Exploring Phone-Based Authentication Vulnerabilities in Single Sign-On Systems

Matthew M. Tolbert, Elie M. Hess, Mattheus C. Nascimento, Yunsen Lei, and Craig A. Shue

Worcester Polytechnic Institute, Worcester MA 01609, USA
{mmtolbert, emhess, mcnascimento, ylei3, cshue}@wpi.edu

Abstract. Phone-based authenticators (PBAs) are commonly incorporated into multi-factor authentication and passwordless login schemes for corporate networks and systems. These systems require users to prove that they possess a phone or phone number associated with an account. The out-of-band nature of PBAs and their security may not be well understood by users. Further, the frequency of PBA prompts may desensitize users and lead to increased susceptibility to phishing or social engineering. We explore such risks to PBAs by exploring PBA implementation options and two types of attacks. When employed with a real-world PBA system, we found the symptoms of such attacks were subtle. A subsequent user study revealed that none of our participants noticed the attack symptoms, highlighting the limitations and risks associated with PBAs.

1 Introduction

To authenticate users, some organizations combine traditional passwords with other verification mechanisms in a multi-factor authentication (MFA) scheme. Others eliminate passwords entirely and use passwordless authentication mechanisms. Both MFA and passwordless schemes can use a proof-of-possession authentication factor in which the end user must prove physical access to a device. Phone-based authenticator (PBA) systems are commonly used for proof-of-possession schemes since users often already have and protect smartphones. These PBA systems require a user to promptly interact with the phone associated with an account. If the user completes that interaction successfully, the system approves the authentication attempt.

PBAs are widespread in MFA schemes associated with financial institutions [31] and online account providers [5]. Experts and vendors have encouraged broader use of PBAs with the promise of reducing account compromise risks [25]. Based on industry surveys [28] and legal directives [16], PBA use is expected to grow in the future.

PBAs are commonly paired with single sign-on (SSO) systems [4] in which an identity provider authenticates users for a set of relying parties. However, if SSO implementations do not cache credentials across applications or are improperly tuned, they may frequently prompt users to authenticate using PBAs [26]. Prior

work in usable security has found that repetitive warnings and confirmations can desensitize users to the importance of the security decisions they are making [6]. This may enable adversaries to deceive users into risky behavior.

In this work, we ask: *To what extent can adversaries deceive end users into authorizing malicious behavior via phone-based mechanisms (e.g., SMS OTP, email OTP, push notifications)? What phone-based authentication mechanisms have greater risk and what symptoms result? Do end users notice these symptoms? Would additional context help end users distinguish malicious phone-based authentication interactions?*

We explore some common PBA configuration options and their implications using an empirical study with a popular production SSO system. We implement techniques to undermine the PBA system and measure their effectiveness. This leads to the following contributions:

- **Exploration of PBA Settings in Two Attack Scenarios:** We explore a range of PBA implementation options and their potential vulnerabilities. We implement two attacks on PBAs: one using a malicious SSO relying party and one using network packet profiling and strategic delay. We find the malicious SSO relying party can compromise each tested PBA option. The profiling and timing attack is effective against application-based approval prompts. The observable characteristics of both attacks appear to be subtle.
- **Report of User Study on Attack Effectiveness:** We conduct an IRB-approved user study with 13 participants to determine if people notice the authenticator attacks when they occur. We found that 12 participants did not notice the attacks, while the last participant was excluded by our testing protocol before reaching the PBA attack test. Our observations and participant reports indicate only cursory review of PBA prompts and notices, providing ample opportunity for adversary deception.

2 Background and Related Work

Our work combines phone-based authentication, social engineering, and deception with computer users' perception and management of risk. To implement our tools, we use established networking techniques. Accordingly, we review background and prior work in each of these areas. To the best of our knowledge, **we are the first research work to explore attacks that undermine phone-based authenticators without compromising the user's endpoint device, their phone, or the phone's connection (e.g., SIM-swapping)**.

Multi-factor authentication (MFA) schemes often consider what the user knows, what the user possesses, and what the user is as different authentication factors [12]. Some organizations, such as Microsoft, indicate that MFA can prevent more than 97% of identity-based breaches [25]. A study on a data set of Google account authentication records found that device-based second-factor authentication blocks more than 90% of account compromise attempts [14]. However, prior work indicates that MFA has several usability challenges that affect

its adoption rate [11]. Ease-of-use, required cognitive effort, and trustworthiness are three major factors that affect MFA's usability [9]. To improve the adoption of MFA, Das et al. [10] conducted a usability study on the Yubico Security Key and observed user difficulties in configuring and using the technology.

Prior research has examined mechanisms to compromise PBAs by compromising the user's endpoint device, their phone, or the communication channel with the phone. The simplest mechanism intercepts unencrypted text messages to phones by falsely registering a device (e.g., "SIM swapping attacks") or by network operators [20], [24]. Alternatively, Konoth et al. [23] explore a scenario where an attacker has compromised the endpoint, including the user's browser, and synchronizes with a SMS-stealing application on the user's phone. More recent report [18] showed that scammers create a fake surveys on behalf of reputable companies to mislead users into scan QR code and falsely authenticate online services. Our work explores a simpler attack scenario that does not require a compromise of the user's phone, phone connection, or endpoint.

