Expiring Barcodes Design Project

December 11, 2016

Red Kernel Garsuta (72186091, kyle.garsuta@gmail.com)
Jae Yeong Bae (44316107, mtbnunu@gmail.com)
Matthew Mackenzie (22537147, mmacken42@gmail.com)
Anni Wang (17574138, annikwang.ca@gmail.com)

Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada

Abstract* - A study in the security of the Starbucks Mobile App payment method is done, and practical improvements are made. The central concept of the study is the ease of gaining a replica of the Starbucks App user's unique barcode using social engineering techniques, and the opportunity of using replay attacks to enable unauthorized purchases made on that user's account. Our solution uses a Time-Based One-Time Password (TOTP) in conjunction with the user's unique payment number to generate expiring, one-time barcodes that provides stronger authentication for purchases while maintaining efficiency and psychological acceptance. We demonstrate the prototype design using a two-part system emulating a Starbucks server and the client using the mobile app. This system is applicable to similar mobile payment systems that relies on a static barcode, and promotes the use of two-factor authentications while raising awareness of vulnerabilities in such systems.

I. Introduction

The Starbucks mobile application, available for Android and iOS devices, is a common form of payment at most Starbucks locations in North America and across the globe. Starbucks reported in 2015 that more than 21% of customers now pay using the mobile app [1]. Upon signing up, the app assigns each user a unique 16-digit number, from which a unique barcode is generated in the PDF-417 format [2]. This barcode represents the user's digital card on the app, which can then be used to make payments at Starbucks locations using a barcode scanner.

Unfortunately, this payment system is remarkably insecure. Using the same barcode for every transaction implies that a simple screenshot of a user's payment page is all that is required to make unauthorized purchases on the user's Starbucks account. We demonstrate using social engineering techniques that it is relatively trivial to obtain a copy of a user's barcode or payment number. Many mobile payment systems other than Starbucks' also rely on static barcodes, such as the QR Code payment system in Best Buy and other retail stores [18]. The design we propose here is certainly applicable to those systems as well. We chose to focus our study on the Starbucks Mobile App due to its current widespread usage and growing relevancy.

Two-dimensional (2D) barcode technology and its use in barcode mobile payment offer convenience to our daily lives, but this system is vulnerable in a variety of ways; most notably to replay attacks. However, the security of barcodes may be improved through combination with other cryptographic mechanisms. In fact, security systems utilizing barcodes are surging in popularity among the crypto-community [13][14][15]. Such systems focus on using barcodes to authenticate banking transactions on untrusted terminals by relying on a trusted device (i.e. smartphones). They also make use of smartphone's camera capabilities to scan barcodes in order to securely transmit cryptographic information over a visual channel. While we applaud the ingenuity of such designs, each of these proposed systems is costly in terms of time and resource, and would certainly be unfeasible in the fast-paced environment of buying one's morning coffee. In a real-life scenario, customers and companies alike would be reluctant to adopt a system that slows down the purchasing process and force them to perform additional steps during payment. Part of the popularity of Starbucks' mobile payment system is attributed to how simple and efficient it is for customers to use, so it is important that our design maintains the psychological acceptability of the application.

Our design dramatically improves the security of the barcode payment transaction, while adhering to the constraints of efficiency and ease of use that users would expect from the system. To overcome the vulnerabilities inherent in using a static barcode for every purchase, we used Time-based One-time Passwords (TOTP) in conjunction with the user's unique Starbucks payment number in order to automatically generate unique, single-use PDF-417 barcodes using the user's trusted device. Even in the situation where an attacker is able to obtain a user's unique payment number, this number alone cannot be used to generate a valid barcode. Furthermore, any screenshot of the user's payment barcode will become useless within a matter of minutes.

II. Related Work

Starbucks App Vulnerability

The Starbucks Mobile apps vulnerability to replay attacks was publicized in various online articles and forums [5][6][7]. However, there are currently no deployed countermeasures since it does not directly threaten Starbucks' assets. This threats to this vulnerability affects the growing number of users of Starbucks' mobile payment system. The confidentiality, integrity, and availability of their personal Starbucks accounts and funds are at risk because of this vulnerability, and that is what our design aims to rectify.

