Proposed Defenses Against Active Concurrent Session Attacks on User Terminal

Rituparna Mandal

Abstract—**This paper provides an in-depth examination of Multi-Factor Authentication (MFA) systems within the context of Concurrent Session Injection (CSI) attacks and proposes defense strategies against such threats. It explores MFA techniques involving user interaction and assesses System Usability Scale (SUS) scores associated with various MFA methodologies. Specifically, the paper focuses on understanding CSI attacks initiated by malware at the client side. Additionally, it details the development and implementation of defense mechanisms aimed at mitigating CSI attacks within MFA systems, emphasizing the need for proactive measures to counter such attacks.**

I. Introduction

In the ever-evolving landscape of cybersecurity, safeguarding sensitive information and digital identities remains an ongoing challenge. As organizations and individuals increasingly adopt Multi-Factor Authentication (MFA) systems to fortify their security measures, cyber adversaries persistently innovate sophisticated strategies to bypass these defenses. This paper explores novel approaches to counter such advanced attacks aimed at compromising MFA systems, focusing on key defensive strategies and their efficacy in thwarting potential threats.

This paper aims to dissect and analyze one particular catagory of attack targeting MFA systems - the active CSI attack orginating from user terminal. [3] It involves a comprehensive understanding of the intricate phases involved in such attacks, including keylogging, login redirection, fabrication of counterfeit MFA prompts, and unauthorized confirmation of MFA challenges. By deconstructing these phases, the paper delves into proposed defensive strategies aimed at impeding each stage of the attack, thereby fortifying the overall resilience of MFA systems.

One of the primary challenges addressed within this paper pertains to the difficulty in differentiating between legitimate user activities and malicious actions at the server end. Consequently, the focus shifts towards implementing defenses within the user's terminal or browser environment, where distinguishing between legitimate and malicious actions is relatively more feasible.

Moreover, the paper sheds light on various strategies to thwart such attacks and the outcomes of testing methodologies employed to validate the efficacy of these defensive measures. Testing involved interactions with different CAPTCHA types, revealing variations in their ability to resist automated script manipulations, thus highlighting the robustness of image-based CAPTCHAs against automation.

II. Literature Review

Some MFA schemes provide protection against remote concurrent attack where the attacker replays the login request and spoofs the user into accepting the MFA request in their device. however these schemes fail against concurrent attacks launched by malware in the user terminal.
The following section briefly introduces two such schemes:
1. **2D-2FA (Two-Dimensional Two-Factor Authentication)** [2] involves users actively participating in the authentication process. Instead of a simple "approve" or "reject" action seen in Push-2FA, 2D-2FA requires users to copy an identifier from the login terminal to their personal device. Once this identifier is entered, a PIN is computed and directly sent to the server to the MFA Device for authentication. This method contrasts with Push-2FA, where users have to make a decision (approve or reject) for authentication requests. In Push-2FA, there's a risk that users might unintentionally approve a concurrent session initiated by an attacker.
2.The **REPLICATE** [1] method proposes improvements for push notification-based Two-Factor Authentication (TFA) by addressing vulnerabilities in this system. It introduces a closed-loop feedback-based approach to enhance security and usability. The core concept is based on users' difficulty in distinguishing between genuine and compromised push notifications due to consistent user interfaces and repetitive login steps. To mitigate this issue, REPLICATE suggests design changes where users must perform actions on their token devices that match the prompts displayed in the web browser or app login interface. These actions are randomly generated for each login attempt. This variability aims to help users differentiate between legitimate and potentially fraudulent login attempts, improving overall security in push-based TFA.
It is interesting to note that introducing complex interaction between user and device can not prevent the active CSI attacks as long the malware on terminal can forward the interaction needed for their request to the user, deceiving users into resolving challenges on their devices. Subsequently, this manipulation results in the inadvertent approval of the attacker's request.