Phishing is a type of social engineering attack that deceives users into providing their personal information. An attacker can create a fake website to facilitate phishing. Attackers may rely on victims' lack of understanding of URL components to deceive victims with little effort to disguise the destination site [19]. In a user study, Dhamija et al. [13] asked participants to evaluate a website for symptoms of fraud. They found victims of impersonation attacks often only consider the content of a webpage to determine its legitimacy and few considered SSL indicators.

Security warnings are widely used to convey risk. However, prior research found users often ignore these warnings due to a lack understanding of the jargon [34] or habituation effects [2]. Akhawe et al. conducted a field study [1] to examine different types of browser warnings and their click-through rates. They found that malware and phishing warnings have a low click-through rate, while SSL warnings can have high click-through rates, depending on the warning's interface design. Later work [17] examined a new design for SSL warnings to improve adherence via simple, non-technical text and promoting a clear cause of action. To examine how habituation affects disregard for security warnings [33], a user study found that participants who learned to ignore warnings in one task were likely to ignore security warnings in a subsequent task. To combat such habituation, researchers proposed using polymorphic dialogues that continuously change the form of user required input [8] or interface appearance [3] to require user attention for security decisions. We explore the impact of PBA prompt messages and the user perception of these PBA prompts.

3 Understanding PBA Goals, Options, and Impacts

Phone-based authenticator (PBA) systems attempt to validate a user's identity by verifying that the user physically possesses a smartphone associated with their account. A typical SSO authentication session using PBA is illustrated in Figure 1. In it, a user visits the relying party's website, and when the user wants

to authenticate, the relying party redirects the user's browser to the identity provider site. Most identity providers have authentication APIs for relying parties to integrate the login processes into their applications. The identity provider then prompts the user for the credential. Upon validating the user's credentials, the identity provider then issues the PBA challenge to the user. The challenge is typically a task with specific instructions that can be completed only through or with the user's phone. For instance, the challenge might ask the user to input a nonce (i.e., a single-use value) that is only transmitted to the user's phone or ask the user to approve the login using an authenticator application installed on the user's phone. In some options, the response to the challenge is sent via the browser; in other options, it is sent via the user's phone. After successful completion of the PBA challenge, the identity provider redirects the user's browser back to the content provider, along with a token. The token both specifies the user and proves that user's identity.

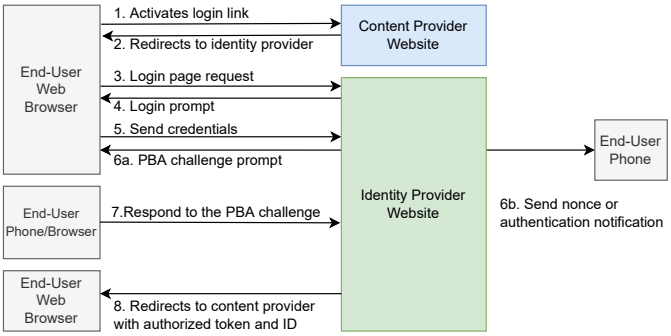

Fig. 1: A general PBA workflow without an attacker

We explored a set of phone-based authenticators as shown in Table 1. The first four entries require the end user to obtain a nonce value and to supply that nonce via the device being authenticated[1]. The underlying mechanism to share the value varies: it can be transmitted via an SMS text message, through a phone call in which an automated system verbally provides a string of numbers, through an email with a code, or through output in a phone application (often implemented via a time-based, one-time password). The next two PBA mechanisms do not require the end user to supply the nonce. In the code matching scenario, the end user selects a value in their phone application that matches what is displayed via the login prompt on the browser. This action links the approval with the active browser session. The "approve" request scenario omits the number matching requirement and simply asks the end user to press an "approve" (or similarly labeled) button on the phone to approve a request; how-

[1] We refer to the device being authenticated as "the browser," for simplicity. However, this approach can also be embedded within other application types.

ever, this option lack an association between the browser session and the phone's prompt. This leads to a timing vulnerability that we explore on its own.

These PBA systems make two key assumptions: 1) the nonce will not be revealed to an adversary and 2) the legitimate user will only respond to the challenge of their own authentication attempts. However, if either assumption is violated, the PBA system will fail to achieve its authentication goals. In the remainder of this section, we discuss the threat model and how an adversary may violate the PBA assumptions to gain unauthorized access.

Table 1: Attack effectiveness by PBA implementation method

PBA Implementation	Responding Device	Implementation Defeated by Malicious Site?	Timing Attack?
Code via SMS	Browser	Yes	Not tested
Code via Phone Call	Browser	Yes	Not tested
Code via E-mail	Browser	Yes	Not tested
App-based One-time-Code	Browser	Yes	Not tested
App-based Code Matching	Phone	Yes	Not tested
App-based "Approve" Request	Phone	Yes	Yes

3.1 Threat Model and Experiment Setup

We scope our focus to attacks on PBA systems in SSO environments. We assume that the user's phone, device, and the identity provider are not compromised. Further, we constrain the adversary such that it does not have access to the user's phone connection. We assume the adversary's goal is to defeat the PBA factor itself and either has already defeated other authentication factors (e.g., passwords) or does not need to (e.g., a single-factor PBA system).