Design Concept Inspiration

Our design concept for expiring barcodes was inspired by the popular Google Authenticator app [8]. Authenticator is an open source implementation of TOTP available for iOS and Android devices [9]. It provides two-factor authentication for client logins on web apps and devices for major companies like Microsoft, Facebook, and of course, Google. Authenticator specifically relies on the HMAC-based One-time Password Algorithm (HOTP) put forth in RFC 6238 and RFC 4226 [10][11][12]. Users of Google Authenticator app are given a random six to eight digit password on a trusted device to enter in addition to their regular username and password login information. This one-time code is short-lived, and will continually update every thirty seconds. This system also requires an initial setup step, in which a shared secret key is established between the user's Authenticator app and the website or app they are attempting to log into using this method of two-factor authentication.

Similar Systems

Our design offers a streamlined implementation of a back-end two-factor authentication system, in which users do not need to make any changes to their current payment routine. By concatenating a TOTP password with each user's unique Starbucks ID, our app automatically generates a corresponding one-time barcode to be used for payment. While there has been a significant amount of research on using barcodes to secure transactions, we have not found other systems that use our approach. However, there have been proposals for similar related systems.

One similar system is the QR-TAN barcode transaction-authentication technique detailed in ”QR-TAN: Secure Mobile Transaction Authentication” [13]. QR-TANs are 2D barcodes that are used in a challenge-response system to enable customers performing bank transactions to directly authenticate a transaction before it occurs. The QR-TAN system also uses one-time passwords, but these passwords are presented in the form of a low-resolution QR barcode that can be scanned by the camera on a user's trusted device. Scanning the QR-TAN barcode provides the user with a short, temporary password that they may enter into an untrusted terminal (e.g. an ATM) in order to authenticate their proposed bank transaction. The QR-TAN system offers many security benefits for bank customers. However, it also involves extra steps on behalf of the user (first using their camera to scan a barcode and then manually entering the resulting code) that make it impractical for use in busy lines for small commercial transactions such as buying coffee. While we believe the QR-TAN system to be a worthwhile proposal for use in the high-stake transactions such as online banking, the stakes of purchasing coffee are certainly lower. Therefore, our design places a higher priority on the ease of use and speed of the system, while still achieving similar levels of security.

Others have pointed out the potentials barcodes hold for use in cryptosystems. For example, the study ”Using Camera Phones for Human-Verifiable Authentication” proposes using camera phones to scan barcodes as a means of establishing a restricted visual communication channel to exchange cryptographic keys [14]. ”The Untrusted Computer Problem and Camera-Based Authentication” similarly proposes using the cameras on smartphones to ensure the integrity of transactions. In this case, the smartphone acts as a monitoring device that will alert users if the information being displayed on an untrusted terminal is incorrect or has been tampered [15]. Like the QR-TAN system, the main application for these designs is banking transactions. However, even moreso than the QR-TAN system, this camera-based authentication system requires much greater effort and time on the part of the user and their device. The camera requires high-resolution capabilities and entail a complicated calibration phase that is prone to errors [15]. Our system does not require calibration or a functioning camera at all and takes place behind the scenes of the app. This keeps the economy of design and shields users from any additional headaches.

III. Adversary Model

In order to devise effective countermeasures, we developed a model to profile an adversary who may wish to take advantage of the vulnerability the Starbucks mobile app. This model allows us to gain an understanding of the adversary's objective and capabilities, so that we may anticipate attacks and build improved threat models.

Objective

The objective of the adversary is to obtain knowledge of a user's unique payment number or the corresponding barcode. Ideally, their target is to obtain a screenshot of the payment page containing the barcode. However, the unique payment number is easily translated into a barcode using freely available tools online, so either will do as a replica. With the replica in hand, the adversary can perform unauthorized purchases on the user's account. The adversary may be a normal person who wishes to get free coffee, or a professional hacker who can further utilize the user's accounts to suit their own needs.

