Security Vulnerabilities in Mobile Health Applications

Mehrdad Aliasgari
California State University, Long Beach
Long Beach, California 90840, USA
Email: Mehrdad.Aliasgari@csulb.edu

Michael Black
California State University, Long Beach
Long Beach, California 90840, USA
Email: michaelblxck@gmail.com

Nikhil Yadav
St. John's University
Queens, NY 11439, USA
Email: yadavn@stjohns.edu

***Abstract*—With the growth of mobile health (mHealth) applications, it is imperative to make sure that users' health records and their privacy are safeguarded properly. In this paper, twenty five mobile health applications are analyzed for security and privacy vulnerabilities. Each of these applications is available from the Google Play Store, has a medical tag, requests permission for full network access, and connects to a medical device. Network security and authentication vulnerabilities together with various Man in the Middle attacks, and HIPAA noncompliance for all applications are inspected. The results of the security and privacy tests indicate that each application suffers from security and/or privacy vulnerabilities. The application security and privacy vulnerabilities results of the analyzed applications was subsequently shared with their development teams. The results from analysis are beneficial for mHealth application developers to be more cognizant of such vulnerabilities. Furthermore mHealth users are encouraged to be more diligent in choosing a robust and secure mHealth application.**

***Keywords*—Mobile Health, security, privacy, HIPAA**

I. INTRODUCTION

Mobile health (mHealth) applications are an ever-expanding frontier in today's use of technology. They allow a user to record health data, contact their doctor, and connect to medical devices from the convenience of a smartphone. According to [1] fifty-eight percent of Americans use mHealth applications. A major concern with such mHealth applications is user privacy loss and its implications in case of a data breach [2]. It is vital for developers of such applications to safeguard user data and define preventive security measures. Additionally, the data transmitted and stored by an mHealth application may be considered Protected Health Information (PHI). PHI is protected by the Healthcare Insurance Portability and Accountability Act (HIPAA), and requires additional security measures.

In this paper the security and privacy protection mechanisms of the top twenty-five Android mHealth applications are explored. Security vulnerabilities in data transmission, unencrypted data revealed by a Man in the Middle (MitM) attack, and HIPAA noncompliance in privacy measures is examined for all the applications.

The rest of this paper is organized as follows: Section II looks at related work in this area; Section III gives a background on the security protocols and tools used for our experiments, and HIPAA obligations. An overview of our testing environment and setup is described in section IV; Section V looks at the results from analysis and observations in experimentation; Finally, section VI concludes the paper with key findings along with best practices to avoid such vulnerabilities. The results show that all the mHealth applications suffer from some vulnerabilities one way or the other. This work can be used by developers and users to be cognizant of such security and privacy pitfalls.

II. RELATED WORK

Security of mHealth technologies has been studied extensively. As noted by Arori, et al., however, there is a sparsity in the research literature despite rapid growth of this mHealth technology [3]. According to Arori, et al., a major concern for mHealth is the secure transfer of sensitive data. In [3], Arori, et al., concluded that at a minimum the HIPAA standards for encryption [4] should be upheld in mHealth technologies. They also proposed that sensitive information should, in some cases, be replaced by non revealing code words in the event that the data is intercepted by an MitM attack. Furthermore, they encouraged two factor authentication for patient logins and consistent auditing of security systems.

Islam, et al., [5] reiterated the need for a base in general computer security concerns for mHealth applications including confidentiality, integrity, authentication, and availability. They additionally reported requirements in the freshness of data so that an adversary cannot replay messages, non repudiation so that a message cannot be denied in sending, authorization for nodes across the network, resiliency, fault tolerance, and self-healing in the case of failures.

There have also been studies done on the security of general mobile applications. Lawrence [6] found that eighty percent of the top two-hundred most downloaded mobile applications have the ability to opt out of using HTTPS. Onwuzurike, et al., [7] tested mobile applications for TLS configuration and found that forty-eight of the top one-hundred most downloaded Android applications had insecure configurations.

Some mHealth applications can connect to sensors and devices. Bandyopadhyay, et al., [8] found that in addition to personal information, devices can also store and transmit a user's physical location and movement in real time. Halperin, et al., [9] surveyed data related to medical devices that in particular could reveal a patient's diagnosis, treatment history, allergies, and medications.

In this paper, the top twenty five Android mHealth applications are examined. Each of these appications can connect to medical devices and require network connection. To the best of our knowledge, this is the first work that examines in-depth security, as well as privacy vulnerabilities of such mHealth applications.