In one scenario, which we label the "Malicious Site" scenario, the adversary interacts with the user as a malicious SSO relying party. This scenario is consistent with a phishing attack in which an adversary successfully lures a user to a phishing website that impersonates a legitimate web site that uses a specific identity provider. In our second scenario, which we label the "Timing Attack" scenario, the adversary is on path between the user's browser and the SSO identity provider and is able to see and delay/drop packets between those endpoints and to interact with the SSO provider on its own; however, it is unable to decrypt or forge packets belonging to the browser or identity provider. The Timing Attack scenario is consistent with an adversary running a malicious public WiFi network [7], that has compromised the user's residential router [27], or is naturally on-path (e.g., an ISP or nation-state adversary).

For clarity, we describe scenarios in which a user is attempting to authenticate on a client device (e.g., a desktop/laptop computer or tablet), which we refer to as "the browser," and performs the PBA step on a separate device (e.g., a phone). These actions could be done on the same device; if so, one must relax the adversary constraint against having access to the phone's connection.

In the remainder of this section, we explore these attack scenarios with different PBA implementation options. We do so using an industry-leading SSO

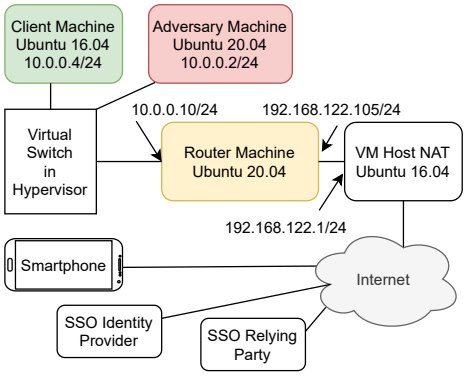

Fig. 2: Our experimental network.

identity provider which has 40% of the identity provider market share and is used by around 80,000 companies globally. We refer to this vendor as *AnonSSO*; we use a pseudonym for the vendor because the vendor employs current industry best practices and the vulnerabilities are inherently due to PBA options themselves, not due to the vendor's implementation. The AnonSSO vendor's approach is representative of other implementations, and there were no implementation-specific details that would prevent the results from generalizing to other implementations. Our study was approved by our Institutional Review Board (IRB) and was conducted with careful attention to ethical conduct. As we further explain in subsequent sections, our scenarios do not harm or attempt to compromise the AnonSSO system.

We perform our experiments using a set of virtual machines that are connected with the AnonSSO system via a bridged network interface. Figure 2 shows our experimental network setup for conducting PBA attacks. We host three virtual machines on a VM server. One virtual machine (top left) acts as the legitimate client, another acts as an adversary for the malicious website (top right), and a third acts as a router that is benign in the Malicious Site scenario but is adversary-controlled in the Timing Attack scenario. These three virtual machines are connected through a virtual bridge created by the physical machine's hypervisor. The router is configured with two interfaces: one to the virtual bridge and one to the hypervisor's network card via a NAT interface. The router provides connectivity to the AnonSSO authentication portals. A smartphone associated with the legitimate user connects directly to the Internet.

3.2 Impact of Malicious Relying Party Sites

Adversaries have had success in luring users into visiting malicious sites [15] and impersonating legitimate entities [32]. Previous work shown that advanced phishing toolkits [22] can mimic the site with high fidelity, which simplifies this process for adversaries.

Accordingly, we explore a scenario in which an adversary creates a malicious relying party website that purports to redirect the user to an identity provider, but actually does not do so. Instead, the malicious site impersonates the identity provider and prompts users to enter their credentials. If the user does not notice the deception, they may submit this information, providing it to the adversary. Upon receiving the user's credentials, the malicious site covertly initiates a connection to the identity provider as if it were a client. It impersonates the end-user and supplies the credentials it obtained to the actual identity provider. This process triggers the identity provider to send the PBA challenge (and transmit the nonce if necessary) that asks the adversary to follow specific instructions. The adversary uses the malicious site to relay those instructions to the real user. The user, who may incorrectly believe they are in the middle of a valid authentication attempt, may follow the instructions (e.g., by echoing the nonce to the adversary's malicious site or through a phone-based application) to respond to the PBA challenge. By doing so, the end-user effectively authorizes the adversary's authentication attempt rather than its own (either by revealing the nonce to the adversary or approving the adversary's login in the authenticator application, which violates both the first and second assumption discussed in Section 3). The adversary succeeds whenever the user authorizes the adversary's login.

As shown in Table 1, the PBA implementation method may require the adversary to relay PBA instructions or request user inputs via the malicious site. In other scenarios, the user may directly interact with their phone's application without requiring the adversary to issue a prompt. Our user study in Section 4.3 suggests that adversaries could deceive users into performing these actions.