Initial Capabilities

We can assume that through observations by proximity, the adversary has knowledge of the user's device (e.g. Android or iOS), as well as the user's payment number. We can also assume that the adversary has physical access to the user's device through social engineering practices, which allows them to examine the user's Starbucks account and see the barcode and payment number.

Capabilities During Attack

During the attack, the adversary gains full knowledge of the user's unique 16-digit Starbucks payment number, which can be used to make a replica of the user's barcode. The adversary also has the capability to use the user's physical device take screenshots of the payment page and send it to themselves for later replay attacks.

IV. System Design

Overview

We implemented a modified Starbucks mobile app payment system that adds TOTP for two-factor authentication. TOTP is an extension of the HOTP algorithm that uses a secret shared key and the current timestamp as inputs to a cryptographic HMAC hash function. This generates a one-time password for the client that is verified on the server side which performs identical TOTP functions. We concatenate the user's 16-digit Starbucks ID, or payment number, with a TOTP in a process that is hidden from the user in order to automatically produce a unique one-time PDF-417 barcode. This barcode is scanned and processed by the Starbucks server to check for the legitimacy of the TOTP value produced by the user and their payment number, then grant legit users authentication. Our proposed system yields expiring barcodes so that even if an attacker obtains a copy of the user's barcode, it quickly becomes useless. Furthermore, if the attacker obtains the user's payment number, they would be unable to generate a valid barcode without simultaneous knowledge of the current value of the TOTP.

Principles of Design

Our system design employs several key principles of secure design, including:

- **Economy of Mechanism.** We followed this principle to ensure that our implementation was as simple as possible.
- **Complete Mediation.** We verify the TOTP value for every purchase, instead of a static ID for all purchases.
- **Open Design.** Our source code is open for evaluation on GitHub. We ensured that security does not rely on the secrecy of our design, but only on the shared secret key.
- **Psychological Acceptability.** We designed the system so that no extra steps are required by the user. From their perspective, the app is exactly the same to use as Starbucks' app.
- **Defense in Depth.** By implementing a TOTP-based barcode system, we have added a layer of defense to the system.
- **Question Assumptions** We question the faulty assumption that the user's unique payment number cannot be stolen.

V. System Prototype

Implementing our design in the real world would require Starbucks to overhaul their payment system and servers, which is beyond the scope of this design project. As a proof-of-concept, we emulated a prototype system to model client-server interactions by implementing (1) an Android mobile application to represent a client using the Starbucks App, and (2) a web application to represent the Starbucks server with a barcode scanner that authenticates users (see Fig. 1 in Appendix for design overview). For this project, we used a smartphone camera on the server side as a substitute for the barcode scanner that is typically found by the Starbucks register.

Web Server

The role of the server-side application is to authenticate legitimate users by scanning their barcodes. To perform authentication, the server retrieves a user's unique ID and the TOTP value from their barcode and validates each value. The server is also responsible for distributing a pre-shared key (PSK) to the user's trusted device upon their registration. For the purpose of this project, we have implemented this through the Diffie-Hellman mutual key-exchange protocol. The PSK is used as a parameter, along with the current timestamp and the barcode's expiration (or update) frequency to generate a TOTP value. The server validates a TOTP by generating its own TOTP value and comparing it to the client's. Since the PSK and timestamp parameters are the same for both server and client, a legitimate client's TOTP should match that of the server's.

Mobile App

The role of the client-side mobile application is to generate expiring barcodes for the server to authenticate. The app allows users to register and log in, and attain a unique 16-digit payment ID as well as a PSK value during registration. When the time interval for an old barcode expires, the app generates a new TOTP from the current timestamp and the PSK. This TOTP is appended to the user's unique ID, and together they are translated into a PDF-417 barcode. This updated barcode is then ready to be read by a barcode scanner(on a second mobile device using the camera), which transmits the corresponding ID and TOTP information to our web server where it is used to authenticate the user.

