Remote Access Banking Fraud Defense

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Anthony Maringo

Spring, 2022

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Rosanne Vrugtman, Department of Computer Science

Daniel Graham, Department of Computer Science

Remote Access Banking Fraud Defense

ABSTRACT

Banking fraud impacts people's lives and general well-being, and customers need a defense against fraudsters remotely accessing their accounts by posing as tech support. To address this problem, I designed a feature that checks whether the banking customer is currently screen-sharing or has screen-shared recently. The system takes notes once the customer begins screen-sharing, temporarily restricting risky transactions and monitoring the account for unusual behavior.

It is currently not possible to track the customer's phone behavior outside of the banking application to take note of recent screen-sharing. Additionally, the user's usage of screen-sharing and remote access applications can easily be tracked while in the banking application, although there is no way to know whether the customer is simply screen-sharing or if they are also providing remote access to an outside entity. Tracking the customer's screen-sharing usage while in the application provides an added layer of security to prevent this type of banking fraud, and it can also be applied to any application that deals with sensitive information.

1 Introduction

There are many different forms that fraud can take, and each has an impact on people's lives and general well-being. In 2020 it was reported that consumers lost more than $3.3 billion to fraud related incidents [1]. In the banking industry specifically, customers are encountering fraudsters posing as tech support agents and remotely accessing the customer's accounts through screen-sharing technology. Remote fraud is committed through phishing attacks or intercepting legitimate tech support service calls from the client.

2 Background

In the remote fraud scheme, customers are typically contacted saying that there is an issue with their account that they need to resolve. The fraudster then connects with the customer and offers to help them through the process if they share their screen. Upon the customer sharing their screen with the fraudster, sensitive information can be revealed. For example, the fraudster could watch the customer type their password in. Although the fraudster may not be able to see which keys exactly were pressed, they could determine the length of the password and if any numbers or symbols were used. Additionally, numerous remote access applications exist to aid tech support in legitimate service calls. When using these applications, consumers grant permission to the tech support, allowing full control over their device. Once the fraudster has control over the device, they are then able to conduct unauthorized transactions.

3 Related Works

When trying to find related work on remote fraud detection specifically through screen-sharing, I was unable to find any discussion about the topic. Using search keywords such as "remote fraud detection iOS" and "screen sharing attack" resulted in various fraud defense systems with widely differing approaches to vastly different types of fraud.

Many of the papers that were found used machine learning models to try and detect fraud. Additionally, it was very common for the defense mechanisms to process large amounts of data to try and draw conclusions about fraudulent acts. I was not able to find articles that addressed creating mobile device defense mechanisms against a phishing attacks of this nature. The lack of literature on this topic may be a result of more prominent threats that are impacting users at a larger scale. Additionally, this seems to be a newer form of attack with few notable occurrences. The largest and most recent use of this attack occurred on the Robinhood trading platform. Approximately 5 million user's email addresses and 2 million user's full names were exposed in this attack [3]. As very limited personal information was revealed and no users were financially harmed, this attack is not seen as an extremely high priority threat.

4 Project Design

There are three major steps in the design of this project, detecting screen-sharing activity, restricting account behavior, and authenticating the user. This method of remote fraud detection is designed specifically for iOS applications running on iOS 11.0 or higher.

4.1 Detecting Screen-Sharing

There are two cases in which detecting screen-sharing is useful. First, detecting if the screen-sharing started before the target application is opened. Second, detecting if screen-sharing started after the user was already in the application.

Detecting screen-sharing before the application is launched is important because the targeted application could instantly restrict behavior and produce warnings to the user about the dangers of screen-sharing their sensitive information. Additionally, it would be important to note if the fraudster is using a tech support application that provides full access to the user's device since this would have to be set up before accessing the targeted application.

Once the application is up and running, the user has already gone through numerous authentication checks to ensure that they should be allowed access to the information in the application. If the user begins screensharing after the authentication checks have been conducted, a fraudster may be able to gain insight to important sensitive information regarding that particular user's account.

In either case of screen-sharing pre or post launching the application, the method of checking to see if the user is sharing their screen is the same. Apple created a flag that is automatically signaled when the device is "actively cloning the screen to another destination" [2]. This flag includes the cases of recording, mirroring, or using Airplay to reproduce the contents of the screen to another device. This flag is the first component of reducing remote access fraud.

4.2 Restricting Account Behavior

Once screen-recording has been detected, the application needs to react in order to try and prevent fraudulent activities. Additionally, the system should alert the user of the risks involved with sharing the sensitive information from their account.

To minimize the damage that can be done during screen-sharing or remote access sessions, the system temporarily restricts high risk actions. Examples of high risk actions include transferring money or changing core account information. The user's account will not be fully restricted by this feature as it is only one countermeasure that aims to provide another layer of security to the application. One goal of restricting these actions from the user is that it will raise awareness about the dangers of screen-sharing or providing remote access to anyone. The restrictions will be applied to the account until the next time the user goes through a strong authentication challenge.