We test the attack scenario using a legitimate client machine (top left in Figure 2) to connect to a malicious website (the adversary machine, top right in Figure 2). Both the client and adversary have unimpeded access to the Internet (i.e., the router machine, center of Figure 2, forward traffic without manipulation or delay). The malicious relying party uses HTTP communication between itself and the client. It uses a custom set of login pages to mimic the login process of the identity provider while displaying user-supplied credentials and nonce values to the adversary. As mentioned above, the adversary must perform its own login attempt quickly upon receiving credentials, but this can be accomplished with an automated process (e.g., using web browser automation tools such as Selenium [30]). We omit this automation step since it has previously been explored. The nonce received by adversaries is valid for multiple minutes and remains valid during the entire attack process. Our tests confirm that an adversary can implement the scenario in a straightforward manner with few observable symptoms.

3.3 Timing Attacks on Unassociated PBA Approvals

The process of using an SSO protocol results in a specific traffic pattern involving redirection of a client from a relying party to an identity provider and back. An on-path adversary may examine traffic to determine such patterns, leveraging DNS requests to identify the servers involved with relying parties and identity providers. By recording such network traffic and browser actions, adversaries

can build a database of actions for each authentication step. The adversary can later use that database when monitoring a target's traffic to time an attack.

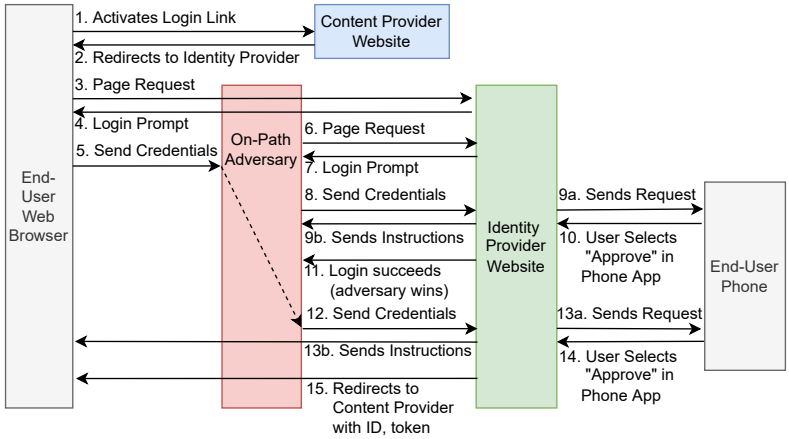

Fig. 3: An on-path adversary launching a timing attack on a PBA workflow

In the timing attack, the adversary monitors all connections from a target to a relying party. The adversary matches the target's packets to each known step in the authentication process. Once the target reaches the step that transmits login credentials, the on-path adversary then can queue the victim's packets and submit its own login request using previously-obtained credentials, as shown in Figure 3. If the login system uses push-based authentication via a phone-based application, the adversary's attempt to log in will create a notification to the end user's phone asking for approval. Since the prompt appears at the expected time during the user own authentication attempt, the user may approve it. However, in doing so, the end user authorizes the adversary's authentication attempt instead, violating the second PBA security assumption discussed in Section 3.

For the timing attack to work, the PBA approval process on the phone must not be explicitly linked with the browser login session. Only the last implementation method in Table 1, the app-based "approve" request, meets this requirement. In that workflow, the "approve" button does not provide context for what session is being approved nor does it require the user to supply a unique identifier (such as a nonce or matching code). This ambiguity allows an adversary to delay and reorder interactions to gain access.

In our experiments, the adversary controls the router machine depicted in the center of Figure 2. The adversary machine (bottom left) is not involved in this attack. The adversary pre-profiles the relying party interaction with AnonSSO and creates an annotated database with packet sizes for each event. AnonSSO prompts for a username associated with the account and ask for password in a second page. The adversary builds a transition map for the initial authentication

page request, the submission of a username, and the submission of a password. For each TCP flow to AnonSSO, it tracks the number of packets and total bytes transmitted to distinguish the password submission step for pausing the legitimate transaction. Since identity providers like AnonSSO provide a uniform API for relying parties, the process can generalize across relying parties.

The client accesses a legitimate relying party website and the AnonSSO infrastructure via the router. The client and servers use TLS, so the router does not have access to the plain-text communication or the ability to forge messages. Using the router machine's built-in `iptables` firewall, we direct all communication between the client and AnonSSO endpoints to a specified queue in the Linux `netfilter` architecture. With a Python script, we use the `NetfilterQueue` Python library [21] to dequeue all the packets in the kernel queue. We then use the `Scapy` [29] tool to dissect the packet headers of the obtained packets to extract host names from the DNS response packets and associate each HTTPS packet's destination IP with a host name. For each intercepted packet, we track the cumulative packets and bytes transmitted to identify which stage of the authentication process the user is in. Once the requisite transmission occurs to send the username, we know the next transmission will be the password submission. Before reaching the password submission stage, we simply forward each packet. However, once we receive the first packet associated with password submission, we queue it and all subsequent traffic in that flow. The adversary then performs an out-of-band login, which sends a PBA approval request to the phone.

3.4 Observable Characteristics of the Attack Scenarios

The Malicious Relying Party Site and the Timing Attack scenarios have subtle symptoms. We describe these symptoms in this section. Our user study in Section 4 found that our participants did not have concerns about these symptoms.