III. Background

In this section Transport Layer Security (TLS) protocol and the HIPPA act is overviewed. Tools used for our experiments are then highlighted.

TLS protocol

Transport Layer Security (TLS) is an Internet protocol that provides data integrity and privacy between two communicating nodes [10]. It was preceded by the Secure Sockets Layer (SSL) and the two acronyms are often used interchangeably. TLS is more secure than SSL due to its addition of authentication to messages and improved algorithms for cipher suites. Each TLS connection between nodes starts with an HTTP handshake [11]. The nodes agree on which cipher suite will be used during this handshake and thereafter encrypt their correspondences with that cipher suite.

A cipher suite is a set of methods used for encrypting data sent between two nodes [11]. Cipher suites are used by TLS and SSL protocols and usually include a key exchange algorithm, an authentication algorithm, a bulk encryption algorithm for encrypting the message data, and a Message Authentication Code (MAC) algorithm for checking the authenticity of a message.

Servers configured to use TLS can enable Strict Transport Security (HSTS). HSTS is a protocol that protects against HTTP Downgrade Attacks [12]. Adversaries use these attacks to force a server to communicate using HTTP, allowing the attacker to view data in plain text. HSTS only allows a server to communicate with a client when using HTTPS to ensure that the channel is encrypted.

There are known vulnerabilities that proper TLS configuration protects against. Three of these vulnerabilities emerged during our experiments. They are OpenSSL Padding Oracle vulnerability, RC4 insecurity, and using common Diffie-Hellman primes.

The OpenSSL Padding Oracle vulnerability is made possible by the use of padding in block ciphers [13]. Here, a block can be thought of a chunk of information. Encrypted messages need to be be padded with extra characters if the message does not take up the complete space of a multiple of blocks. A Padding Oracle is an approach that helps expose whether or not the used padding is acceptable. An adversary can use this oracle to reverse engineer bytes of a message by tampering with the content and observing whether or not the padding is valid.

RC4 is a cipher that has been observed to be insecure due to several vulnerabilities and using this cipher weakens a server's implementation of TLS [14].

Using common primes is insecure when Diffie-Hellman key exchange (DH) method is selected [15]. DH is a method of exchanging keys between a client and a server so that their communication can be encrypted. This exchange uses an algorithm that relies on very large prime numbers. Using common prime numbers for this algorithm gives an adversary the ability to use a lookup table or make feasible computations in order to decrypt messages and view them in plain text.

HIPAA

HIPAA defines PHI as any information that is "created or received by a health care provider, health plan, public health authority, employer, life insurer, school or university" and "relates to the past, present, or future physical or mental health or condition of any individual" [16]. HIPAA is designed to secure PHI by addressing audit control (the ability for a user to read and modify their data), unique user identification, automatic logoff, encryption and decryption, auditing, authentication, data storage of PHI, transmission security, and notifications of security breaches.

Failure to comply with HIPAA's standards when dealing with PHI can result in a fine [17]. Unknowingly violating HIPAA results in a $100 minimum fine, reasonably violating HIPAA results in a $1,000 minimum fine, and willingly neglecting to comply with HIPAA results in a $10,000 fine. Multiple fines can be given and they can reach up to an annual maximum of $1,500,000 per individual or company.

Tools

Wireshark, SSL Labs, and Fiddler are tools used in our experimentation process. In this section a brief overview of these tools is provided.

A. Wireshark

Wireshark is a an open source tool used to capture packets and analyze the use of Internet protocols [18]. Wireshark was used to examine the HTTP requests made by each app in this paper. It gave us information about which servers an application makes API calls to.

B. SSL Labs

SSL Labs is a free-to-use website that tests a server's certificate and configuration to determine the server's TLS security grade [19]. It also gives information regarding the security of a server's protocol support, key exchange, and cipher strength. SSL Labs provides a detailed description of handshake simulations using different platforms including Android, Chrome, and Firefox. Additionally, it tests a server for various known vulnerabilities, including OpenSSL Padding Oracle vulnerability [13], RC4 insecurity [14], and using common Diffie-Hellman primes [15].

C. Fiddler

Fiddler is an open source web debugging proxy [20]. A proxy is a middleman between two communicating nodes that is used to filter or observe the nodes' messages. A reverse proxy is a proxy that forwards resources to server nodes from client nodes [21]. We used Fiddler as a reverse proxy to perform MitM attacks on the applications. An MitM attack is one in which an adversary intercepts messages of a node before forwarding them to the intended receiver. An adversary can either eavesdrop on the messages as a passive adversary or modify the messages as an active adversary.

