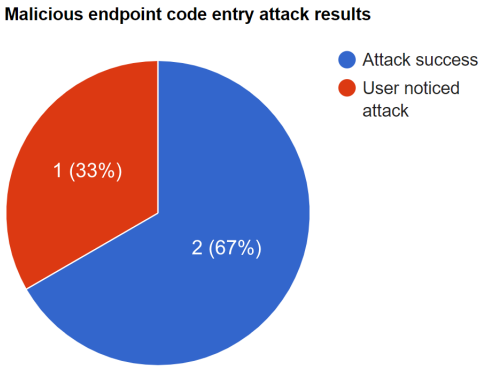

Figure 27: Results of malicious endpoint attack on code entry

there via phishing to achieve the same result. Furthermore, the malicious endpoint attack can compromise more implementations of MFA than the packet pausing attacks, and unlike the packet pausing attack, does not require the attacker to already have the victim's username and password. Despite these advantages, its symptoms are more noticeable. These include only transmitting the webpage through HTTP (and thus triggering any HTTP warning systems on the user's browser), using a misspelled domain (which is visible in the user's URL bar), and potentially having an inconsistent UI when compared to the real site it mimics.

5.2.1 One-time code entry via SMS

The first of our two sets of user studies involving the malicious endpoint attack was done using SMS code-based MFA. We did not test phone call, email, or time-based one-time passwords because all of them are similar to SMS codes in terms of what attack symptoms a user can notice. Any differences between them were thus unlikely to affect our study. As a result, the data we gathered may apply to all of these implementations while limiting the complexity of the user study itself. As outlined in Table 4, only one of three participants noticed that something was suspicious and opted to quit the login sequence before the attack could be completed. The other two participants progressed past the login screen and were successfully deceived by our attack. Of these two participants, one mentioned noticing symptoms of the attack when asked during our post-study interview.

While we were unable to recruit enough volunteers to gather statistically significant results, we did gain insight into what participants did and did not notice during this attack. In particular, we infer that SMS codes do not provide enough context for users to be aware that they were under attack. When considering this, we note how SMS works and what information it provides to the end user. SMS sends a code to the user's phone with no context about what service is requesting the code, nor information about where the request is being made from (Figure 28). This lack of context associated with the code makes it impossible for an end user to determine if the request is fraudulent.

User 9 was the only user across all of our 13 studies who realized that the malicious endpoint was fraudulent. This user noticed that the site was fraudulent after entering only their email address. They told us that some of their accounts had recently been compromised

Participant num.	Noticed attack during experiment	Mentioned noticing symptoms of attack in post-experiment interview?
5	No	No
6	No	Yes
9	Yes	Yes

Table 4: Malicious endpoint (with SMS MFA) results by participant

Figure 28: Google SMS MFA challenge

through a fraudulent login site and they had since educated themself on how to identify fraudulent sites. In particular, they noticed that our malicious endpoint did not have the same UI design animations present on the authentic Google login site. After noticing these inconsistencies, they attempted to click on other links on the login page that we had not configured and discovered the attack.

Our remaining two participants (Users 5 and 6) made it to the step of the attack where they were shown an SMS code. Both told us that MFA gives them a sense of security. Participant 5 said, "I know it's me logging into my account because only I have my phone with my phone number." However, SIM-swapping attacks make it possible for attackers to gain access to a victim's phone number without physical access to the device itself. Additionally, being the only one with access to their phone does not protect users from the malicious endpoint. As the user entered their information into the fake login, we made a simultaneous login request to send the SMS code to their device. This message then arrived at the end user's phone as if it were to correspond to their fake login request. Without any context about how the SMS MFA message was generated, it is unsurprising that these participants proceeded to enter the code into the malicious endpoint without hesitation.

Google is not the only popular identity provider that sends SMS MFA messages lacking in context. Microsoft, another popular identity provider, provides no additional context in their SMS MFA messages (Figure 29). Clearly, delivering MFA codes over SMS without context leaves room for this exploit to work consistently.