For the malicious relying party scenario, the primary non-adversary controlled symptom of the attack is the lack of redirection from the relying party to an identity provider in the browser's address bar. Otherwise, an adversary can convincingly replicate the page visuals to mimic a legitimate relying party and the identity provider. An adversary may choose to continue its deception after succeeding (i.e., after the user completes the PBA process to authorize the adversary) by redirecting the user to the legitimate relying party's site. The user may notice they are not logged in and may retry the process. The user may incorrectly believe the site had an issue performing the login rather than recognizing that they had been attacked.

For the timing attack scenario, the primary non-adversary controlled symptom of the attack is that the browser will not provide confirmation of password submission and a prompt to complete the PBA process before the PBA application prompts the user for approval. Until the user proceeds with the PBA approval, the browser will appear as if it is awaiting a response from the identity provider. After the user approves the attack, the adversary can choose to drop or deliver the queued packets. If dropped, the web request will time out. If

delivered, the user will receive a second PBA prompt, which if completed, will authorize the user's access.

In our own personal usage of PBAs, we occasionally receive duplicate PBA prompts for a login attempt or must retry a PBA attempt to successfully sign-in. While the causes of these scenarios are unclear, we suspect network transmission issues or server-related errors. In our user study (Section 4.4), we explore whether our human subjects have had similar experiences. We find that they did, and that these experiences seem to desensitize participants to such PBA attack symptoms.

4 User Study and Findings

Given the subtle symptoms of the PBA attacks, we next explore whether users notice them and whether the symptoms raise concerns. To do so, we conducted a user study to gather participants' impressions. We recruited 13 participants, conducted the study, and debriefed each participant to understand their actions. We found that the symptoms of the PBA attacks did not concern the participants.

4.1 IRB Process and Participant Recruiting

We used our organization's Institutional Review Board (IRB) to ensure appropriate protections for our human subjects. The main concern in our study was the use of distractions and ambiguity. Our goal was to measure participants' responses to phone-based security prompts on an account, without biasing the results by revealing that we were specifically monitoring their security decisions.

In our informed consent process, we indicated that our study would explore "how website design affects [the] user experience" and that the study would "measure how various design choices affect how easily and quickly a user notices that information being presented to them is important." The protocol procedures indicated that participants would use video conferencing and screen sharing software to log in to a puzzle website, complete several puzzles, and review the results. The participants received a $5 USD gift card incentive.

We recruited participants via email. Our participants were undergraduate Computer Science students, which may result in biases making them more sensitive to computing details that could reveal a security risk.

4.2 Experimental Setup

Our participants met with the researchers via video conferencing, which was necessary safety consideration during a high propagation phase of the COVID-19 pandemic. The subjects were experienced with video conferencing and, except where noted, we do not believe that the format affected the study.

The researchers used the same experimental VM infrastructure used in Section 3.1 using shared screen control software that enables remote control for participants. The researchers then allowed the human subjects to control that VM system via the video conferencing software. The participants were told that

as part of our user interface study, the participant would need to remotely control the entire process. It was pointed out that the system may need to use smartphones, and if so, the researchers would hold the phone up to their video conferencing camera and ask the participant how to proceed.

The participants were asked to provide careful feedback about the website associated with a puzzle game. They were asked to log in to the site and were supplied with a researcher-provided account and were told that doing so would enable tracking their progress in the game. For the identity provider, we used the same vendor as in Section 3. The researchers ensured that participants used the researcher-provided system and credentials to avoid risk to participants.

If at any point the user expressed security concerns, we immediately ended the study to allow us to debrief the individual and address those concerns. If the participant completed the authentication process without expressing concern, we asked them to play an online game for a few minutes while commenting on any design aspects that they noticed. The requested commentary was to focus users on the website rather than the authentication process during our interview.

During the experiment, one researcher acted as the host and guided the participant through the study. The host allowed the participant to control their screen and VM through the conferencing software. Another researcher focused on performing the attack on the VM system while it was controlled by the participant. If the host needed assistance, the second researcher would provide it. Otherwise, the second researcher remained silent as if they were an observer taking notes, while actually performing the adversary actions in the experiment.

Except as noted, each interactive segment with participants ended with a short interview. We asked some questions before informing the participant of the focus on PBAs and asked others afterward. The researchers answered any questions for the participants, offered online account security advice, and then ended the session. We split the participants into groups to study both the malicious website and the packet delay attacks.

4.3 Participant Responses to the Malicious Relying Party Scenario

In the malicious website attack, we explored both SMS-based delivery of nonce values and application-based approval verification mechanisms. We omitted exploration of code delivery via email, audio phone calls, or application-based code display since they have similar user-facing characteristics as the SMS-based delivery of nonce values.

In our experiments, we created a site that was delivered over HTTP with a similar-looking URL host name (in which the period character was missing from a host name, resulting in the concatenation of a domain and host-name in the domain portion of the URL). When the host directed the participant to log in to the game site, the malicious website showed a fake variant of the Anon-SSO authentication portal. As the user entered their credentials, the adversary observed the console of the website. Once the user submitted authentication credentials, the website displayed the credentials to the adversary's console and took no subsequent action. The adversary viewed those credentials and quickly

submitted a separate login attempt to the real AnonSSO authentication portal with the participant-supplied credentials. The adversary's action caused the actual AnonSSO system to send a PBA request to the host's phone.