Tools

We used the cross-platform app development framework, Xamarin, to build the prototype of our system. We chose Xamarin because it uses the .NET framework, which provides reliability and programmability through its extensive libraries (including crypto-libraries) and web services. It also provides scalability as it allows easy installment of our app onto Windows and iOS devices if we wish to do so in the future. We also wanted to maintain a consistent language (C#) between client and server apps because it offered a streamlined development process.

For this project, we only used the Android platform for prototyping. This is because the alternative iOS platform greatly increases the complexity and costs of the project (due to paying for the Apple Developer's account), with little benefit to proving our concept [16]. Also, as we mentioned, we used a smartphone camera on the server side along with existing barcode scanner/generator libraries to demonstrate the uniqueness and one-time usage of the client's barcodes generated using our system.

VI. Evaluation

Evaluation Methodology

There will be three tests performed. One to test for the functionality of the system, another to test for its security against replay attacks, and finally a survey to test for the usability of our app.

To verify the functionality of the system, we wait for the client-side app to generate multiple barcodes, record them at no more than one barcode per update interval, and compare them with each other. A barcode generated at time X on the mobile app is scanned or saved as a screenshot. Another barcode generated later at time Y where $Y = X + updateinterval$ is also scanned or saved as a screenshot. The test was repeated many times to obtain more than ten samples of scanned values or screenshots of barcodes. These scanned values or screenshots were then compared with each other to demonstrate that the barcodes generated at different time intervals are not equivalent; each of them is unique. This test demonstrates the functionality of the barcodes to expire at the set intervals at which the TOTP expires, and generate newly updated barcodes with non-repeating TOTP values.

In order to demonstrate the security of our design, we performed replay attacks as we did on the original Starbucks app, and attempted to gain authentication via a screenshot of the user's barcode. A barcode scanner on the server side (we used a smartphone camera) was used to perform this test. Our system prevents the problem of unauthorized purchases, which was demonstrated by the server giving an "unauthorized" error and denying access to attackers with screenshots of the expired barcodes, and allowing access with "authorized" messages only to legitimate users with the correct barcodes generated from TOTPs.

To determine the usability of the application, we deployed the mobile app to a group of students with Android devices and surveyed their thoughts on the app. Respondents were given a short paragraph to read on the improvements that our app provides, and then they were asked to (1) rate the ease-of-use of the current Starbucks app on a scale of one to ten, (2) rate the ease-of-use of our app on a scale of one to ten, and (3) answer if they would continue to use the Starbucks mobile payment app if our system was implemented in its place.

Results of the Evaluation

The functionality test for the uniqueness of the barcodes generated using our system was a success in every instance. Within the ten samples, each barcode generated was unique and non-repeating.

The security test which attempted replay attacks were successfully blocked by our system's design. The server was able to differentiate between legitimate users and adversaries. The app rejected the adversary with "unauthorized" error and authenticated the user with "authorized" message in each iteration of the test.

Our usability survey collected feedback on our design from twelve UBC students who already use the Starbucks app. All respondents rated the ease-of-use of Starbucks' current app (with static barcodes) at an eight or above on a scale of one to ten. All respondents rated our redesigned app with the same or higher score that they gave the original app. This results in Starbucks' original app being rated 8/10 for usability, and our app being rated 8.4/10 for the same feature [17]. All respondents affirmed that they would continue to use the Starbucks app if it replaced its current static barcode system with our redesigned expiring barcode system.

Discussion of the Evaluation Results

By demonstrating the uniqueness of the barcodes generated by our application for the same user at different times, we have demonstrated the major improvement our design makes to the current static barcode system. Due to the expiring nature of our barcodes, adversaries now have a much smaller window of attack, which greatly reduces the chances of a replay attack being effective.

Through the successful blocking of replay attacks, we have demonstrated that our design is functional and more secure than the current payment system. This test demonstrated the correctness of our initial key exchange, as well as the secure authentication of each client on the server using correctly generated TOTP values. Thus we have satisfied the major goal of our design project, which is to improve the security of the Starbucks app and provide a countermeasure for its vulnerability to replay attacks.