4.3 Authenticating Users

A core component of application security, especially in the case of a banking application, is verifying the person accessing the account is the person who should be allowed to access the account. Without proper verification, fraudsters could easily conduct unauthorized transactions on accounts with minimal friction.

To authenticate users, there are various levels of challenges the user must go through to get to certain levels of access.

The level of challenge is representative of how sure the application is that the person going through the challenge is who they claim to be. A lower level challenge would be logging into the account with a username and password to a new device. This type of challenge would warrant the lowest level of trust because it is from an unfamiliar device only using the user's credentials. There are many ways that a user's credentials could be stolen and used by a fraudster.

An example of a higher level challenge is doing a multi-factor authentication. This type of challenge would require the user to have the correct credentials of the account to first login, then they would also need to be able to access the phone number or email associated with the account for a one time passcode. Additional multi-factor authentication applications could be used for this purpose rather than relying on a phone number or email. Once a high level challenge is completed, the user will have access to all high risk activities on the account, such as transferring money. After the user completes a high level challenge, they will not be asked to complete another high level challenge again for a few weeks, unless there are questions about fraudulent activity on the account.

5 Results

When creating iOS applications, there is a limited scope in which the application can gather its information. The operating system does not allow for individual applications to know about the user's actions outside of the scope of a given application by default. To gain access to user behavior outside of a particular application, an application must request permission from the user. This creates privacy concerns for the customer because they often do not understand how these permissions will be used. Also, in a large scale corporate application it is required to have these permissions reviewed by multiple teams and multiple levels of the chain of command before it can go into production. This is a rigorous process because of the negative impact that requesting additional information from the client causes. As a result of this process for gaining information on the user outside of the scope of the application, it is currently not possible to perform a pre-check of the user's screen-sharing prior to the launch of the particular banking application I was working on.

Once the user has launched the banking application, the check to see if the device is currently screen-sharing or allowing remote access to an outside device can be conducted. Upon researching how iOS stores the flags for screen-sharing and remote access, I found that both screen-sharing and remote access are stored under the same flag. With this finding, anytime a flag is found, the application must assume that the user is both screen-sharing and providing remote access to another device. This would be the best practice to ensure account security until this implementation detail is changed in future operating systems.

6 Conclusion

A remote fraud defense mechanism to protect consumers from fraudulent tech support calls is an important feature for applications that contain sensitive information. Screen-

sharing and remote access fraud are significant issues in banking, but this defense mechanism can provide added security to many applications in other industries. By monitoring the screen-sharing flag in iOS, user accounts can become temporarily restricted until their identity is further authenticated to prevent high risk actions from occurring. This defense mechanism has the potential to save consumers thousands of dollars, and many hours of headaches recovering from banking fraud.

7 Future Work

At the conclusion of my internship this project was in the prototyping phase of production. This project was of great interest to upper management in the company and will continue to move forward and become integrated within the banking application to thwart remote fraud attempts. The authentication and detecting screen-sharing aspects of this project were able to be completed in the time span of the internship, which leaves the actual restricting of the account from conducting risky transactions to be implemented.

Additional considerations that will need to be evaluated is how iOS 15 handles screen-sharing flags and screen-sharing over facetime. During the design and implementation of this project iOS 15 had not been released. In iOS 15 screen-sharing capabilities are expanded, so a re-evaluation of detecting screen-sharing should be done before this project goes into production.

8 UVA Program Evaluation

To complete this project, I had to learn the programming language Swift and several other internal software development tools. By taking Introduction to Computer Science, Computer Organization and Architecture, and Data Structures and Algorithms, I was confident in my ability to learn a new language. In these classes we learned the fundamentals of coding and were shown how they translate between various languages to solve problems. Additionally, Advanced Software Development Techniques was a significant help because it provided me with the foundational knowledge to manage a GitHub repository.

Although these courses provided me with a good foundation for the core computer science principles, soft skills like presenting your work to people who are not as deeply involved with the technology were learned through the internship. Another soft skill that I developed over the course of my internship was how to effectively communicate with a team. In my coursework at UVA, there have been very few group projects that spanned a long duration and required members to work independently on smaller parts of the project to create something greater. Typically, most of the projects in my coursework were best completed in one or two group meetings with everyone focusing on a single task.

REFERENCES

[1] FTC. 2021. New Data Shows FTC Received 2.2 Million Fraud Reports from Consumers in 2020, 4 (Feb, 2021). DOI: https://www.ftc.gov/news-events/press-releases/2021/02/new-data-shows-ftc-received-2-2-million-fraud-reports-consumers

[2] Apple. isCaptured. DOI: https://developer.apple.com/documentation/uikit/uiscreen/2921651-iscaptured

[3] Robinhood. 2021. Robinhood Announces Data Security Incident, 8 (November, 2021). DOI: https://blog.robinhood.com/news/2021/11/8/data-security-incident