When the PBA challenge appeared on the host's phone, the host displayed the prompt to the participant. The participant either had to choose to proceed or abort, in the case of the application-based approval process, or to enter the displayed code into the website for the SMS-based code delivery option. If the participant chose to proceed in application-based approach, the adversary's attempt was authorized. Likewise, if the participant typed in the correct code into the malicious website, it was displayed to the adversary via the website's console and the adversary could then enter the code to log in. Both of these outcomes were considered successful attacks. If the participant chose to abort the log in, the attempt was considered an unsuccessful attack.

Table 2: User study results indicating whether individuals identified attacks. One participant was disqualified due to detecting an experimental setup issue unrelated to the phone-based authenticators.

	Malicious Site		Timing Attack		
	SMS	App, OS Context	App, No Context	App, Distance	App, Screenshot
Number Participants	3	3	3	2	2
Disqualified	1	0	0	0	0
Attack Failed	0	0	0	0	0
Attack Succeeded	2	3	3	2	2
Symptoms Noted	1	1	0	0	0
No Symptoms Noted	1	2	3	2	2

In the second column of Table 2, we show the results of SMS-based code delivery experiments. One participant was disqualified before proceeding to the PBA test. Another did not detect the attack, but described symptoms of the attack during the post-experiment interview. The third participant did not notice the attack or any symptoms of a problem.

Our testing protocol required us to abort the user study for one of our participants before conducting the PBA attack. The disqualified participant noticed discrepancies in the site content of the fake sign-in page before reaching the stage where a password was entered and before the PBA could be tested. The participant indicated they had previously been the victim of an attack and observed an inconsistency in the animation associated with our mimicry of the vendor's site. This participant did not notice the host name's mismatch or the HTTP indicator. We thus were unable to obtain PBA data for that participant.

When we explored the application-based approval approach, none of the three participants detected the attack live, as shown in the third column of Table 2. Only one participant indicated any symptoms; the one identified was related to a mismatch in operating system on the PBA prompt. This response hinted at the value of context; however, a more sophisticated adversary would be able to

observe OS details of the legitimate client and forge browser or OS headers when interacting with the identity provider, causing the results to match.

4.4 Participant Responses to Timing Attack Scenario

In this attack, the host directed the participant to log in to the game website, which started an authentication session via the vendor's authentication page. At the same time, the adversary researcher initiated a second authentication session on a separate system, supplying the same credentials. However, the adversary researcher did not submit the password credentials immediately. Instead, the adversary researcher monitored the attack script running on the router VM. The adversary researcher activated the attack script while the participant was logging in[2]. When the attack script from Section 3.3 observed the trigger condition, it automatically paused all packets associated with the participant's login session. Then, the adversary researcher started the login attempt for the second session. The submission of the second session's information resulted in a PBA request via the host's phone.

The host researcher then showed their phone to the participant with the PBA prompt and asked the participant what buttons should be pressed. If the victim told the host to press a button that allowed the request, the adversary's attempt was approved and the adversary could observe the success. The adversary then instructed the attack script to unpause the participant's connection and deliver all the queued packets associated with the participant's log in attempt. This resulted in a second PBA request at the host's phone, which the host then displayed to the participant and asked for instruction.

In exploring the packet pausing attack, we considered two variants: the standard authentication prompt and one enhanced with additional context. As with the lab-based study, both variants considered only the application-based "approve" request workflow in which the participant was asked to confirm whether they initiated the request or not.

In the standard authentication scenario, the user was provided with a prompt that indicated the account being signed in, the device operating system and architecture performing the log in, a rough location (country-level granularity), an indication that the log in was being performed "now," a button indicating this was correct, and one indicating it was incorrect.

As noted earlier, the only symptoms of the timing attack are that the web page where the user submits a credential is briefly delayed and the end user receives multiple requests to authenticate. As we see in the fourth column of Table 2, none of the three participants detected the attack or reported suspicious symptoms during the interview process. In fact, one of the participants indicated that receiving a second phone-based notification request "seemed pretty standard for [AnonSSO]." That participant proceeded through both verification prompts

[2] Future engineering efforts may allow the script to run continuously and to automatically identify the login session. Since our goal was to measure participant reactions, for simplicity, we manually activated it in the user study.

quickly. Another participant took longer to consider both PBA prompts, but proceeded in each case.

We next explored user behaviors when they have additional context that might alert them to something awry. In this case, the host showed the participant a false notification screen during the first authentication request with two discrepancies from a real notification: it showed that the request originated from a location that was thousands of miles away and was from an operating system that mismatched what the participant was using. As shown in the fifth column of Table 2, neither of the two participants in this scenario detected the attack or reported suspicious behavior in the interview.

In our final scenario, participants were shown a screenshot of a computer desktop in the phone-based authenticator prompt and asked if the image matched what they were trying to do. The researchers intentionally ensured that the screenshot did not match the participant's screen: the screenshot showed a different browser, a different OS, and different screen size. Further, the contents of the window did not match: one displayed a username entry page for a login to a different website whereas the study participant was viewing the password entry page for a login to the game website. Both participants chose to proceeded (as shown Table 2, column 6), despite examining the prompt for over a minute. During the post-experiment interview, both participants indicated the picture was difficult to view through the video conferencing software, so they could not clearly see the details or differences. One of the participants indicated the screenshot looked like their PC's desktop, which may have been a false assurance.