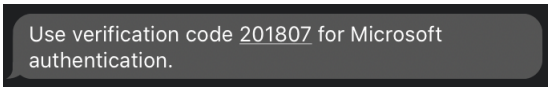

Figure 29: Microsoft SMS MFA challenge

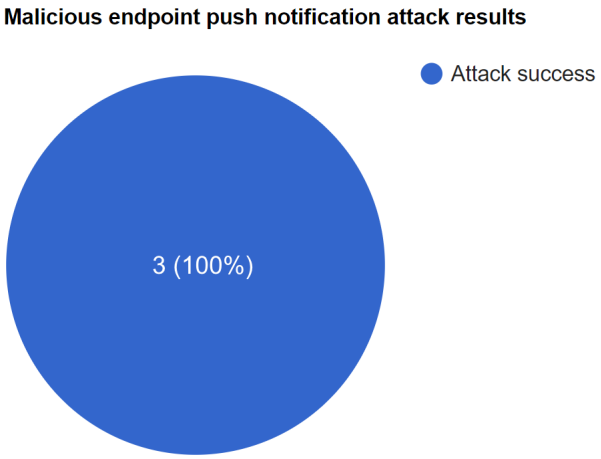

Figure 30: Results of malicious endpoint attack on push notifications

Participant	Noticed attack during experiment?	Mentioned noticing symptoms of attack in post-experiment interview?
3	No	Yes
8	No	No
10	No	No

Table 5: Malicious Endpoint (with push notification MFA) v. participants

5.2.2 Push Notifications

This section analyzes the results of our user study which were based around user's clicking "Yes" when presented with a push notification MFA challenge. We were able to gather results from three participants. With these studies we hoped to determine what symptoms of the attack, if any, participants were most likely to notice and which would raise concern. As summarized in Table 5, all three participants were deceived by this attack and only one mentioned noticing symptoms in their post-experiment interview.

While we were unable to gain statistically significant data based on the limited number of participants, we were able to gather insight into what aspects of push notifications participants paid the most attention to during the attack. In particular, Users 8 and 10 did not appear to read the information on the notification at all and simply told the researcher to click "Yes" to accept the login request.

While many SMS MFA messages contain little or no context about the request that generated the message, push notifications generally do provide much-needed context. As seen in Figure 31, Google's version of these notifications provide information about the OS making the request, the general location of the request, and the time of the request. During the study, we spoofed these fields by making the request from a virtual machine running the same browser and OS version, and in the same general location, as the study participant. By matching the information on the notification to the user's expectation, we successfully

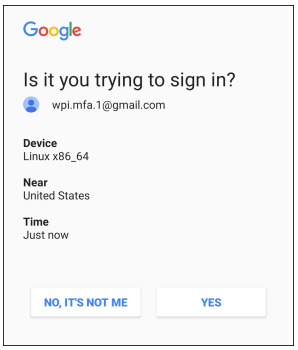

Figure 31: Google push notification MFA challenge

bypass the safeguard of the push notification. We suspect that intentionally matching the relevant criteria between the attacker's computer and the victim's computer was one factor that contributed to our study participants being unable to recognize that the request was malicious and being successfully compromised.

5.2.3 Malicious endpoint symptoms analysis

By conducting the malicious endpoint user studies, we hoped to determine what attack symptoms, if any, end users would notice. In particular, we wanted to know if users would notice that the site was served over insecure HTTP, was a typosquatted domain, and that the site itself was not a perfect replica of the familiar Google login site. When considering the results from both the SMS and push notification user study results, we are able to not only answer these questions, but also draw conclusions about participants' perceptions of MFA.

Interestingly, User 9 was the only participant across all of our user studies who noticed symptoms of the attack and stopped logging in before the account could be successfully compromised. This participant did not notice the typosquatted domain name nor the HTTP lock warning. In fact, none of the six participants in our malicious endpoint study noticed these symptoms. We found this surprising as these attributes are the primary methods of identifying fraudulent sites that follow this threat model and are independent of how good the replication of the user interface design is. However, Spaulding et al. confirm that even experienced users are still vulnerable to typosquatting attacks [24].

Another insight from these participants is that all but User 8 agreed that MFA is an effective way for the average user to secure their accounts. Participant 8 told us that not only did they not find MFA to be effective, but that they found it to be annoying. In contrast, Participant 3 told us, "[MFA] definitely eases some concerns about security of the website and other people gaining access to my information." We found that four of our six participants shared this perspective. It is also interesting to note that of the two participants who did not share this perspective, one felt as though MFA makes security better, but that it is not necessarily the best system. They told us, "[MFA] is an effective way to make [security] better, but there are still ways to make it even better." MFA does provide added security, but it should not be trusted as an unbreakable system. Based on these perceptions, we conclude that MFA provides end users with a sense of added security versus using just a

username and password.