User adoption of MFA Practices: Another important aspect of designing MFA schemes is to ensure that schemes remain user-friendly without introducing additional cognitive difficulty to solve the challenges. The research conducted by Reese et al. [4] compared the usability of five common 2FA (Two-Factor Authentication) methods: SMS, TOTP (Time-based One-Time Password), pregenerated codes, push notifi-

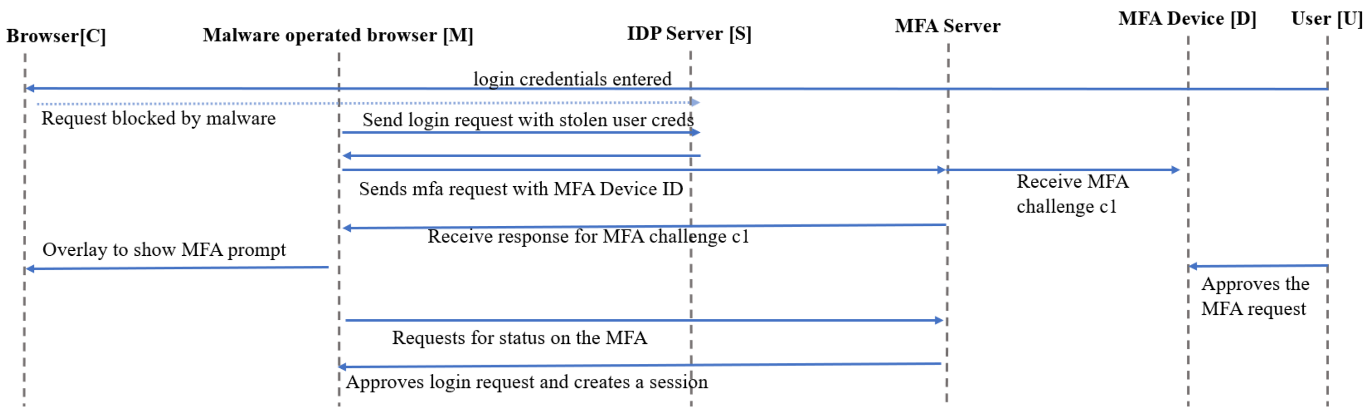
Browser[C]
Malware operated browser [M]
IDP Server [S]
MFA Server
MFA Device [D]
User [U]
login credentials entered
Request blocked by malware
Send login request with stolen user creds
Sends mfa request with MFA Device ID
Receive MFA challenge c1
Receive response for MFA challenge c1
Overlay to show MFA prompt
Approves the MFA request
Requests for status on the MFA
Approves login request and creates a session

Fig. 1: CSI Attack framework

cations, and U2F (Universal 2nd Factor) security keys. The study uniquely evaluated all these methods within a single simulated web application to minimize potential confounding factors and measure authentication times.

The findings of the study revealed that passwords without a second factor received the highest System Usability Scale (SUS) score, indicating the highest perceived usability among the evaluated authentication methods. Following this, TOTP using Google Authenticator scored second-highest, showing relatively good usability but lower than passwords without a second factor.

This research underscores the importance of considering usability alongside security when implementing 2FA solutions. While TOTP using Google Authenticator demonstrated reasonable usability, passwords without a second factor were perceived as the most usable among the evaluated authentication methods. It is apparent that an overly complex or intricate design geared solely toward strengthening security measures may inadvertently deter user adoption.

III. CSI ATTACK FRAMEWORK

A generic user interaction during login with MFA occurs as follows:
1. The user enters login credentials on the client side, C.
2. The client interacts with the server S (MFA/IDP server), sending an authentication challenge to the user's device, D. The server displays an authentication prompt to the client, C.
3. The user responds to the MFA prompt on the device, D, which may include approving the push notification and typing the OTP from the client-side prompt.
4. The web browser sends a status check request to the server to inquire about the MFA status. The server responds with 'expired request,' 'approved,' or 'denied' based on the device's response.

In this interaction, the attack [] described in Fig 1 unfolds in the following manner:
1. Upon typing the login credentials for a request (say R1), a keylogger steals the credentials.
2. Malware launches a headless browser, capturing the credentials and initiating a second request (R2) using the stolen login credentials. The server responds with the MFA challenge for request R2.
3. The malware redirects the user's web browser (C), blocking request R1 and redirecting it to a fake MFA prompt page created by the malware, utilizing the resources from the user.
4. The user, believing the MFA request received on their device is for R1, completes the interaction with the device (D) based on the fake MFA prompt seen on client C.
5. Upon completion, the server mistakenly approves the login request for the attacker.