This exploration confirmed our hypothesis that the timing attack was too subtle to seem suspicious to end users and that the duplicate authentication prompt would not raise concerns for them. The user study partially refuted our hypothesis that additional context would help. The details about location and machine type provided little value for user verification. While screenshots may have been useful, the experimental setup appeared to affect the results and further study may be needed. However, the post-experiment interview indicated that even when screenshots do not match, users may still proceed anyway, as long as the image looks familiar. One participant expressed privacy concerns if accurate screenshots of the system were to appear within the PBA prompt.

4.5 Participant Feedback and Study Limitations

In our post-experiment interviews, most participants indicated that having PBAs as part of a MFA scheme increased their confidence in the security of their accounts. To them, the approach was worth the inconvenience. However, two participants indicated that it was not worthwhile.

The presence of PBAs increased some participants' confidence that they were interacting with an authentic website. One user believed that an email with a nonce serves as "proof" of security. This reaction indicates that users may be particularly vulnerable to social engineering attacks that incorporate PBAs.

User studies have inherent limitations in terms of realism, representative populations, and scale. Our use of researcher-provided credentials and a researcher

observing the login may have affected realism, possibly by providing inherent assurance and by heightening user attention to the process. The video conferencing tool affected the screenshot study, but based on participant feedback and actions, it did not affect the other results. Finally, our participant pool was small, with 13 Computer Science majors, which is subject to bias. However, that bias should have increased the attack detection rate and none of our participants reported PBA attacks. This highlights real risks with PBAs in practice.

4.6 Potential Mitigations for Deployment

We recommend that deployers of PBA systems eliminate the usage of the simple application-based prompt to approve or deny a request. Instead, organizations would be more resilient against timing attacks by using the "code matching" requirement for applications since that requires the user to link the action being authorized on the phone with the device being authorized. This can entirely defeat the Packet Delay Attack.

Providing additional context about the relying party or service being authorized may allow end-users to identify mismatches. Such context was examined by our participants, but despite the presence of mismatches, they did not abort the authentication process. We recommend end-user training about how PBAs work and the symptoms of an attack.

Additional training about typo-squatting and website mimicry would also be useful to help users avoid malicious site impersonation. Because the content of a website is under the adversary's control, attackers can create convincing replicas of legitimate sites. Training users to understand URLs may help manage this risk.

Finally, we recommend that organizations consider human interaction with PBAs and minimize the number of times users must employ them. A misconfigured environment may unnecessarily prompt individuals to use phone-based authenticators, which may desensitize users and cause them not to carefully vet authentication prompts. With the sparing usage of such authentication prompts, users may review them more carefully.

5 Concluding Remarks

We explored the use of phone-based authentication systems, which are in widespread use on the Internet. Despite assurances to the contrary, we showed that these systems offer little resistance to phishing attacks. One common phone-based authenticator mechanism can also be defeated by strategic timing attacks. We explored the attack scenarios and showed that they were unnoticed by technically-inclined participants in a user study.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1651540.

References

1. Akhawe, D., Felt, A.P.: Alice in warningland: A large-scale field study of browser security warning effectiveness. In: USENIX Security Symposium. pp. 257–272 (Aug 2013)
2. Amran, A., Zaaba, Z.F., Mahinderjit Singh, M.K.: Habituation effects in computer security warning. Information Security Journal: A Global Perspective **27**(4), 192–204 (2018)
3. Anderson, B.B., Kirwan, C.B., Jenkins, J.L., Eargle, D., Howard, S., Vance, A.: How polymorphic warnings reduce habituation in the brain: Insights from an FMRI study. In: ACM Conference on Human Factors in Computing Systems. pp. 2883–2892 (2015). https://doi.org/10.1145/2702123.2702322
4. Avatier: Azure active directory seamless single sign-on. `https://docs.microsoft.com/en-us/azure/active-directory/hybrid/how-to-connect-sso` (2020), accessed April 29th, 2021
5. Avatier: Which companies use multi-factor authentication with their customers? `https://www.avatier.com/blog/companies-use-multi-factor-authentication-customers/` (2021), accessed April 29th, 2021
6. Bravo-Lillo, C., Cranor, L.F., Downs, J., Komanduri, S., Sleeper, M.: Improving computer security dialogs. In: IFIP Conference on Human-Computer Interaction. pp. 18–35. Springer, USA (2011)
7. Breński, K.P.: Evil Hotspot–are public hotspots safe? Ph.D. thesis, Zakład Strukturalnych Metod Przetwarzania Wiedzy (2017)
8. Brustoloni, J.C., Villamarín-Salomón, R.: Improving security decisions with polymorphic and audited dialogs. In: ACM Symposium on Usable Privacy and Security. pp. 76–85 (2007). https://doi.org/10.1145/1280680.1280691
9. Cristofaro, E.D., Du, H., Freudiger, J., Norcie, G.: Two-factor or not two-factor? A comparative usability study of two-factor authentication. CoRR **abs/1309.5344** (2013), `http://arxiv.org/abs/1309.5344`
10. Das, S., Dingman, A., Camp, L.J.: Why Johnny doesn't use two factor: A two-phase usability study of the FIDO U2F security key. In: Financial Cryptography and Data Security. pp. 160–179 (2018)
11. Das, S., Wang, B., Tingle, Z., Camp, L.J.: Evaluating user perception of multi-factor authentication: A systematic review. CoRR **abs/1908.05901** (2019), `http://arxiv.org/abs/1908.05901`
12. Dasgupta, D., Roy, A., Nag, A.: Multi-factor authentication. In: Advances in User Authentication, pp. 185–233. Springer, USA (2017)
13. Dhamija, R., Tygar, J.D., Hearst, M.: Why phishing works. In: ACM SIGCHI Conference on Human Factors in Computing Systems. pp. 581–590 (2006). https://doi.org/10.1145/1124772.1124861
14. Doerfler, P., Thomas, K., Marincenko, M., Ranieri, J., Jiang, Y., Moscicki, A., McCoy, D.: Evaluating login challenges as a defense against account takeover. In: The ACM World Wide Web Conference. p. 372–382 (2019). https://doi.org/10.1145/3308558.3313481
15. Downs, J.S., Holbrook, M.B., Cranor, L.F.: Decision strategies and susceptibility to phishing. In: ACM Symposium on Usable Privacy and Security. pp. 79–90 (2006). https://doi.org/10.1145/1143120.1143131
16. European Commission: Payment services (psd 2) - directive (eu) 2015/2366. `https://ec.europa.eu/info/law/payment-services-psd-2-directive-eu-2015-2366_en` (2015), accessed June 6th, 2022