When analyzing the perceptions of MFA voiced by our participants, we also considered whether or not participants put too much trust into MFA. We found that some participants had their suspicions lessened while interacting with the malicious site after they saw that MFA was enabled. In particular, Participant 3 was very hesitant to use a Google account to sign into an unrelated website due to privacy concerns. However, as soon as they arrived at the fake MFA screen, their concerns subsided. They later told us, "Had I done this with my email and received an authentication [notification]... that would have eased my concerns. When you get that [authentication request], it gives you proof of security." This comment is interesting to consider because MFA seems to have given this participant an excessive sense of security. It is also important to note that the participants are students at WPI and are required to use MFA on their university accounts. Additionally, WPI has sent out information to students indicating that MFA enhances security. These factors may have affected user's impressions of MFA. Future studies may be needed to determine the extent to which this perspective is shared by computer users in general.

6 Concluding Discussion and Remarks

Our laboratory and user studies, described in Section 4 and 5 respectively, demonstrated that many MFA implementations are susceptible to either or both of the packet pausing and malicious endpoint attacks. In particular, the packet pausing attack is effective only against certain push notification MFA implementations while the malicious endpoint is effective against SMS, email, phone call, and time-based one-time password codes in addition to all types of push notifications. Through our user study, we found that some users found that the MFA process was inconvenient. However, nearly all of these users felt a greater sense of security with MFA. When considering the wide variety of MFA implementations that are susceptible to our threat models, along with the fact that users were generally unable to detect these attacks, the attacks may pose serious risks.

We also found that our user study results corroborated our hypothesis that the malicious endpoint attack is more likely to be caught by end users. While we expected end users to notice the browser warning about HTTP or the typosquatted domain name, the only user who identified the site as fraudulent noticed as a consequence of UI design inconsistencies between our fake site and the real one. On the other hand, the packet pausing attacks were much more successful than we anticipated. We provided additional context to the end user with the hope that users would notice inconsistencies between their actions and the information in the push notification, and would thus be able to determine that they were under attack. However, we suspect that difficulties with image resolution over Zoom was the primary factor resulting in the failure of our screenshot solution and had participant feedback suggesting that it would be effective had the users been able to see it in person.

Additionally, we found that users tended to not pay close attention to the MFA stage of the login process. While we noticed this behavior most often during the packet pausing attack, we observed it during the malicious endpoint attacks as well. We believe that this behavior may be a result of MFA fatigue, which may occur when users are so accustomed to taking the same action every time they are presented with an MFA prompt that they do not

fully consider the consequences of their actions. We noticed this phenomenon more often with users who said that they found MFA to be annoying or an obstacle more than anything else. More research is needed to determine if MFA fatigue is indeed a problem, and if so, how prevalent it is, what factors contribute to it, and what steps MFA providers can take to combat it.

Future research in this area should explore solutions that provide greater context to end users. With more context, users will have more information at their disposal when determining the legitimacy of login attempts. We also recommend that future researchers gather more user data to validate. By gathering more user data, researchers will also be able to gain greater insight into the problem of MFA fatigue, which may be useful in determining how to make users pay greater attention to their security. Finally, we recommend that future researchers attempt to apply similar methods to break USB key implementations of MFA such as Yubikey.

7 Appendices

7.1 Appendix A - User Study Interview Questions

- How would you describe your overall experience using the website?
- Have you used any multi-factor authentication (MFA) systems before? If so, which ones?
- When authenticating yourself for the study today, did you prefer using MFA over just a username and password?
- Do you think that MFA is effective for the average person's security needs?
- Did you notice anything out of the ordinary during the login process today?
- (If yes to previous question) Would that have made you stop logging in if you were using your own account?
- Do you have any other comments, concerns, etc.?

References

[1] M. Dubie. (2020) New LastPass study finds identity and access management is critical to securing a remote workforce. https://blog.lastpass.com/2020/06/new-lastpass-study-finds-identity-and-access-management-is-critical-to-securing-a-remote-workforce/.

[2] "Demographics of internet and home broadband usage in the United States," https://www.pewresearch.org/internet/fact-sheet/internet-broadband/, 2021.

[3] G. Kessler. (1996) Passwords strengths and weaknesses. https://www.garykessler.net/library/password.html.