IV. POSSIBLE DEFENCES

The attack mimics the scenario where the user appears to cancel their initial login request and creates another login attempt using a different browser. Because it's hard to tell apart the user's genuine activity (C) from the malicious one (M) at the server's end, it's crucial to fix this issue within the browser or terminal.

Since it's challenging to distinguish between these actions at the server's end, the solutions should be implemented within the browser or terminal environment. This approach, such as using browser extensions or terminal-based security measures, helps in identifying and stopping these attacks directly where the user interacts with the system.

To start devising solutions for countering the attack, it's imperative to comprehend the key stages involved in the attack sequence. This attack can be categorized into several distinct phases:
1. **Keylogging**: Involves the capturing of user login credentials through a keylogger. For passwordless login designs, this step is not required.
2. **Login to MFA redirection** : Represents the process where the attacker redirects the login attempt to a different URL where they plan to fake MFA prompt.
3. **Launching malware operated browser**:Represents the process where the malware launches a browser in the

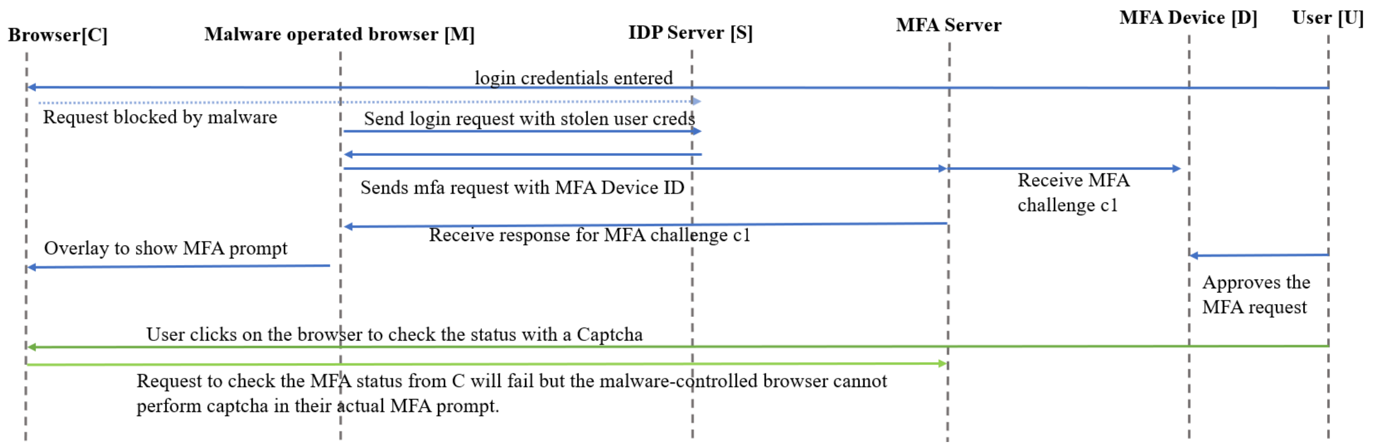
Browser[C]
Malware operated browser [M]
IDP Server [S]
MFA Server
MFA Device [D]
User [U]
login credentials entered
Request blocked by malware
Send login request with stolen user creds
Sends mfa request with MFA Device ID
Receive MFA challenge c1
Receive response for MFA challenge c1
Overlay to show MFA prompt
Approves the MFA request
User clicks on the browser to check the status with a Captcha
Request to check the MFA status from C will fail but the malware-controlled browser cannot perform captcha in their actual MFA prompt.

Fig. 2: Captcha based MFA scheme

background and sends a login request with stolen credentials.
4. **Construction of a counterfeit MFA prompt and its display on client C**: This phase involves creating a fake Multi-Factor Authentication (MFA) prompt with any identifier that might be required for user interaction with device and presenting it to the user on the client side, deceiving them into interacting with the counterfeit prompt.
5. **Confirmation of the MFA Challenge on client D**: The user, believing the fake MFA prompt is legitimate, completes the Multi-Factor Authentication process on their device (client D), unaware that it's for the attacker's request.