We had a small sample size for user evaluation of usability (twelve UBC students), and before implementing our system in the real world we would seek to have larger sample sizes to verify our preliminary results. However, we were pleased that all of our respondents unanimously agreed that our system was on par with the already popular Starbucks payment system. The fact that all respondents gave our app the same rating or higher for ease-of-use when compared to the original app indicates that we have succeeded in adhering to the economy of mechanism and psychological acceptability design principles. Most importantly, all those surveyed responded in the affirmative when asked if they would continue to use the app if our system was implemented, which indicates that Starbucks would be unlikely to see any loss of popularity for their app if they chose to implement our payment system. Finally, an added benefit of our survey was that we managed to educate our respondents on the security vulnerabilities that exist in the current payment system, and we raised awareness of the usefulness and simplicity of TOTP authentication systems.

VII. Discussion

The chief advantage of our design is that it greatly improves the security of barcode transactions while remaining completely unobtrusive for customers. Our system is fully automated; unlike other systems that rely on TOTP values, we do not require the user to manually enter the one-time key, and switch back and forth between apps. The initial key exchange takes place behind the scenes and does not impact the user's experience in any way. The TOTP generated by the app is also completely hidden during use. The only thing the user sees when they use our app is the barcode they need to scan at the register. In these ways, our app is precisely as easy to use as the current Starbucks app. Usability was the focus of this design project. If we did not maintain the psychological acceptability of the payment system, our design would have been like the projects we noted in our Related Work section: a good for security, but too slow and complex for use in real-life. We avoided this issue by ensuring that our app is just as easy to use as the original, and our survey respondents confirmed our success on this front.

Another advantage of our app is its low capabilities requirement. Unlike most of the projects listed in our Related Work section, our design does not require the use of a phone's camera (we only used a phone's camera as a substitute for a barcode scanner for testing purposes). As mentioned, phone cameras can be finicky and calibration can be fraught with errors that discourage users from adopting the system. In addition, the TOTP and barcode generation in our app requires minimal performance overhead, meaning the app and server remains fast and responsive. Our application does not require constant communication between the client's device and the server, and is able to function even without internet access. This reserves the device's battery and data resources and provides convenience to users where there is no internet access, making it feasible to implement for real-world power-sensitive end devices.

Finally, a major advantage in consideration of Starbucks is that it would require minimal changes to the existing system in order to be implemented in the real-world. Starbucks would not need to purchase new barcode scanners or new equipment. Their app and servers would only require a minor update to implement TOTP barcodes feature. Those who already have the Starbucks app installed would not need to request for a new PSK. An important criteria for any new security system to meet are economic concerns. Our system is cheap to implement, both in terms of cost and performance. In addition, if the results of our survey are any indication, customers feel more secure with our system integrated and, therefore, more likely to use Starbucks' mobile payment system knowing that the security of their assets is being considered.

The major disadvantage of our design is that it is still vulnerable to man-in-the-middle attacks. For example, if the user's device was compromised by malware, like a Trojan Horse, the adversary could theoretically obtain the value of the secret PSK key that is used to generate the TOTP values, which is merely a hash of the current timestamp using the PSK. While we acknowledge this possibility, we also point out that if a user's device is compromised by a Trojan, they likely have more pressing security concerns than their Starbucks account assets. Also, a need for time accuracy and synchronization is required for systems like TOTP that uses timestamps. We counter this disadvantage by allowing a small "grace period" that allows the server to still accept a barcode that is technically expired for a short amount of time. This accommodates for the slight differences in time between the server and the client's devices.

As we all know, there is no such thing as a perfectly secure system. We are aware that the security of our design could be improved further, but our goal from the outset was to provide substantial security improvements over the original system while rigorously maintaining the app's simplicity and usability for customers. In that sense, our design was a great success, and the vulnerabilities that remain are acceptable when considering the amount of work required to exploit them versus the relatively low monetary value of the assets at risk.