[4] J. Still. (2021) What is Microsoft Authenticator? here's what you need to know about the two-factor authentication app that can secure your online accounts. https://www.businessinsider.com/what-is-microsoft-authenticator.

[5] A. Hashim. (2024) Dangers of reusing passwords know why is it bad and how you can avoid it. https://privacysavvy.com/password/guides/reusing-passwords/.

[6] K. Bankston and R. Schulman. (2021) Case study #2: Offering two-factor authentication. https://www.newamerica.org/in-depth/getting-internet-companies-do-right-thing/case-study-2-offering-two-factor-authentication/.

[7] X. Huang, Y. Xiang, A. Chonka, J. Zhou, and R. H. Deng, "A generic framework for three-factor authentication: Preserving security and privacy in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, p. 13901397, Aug. 2011. [Online]. Available: https://doi.org/10.1109/TPDS.2010.206

[8] S. Alder, "57% rely on multi-factor authentication to improve security but mfa is not infallible," *HIPAA Journal*, 2019. [Online]. Available: https://www.hipaajournal.com/57-rely-on-multi-factor-authentication-to-improve-security-but-mfa-is-not-infallible/

[9] C. Cimpanu, "Microsoft: 99.9% of compromised accounts did not use multi-factor authentication," *HIPAA Journal*, 2019.

[10] ——. (2019) FBI warns about attacks that bypass multi-factor authentication (MFA). https://www.zdnet.com/article/fbi-warns-about-attacks-that-bypass-multi-factor-authentication-mfa/.

[11] "Google Authenticator WordPress two factor authentication (2fa , mfa, otp sms and email) passwordless login," https://wordpress.org/plugins/miniorange-2-factor-authentication/.

[12] "Two-factor authentication (2FA) app & guides," https://authy.com/.

[13] "YubiKey strong two factor authentication." [Online]. Available: https://www.yubico.com/

[14] "The best security key for multi-factor authentication," https://www.nytimes.com/wirecutter/reviews/best-security-keys/, 2020.

[15] "Works with YubiKey catalog," https://www.yubico.com/works-with-yubikey/catalog/, 2020.

[16] J. Reynolds. A tale of two studies: The best and worst of YubiKey usability. "http://www.youtube.com/watch?v=ugD-YxDuSX8". YouTube.

[17] "How FIDO works - standard public key cryptography & user privacy," https://fidoalliance.org/how-fido-works/, 2019.

[18] "FIDO2: Moving the world beyond passwords using webauthn & ctap," https://fidoalliance.org/fido2/, 2020.

[19] "HTTP Strict Transport Security," https://https.cio.gov/hsts/.

[20] "Usage statistics of HTTP Strict Transport Security for websites," https://w3techs.com/technologies/details/ce-hsts, 2021.

[21] "Fake calls from Apple and Amazon support: What you need to know," https://www.consumer.ftc.gov/blog/2020/12/fake-calls-apple-and-amazon-support-what-you-need-know, 2020.

[22] "Sim swap scams: How to protect yourself," https://www.consumer.ftc.gov/blog/2019/10/sim-swap-scams-how-protect-yourself, 2019.

[23] "Pricing — political robocall & rvm pricing," https://robocent.com/pricing/robocent-pricing/, 2021.

[24] J. Spaulding, D. Nyang, and A. Mohaisen, "Understanding the effectiveness of typosquatting techniques," in *Proceedings of the Fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies*, ser. HotWeb '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3132465.3132467

[25] P. Fasulo. (2020) What is typosquatting and why is it a risk to your organization? https://securityscorecard.com/blog/what-is-typosquatting-why-are-typosquats-a-risk-to-your-organization.

[26] A. B. J. Hodges, C. Jackson. (2012) What is typosquatting and why is it a risk to your organization? https://www.hjp.at/doc/rfc/rfc6797.html.

[27] "Analysis of the Alexa top 1m sites," https://blog.mozilla.org/security/2018/02/28/analysis-alexa-top-1m-sites-2/, 2018.

[28] D. Roethlisberger. (2021) Sslsplit - transparent ssl/tls interception. https://www.roe.ch/SSLsplit.

[29] "sslstrip," https://github.com/moxie0/sslstrip.

[30] "bettercap," https://www.bettercap.org/.

[31] A. Chowdhary. (2020) Typosquatting and mitigating its consequences. https://tec-refresh.com/typosquatting-and-mitigating-its-consequences/.