To defend against this attack, proposed strategies include:
1. Defense against Login Redirection : One proposed solution involves a browser extension that monitors suspicious redirections. The extension can block local URLs where fake MFA prompts are hosted, preventing their display to users. Alternatively, the extension, designed by a trusted party, can verify the URL sequence during login attempts, ensuring it aligns with expected pattern.
2. Defense against Launching Malware-operated browser: The malware script does not require any administrative access to launch the hidden browser in the background.

3. Defense against Confirming MFA Challenge: To counteract automation scripts replicating interactions with fake MFA prompts, a simple CAPTCHA is suggested for the client's MFA prompt. The introduction of a CAPTCHA after user approval on device D aims to hinder malware-operated browsers.
To counteract the automation script's ability to replicate user interactions from a fake MFA prompt to the genuine one, a simple CAPTCHA can be introduced on the MFA prompt displayed in client C after the user approves the login request on device D. This addition aims to hinder the automation script in the malware-operated browser (M) from successfully passing the CAPTCHA to confirm the request.

Testing Malware script response with Captcha:
We conducted testing using Selenium scripts to interact with CAPTCHAs from two different companies, Google reCAPTCHA [5], and Friendly CAPTCHA [6]. Upon attempting to click the "I'm not a robot" verification button for both CAPTCHAs, the outcomes varied:
For Friendly CAPTCHA, the script failed to complete the verification.
Google reCAPTCHA prompted a different image selection challenge for verification.
CAPTCHAs are designed to assess mouse movements, browser headers, and prior interactions or cookies within the browser. While more advanced malware scripts might attempt to replicate human-like mouse movements, solving image selection CAPTCHAs typically demands sophisticated machine learning-based automation tools.

V. Future Work

There is significant scope for development and exploration for such protocols to avoid such sophisticated attacks as the CSI attack. In this work, we will focus on two primary objectives. Firstly, we should develop and test proof-of-concept solutions that integrate proposed fixes aimed at thwarting similar attacks. These fixes involve implementing CAPTCHAs in the MFA process and deploying browser extensions to detect and prevent suspicious redirects. We'll evaluate their performance against the malware we've designed. Secondly, our focus will be on evaluating user adoption and the usability of these POCs. This assessment includes analyzing authentication times, setup processes, and gathering qualitative opinions from users about the solutions. We should involve users in studies, surveys, and usability tests to understand their experiences and viewpoints regarding these advanced security features. The insights gained from user feedback will guide us in refining the design, ensuring that the security measures not only offer robust protection but also maintain user-friendliness. Our goal is to enhance these defenses based on real-world feedback such that they can seamlessly integrate into users' existing workflows while providing effective protection.

REFERENCES

[1] J. Prakash et al., ”Countering Concurrent Login Attacks in “Just Tap” Push-based Authentication: A Redesign and Usability Evaluations,” 2021 IEEE European Symposium on Security and Privacy (EuroSP), Vienna, Austria, 2021, pp. 21-36, doi: 10.1109/EuroSP51992.2021.00013.

[2] Maliheh Shirvanian and Shashank Agrawal. 2021. 2D-2FA: A New Dimension in Two-Factor Authentication. In Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC ’21). Association for Computing Machinery, New York, NY, USA, 482–496. https://doi.org/10.1145/3485832.3485910

[3] Ahmed Tanvir Mahdad and Nitesh Saxena. 2023. SoK: A Comprehensive Evaluation of 2FA-based Schemes in the Face of Active Concurrent Attacks from User Terminal. In Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec ’23). Association for Computing Machinery, New York, NY, USA, 175–186. https://doi.org/10.1145/3558482.3590183

[4] Reese, Ken, et al. ”A Usability Study of Five Two-Factor Authentication Methods.” Proceedings of the Symposium on Usable Privacy and Security (SOUPS), USENIX Association, 2019

[5] ”Google reCAPTCHA.” Google, n.d.www.google.com/recaptcha/about/.

[6] ”Friendly Captcha.” Friendly Captcha, n.d., friendlycaptcha.com/.