17. Felt, A.P., Ainslie, A., Reeder, R.W., Consolvo, S., Thyagaraja, S., Bettes, A., Harris, H., Grimes, J.: Improving SSL warnings: Comprehension and adherence. In: ACM Conference on Human Factors in Computing Systems. pp. 2893–2902 (2015). https://doi.org/10.1145/2702123.2702442
18. Government of Singapore: Police advisory on scam survey leading to the misuse of singpass access to digital services. `https://ec.europa.eu/info/law/payment-services-psd-2-directive-eu-2015-2366_en` (2022), accessed June 6th, 2022
19. Hong, J.: The state of phishing attacks. Communications of the ACM **55**(1), 74–81 (Jan 2012). https://doi.org/10.1145/2063176.2063197
20. Jover, R.P.: Security analysis of sms as a second factor of authentication: The challenges of multifactor authentication based on sms, including cellular security deficiencies, ss7 exploits, and sim swapping. Queue **18**(4), 37–60 (2020)
21. Kerkhoff Technologies, Inc.: Netfilterqueue. `https://github.com/kti/python-netfilterqueue` (2021), accessed April 29th, 2021
22. Kondracki, B., Azad, B.A., Starov, O., Nikiforakis, N.: Catching transparent phish: Analyzing and detecting MITM phishing toolkits. In: ACM Conference on Computer and Communications Security. p. 36–50 (2021). https://doi.org/10.1145/3460120.3484765
23. Konoth, R.K., van der Veen, V., Bos, H.: How anywhere computing just killed your phone-based two-factor authentication. In: International Conference on Financial Cryptography and Data Security. pp. 405–421. Springer (2016)
24. Lee, K., Kaiser, B., Mayer, J., Narayanan, A.: An empirical study of wireless carrier authentication for SIM swaps. In: Symposium on Usable Privacy and Security. pp. 61–79 (2020)
25. Microsoft: Microsoft digital defense report. `https://www.microsoft.com/en-us/security/business/security-intelligence-report` (2020), accessed April 29th, 2021
26. Microsoft: Optimize reauthentication prompts and understand session lifetime for Azure AD multi-factor authentication. `https://docs.microsoft.com/en-us/azure/active-directory/authentication/concepts-azure-multi-factor-authentication-prompts-session-lifetime` (2020), accessed April 29th, 2021
27. Niemietz, M., Schwenk, J.: Owning your home network: Router security revisited. CoRR **abs/1506.04112** (2015), `http://arxiv.org/abs/1506.04112`
28. ReportLinker: Global multi-factor authentication (mfa) industry. `https://www.reportlinker.com/p03329771/Global-Multi-Factor-Authentication-MFA-Industry.html` (2021), accessed April 29th, 2021
29. SecDev: Scapy. `https://github.com/secdev` (2021), accessed April 29th, 2021
30. Selenium: Seleniumhq browser automation. `https://www.selenium.dev/` (2021), accessed April 29th, 2021
31. Sinigaglia, F., Carbone, R., Costa, G., Zannone, N.: A survey on multi-factor authentication for online banking in the wild. Computers & Security **95**, 101745 (2020)
32. Spaulding, J., Nyang, D., Mohaisen, A.: Understanding the effectiveness of typosquatting techniques. In: ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies (2017), `https://doi.org/10.1145/3132465.3132467`
33. Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., Cranor, L.F.: Crying wolf: An empirical study of SSL warning effectiveness. In: USENIX Security Symposium. pp. 399–416 (2009)
34. Zaaba, Z.F., Boon, T.K.: Examination on usability issues of security warning dialogs. Age **18**(25), 26–35 (2015)