VIII. Conclusion

In this project, we studied the Starbucks mobile app payment system's vulnerability to replay attacks and noted that our study may easily be extrapolated to other payment systems that rely on static barcodes. We discussed the plausible social engineering attacks that an adversary may use to gain access to a user's account information and devised an adversary model to understand and predict the adversary's behaviour.

We explained the related work on the use of barcodes in proposed cryptosystems, and why those systems are unfeasible for a fast-paced environment. We showed that we were inspired by the Google Authenticator app, our design is easier to use since it does not require users to manually enter codes or switch between apps. It inspired us to come up with our own solution of a back-end two-factor TOTP authentication system to improve the security of barcode payment systems. We gave an overview of our design about the distribution of PSK between client and server and generation of unique expiring barcodes by appending TOTP to user IDs. We discussed the design principles, as well as implementation details of the two-part client-server prototype system and our tools used. We also detail the evaluation methodology for testing functionality, security, and usability of our system, then gave an interpretation for each result.

We considered the advantages and disadvantages of our design, and concluded that we have successfully created a practical and realistic solution to a real-world problem. We contended that the solution to the problem of improving mobile app payment security is easy and cheap to implement. History has shown that small simple improvements to security can go a long way towards securing our digital safety and privacy, and we hope that our design may contribute to this forward momentum. The simplicity and usability of our expiring barcodes design also make it applicable to a variety of relatively low-risk user authentication scenarios, such as using public printers at a library. We believe that just because the threat is not extreme does not mean that we can skimp on security. Finally, we hope that our design project may educate readers on the vulnerabilities inherent in static barcode systems, as well as raise awareness of two-factor authentication as an easy and practical way to improve day-to-day security in our lives.

REFERENCES

[1] T. Soper, "Mobile payments account for 21% of transactions at Starbucks as coffee giant rolls out new technology", GeekWire, 2015. [Online]. Available: http://www.geekwire.com/2015/mobile-payments-account-for-21-of-sales-at-starbucks-as-coffee-giant-rolls-out-new-technology/. [Accessed: 12- Oct- 2016].

[2] "Starbucks Mobile Payment 2d-code", 2d-code.co.uk, 2011. [Online]. Available: http://2d-code.co.uk/starbucks-mobile-payment/. [Accessed: 12- Oct- 2016].

[3] "Free Barcode Generator - Barcoding.com", Barcoding.com, 2016. [Online]. Available: http://barcoding.com/resources/barcode-generator/. [Accessed: 11- Oct- 2016].

[4] "Free Online Barcode Generator: Create Barcodes for Free!", Barcode.tec-it.com, 2016. [Online]. Available: http://barcode.tec-it.com/en/PDF417. [Accessed: 11- Oct- 2016].

[5] S. Clark, "'Hacker' shows how to crack Starbucks mobile payments app NFC World", NFC World, 2011. [Online]. Available: http://www.nfcworld.com/2011/02/09/35844/hacker-shows-how-to-crack-starbucks-mobile-payments-app/. [Accessed: 11- Oct- 2016].

[6] J. Smith, "Starbucks iPhone App Can be Hacked, Can be Used on Android", Notebooks.com, 2011. [Online]. Available: http://notebooks.com/2011/02/10/starbucks-iphone-app-can-be-hacked-can-be-used-on-android/. [Accessed: 11- Oct- 2016].

[7] "How-To: Starbucks coffee card on your BlackBerry- no app required :) - BlackBerry Forums at CrackBerry.com", Forums.crackberry.com, 2016. [Online]. Available: http://forums.crackberry.com/general-blackberry-discussion-f2/how-starbucks-coffee-card-your-blackberry-no-app-required-709953/. [Accessed: 12- Oct- 2016].

[8] J. Kincaid, "Google Is Making Your Account Vastly More Secure With Two-StepAuthentication", TechCrunch, 2016. [Online]. Available: https://techcrunch.com/2010/09/20/google-secure-password/. [Accessed: 11- Oct- 2016].

[9] "google/google-authenticator", GitHub, 2016. [Online]. Available: https://github.com/google/google-authenticator. [Accessed: 12- Oct- 2016].

[10] "RFC 6238 - TOTP: Time-based One-time Password Algorithm", Tools.ietf.org, 2016. [Online]. Available: https://tools.ietf.org/html/rfc6238. [Accessed: 12- Oct- 2016].

[11] "RFC 4226 - HOTP: An HMAC-Based One-Time Password Algorithm", Tools.ietf.org, 2016. [Online]. Available: https://tools.ietf.org/html/rfc4226. [Accessed: 09- Nov- 2016].

[12] "google/google-authenticator", GitHub, 2016. [Online]. Available: https://github.com/google/google-authenticator. [Accessed: 09- Nov- 2016].

[13] G. Starnberger, L. Froihofer and K. Goeschka, "QR-TAN: Secure Mobile Transaction Authentication", 2009 International Conference on Availability, Reliability and Security, 2009.

[14] J. M. McCune, A. Perrig, and M. K. Reiter, "Seeing-isbelieving: Using camera phones for human-verifiable authentication," in IEEE Symposium on Security and Privacy. IEEE Computer Society, 2005, pp. 110-124.

[15] D. E. Clarke, B. Gassend, T. Kotwal, M. Burnside, M. van Dijk, S. Devadas, and R. L. Rivest, "The untrusted computer problem and camera-based authentication," in Pervasive, ser. Lecture Notes in Computer Science, F. Mattern and M. Naghshineh, Eds., vol. 2414. Springer, 2002, pp. 114-124.

[16] "Purchase and Activation - Support - Apple Developer", Developer.apple.com, 2016. [Online]. Available: https://developer.apple.com/support/purchase-activation/. [Accessed: 10- Nov- 2016].

[17] J. Y. Bae, R. K. Garsuta, M. Mackenzie, A. Wang, "Expiring Barcodes Design Project Survey", 2016. [Print].

[18] "Best Buy Deploys QR Codes to Enhance Shopping Experience - retailgeek.com", retailgeek.com, 2016. [Online]. Available: http://retailgeek.com/best-buy-deploys-qr-codes-to-enhance-shopping-experience/. [Accessed: 11- Dec- 2016].

APPENDIX

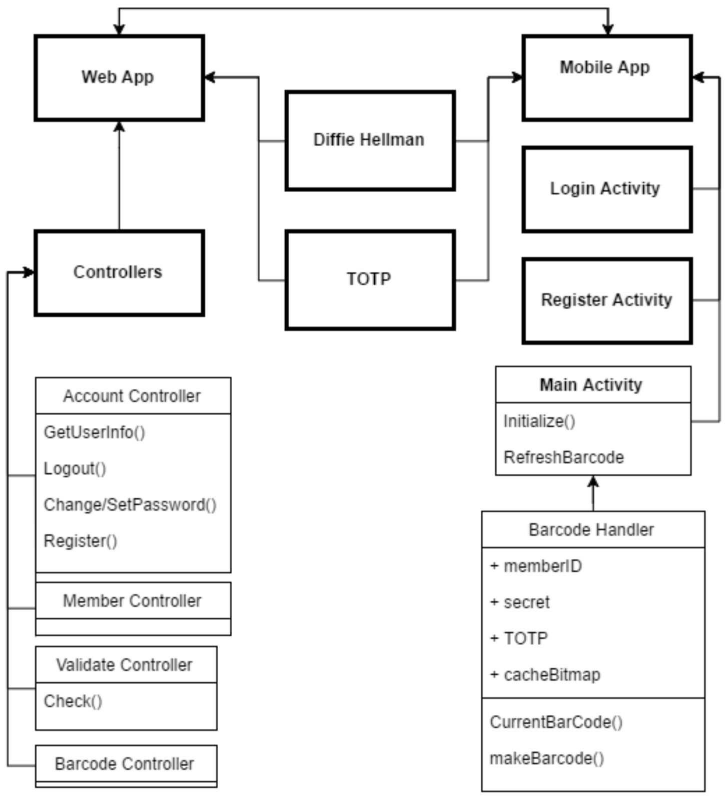

Fig. 1. Overview of Software Modules in the Prototype System