TABLE I
LIST OF TESTED APPLICATIONS

ACCU-CHEK	Missi Remote Health
Air Smart Spirometer	MobECG
Alert	MyChart
Blood Glucose Tracker	mySugr
Cornerstones4Care	PeriCoach
FibriCheck	Qardio Heart Health
Health2Sync	SaniQ Asthma
Heart Rate Monitor	TACTIO HEALTH
HoMedics	TempTraq
iHealth Gluco-Smart	UDI Check
iHealth MyVitals	Valedo
Mate	Walgreens Connect
MedM Health	

IV. ENVIRONMENT

The following section gives insight into our environment. The applications tested are specified, togther with the environment set up, searching for HIPAA compliance, and ethics.

APPLICATIONS

The top twenty five applications ranked by Google Play store as shown in Table I were identified as candidates. Each of these applications were downloaded from the Google Play store and had to receive either a medical or healthcare tag. In addition, each of the applications had to request full network access and have the ability to connect to a medical device.

SETUP

Our testing environment included a Lenovo Yoga laptop running Kali Linux Rolling and a Samsung Galaxy S-6 running Android 7.0 (Nougat). The latter was used for providing Wifi connection to a Pineapple Tetra auditing system connected to the laptop that was used to analyze packets via Wireshark. SSL Labs was accessed on the laptop and utilized to test for TLS configurations. Fiddler was also accessed on the laptop and configured as a reverse proxy. The phone used as a host for the applications was a Samsung Galaxy S-5 running Android 6.0 (Marshmallow).

Wireshark was applied to detect which server an application made API calls to. The domain name of the server was entered into SSL Labs. This provided information about the server's configuration, including TLS version, whether or not the server's certificate is trusted, and the preferred cipher suites. This domain name was revealed by using the IP's in HTTP requests made by the application observed in Wireshark.

Fiddler was configured as a reverse proxy on the laptop. It was configured to generate a Certificate Authority certificate. This new certifcate was added to the list of Certificate Authority's on the test Android device. For each domain name, a new TLS certificate was generated signed by the Certificate Authority's private key. Fiddler replaces the genuine certificate that belongs to a domain with this newly signed "fake" certificate. This allows Fiddler to act as a MiTM adversary and intercept, decrypt, re-encrypt and forward any message exchanged between the test device and domain name servers. All packages without any TLS encryption were then inspected.

SEARCHING FOR HIPAA COMPLIANCE

HIPAA requires that all application developers provide confidentiality, integrity, and availability when processing a user's PHI. Additionally, these developers must protect their users from reasonably anticipated threats. HIPAA also explicitly addresses the need for a developer to encrypt users' PHI [22]. These requirements were kept in mind when testing the applications. It was determined that if a mobile app is obligated to be HIPAA compliant, that app must also be responsible for implementing the mechanisms specified by HIPAA.

Searching for HIPAA compliance was first performed by reading through each application's terms and agreements, websites, and Google Play Store descriptions. Initially all development teams for the applications were emailed with an inquiry regarding general HIPAA compliance questions. These same development teams were sent a follow up email with more specific HIPAA compliance questions regardless of whether or not they had responded to the initial email. Not all teams replied.

ETHICS

Throughout our research, no individual's data was subject to our investigations. All user info within apps was generated using fake data, including a fake email address, name, and password. All devices used in this work are owned by the researchers and at no point was any application code tampered with in any way. Upon completion of this work, the respective application development teams were notified of the results and given ample time to respond.

V. RESULTS

Our observations can be broken down into the following categories:

- *SSL Lab Results:* This includes TLS version, key exchange, cipher suites, and HSTS.
- *Fiddler Results:* This focuses on results of unencrypted sensitive data using a Man in the Middle attack and determining which applications may be subject to HIPAA compliance.
- *HIPAA Compliance Results:* How accountable the application is in terms of HIPAA.

SSL Lab Results

One of the twenty-five mobile applications (Heart Rate Monitor) makes no API calls to a server. Three of the applications (Gluco-Smart, MyVitals, Missi Remote Health) made API calls to a server with an untrusted certificate, giving them a "T" rating. Only four of the applications (Health2Sync, MedM Health, Mob ECG, and SaniQ Asthma) received the ideal grade of A+. The remaining applications were given poor grades from SSL Labs, ranging from A to F ratings, as seen in Table II. Grade of A+ is an indicator of appropriate TLS

TABLE II
SSL LABS GRADE RESULTS

App	Website SSL Labs Rating
ACCU-CHEK	A
Air Smart Spirometer	A
Alert	C
Blood Glucose Tracker	A
Cornerstones4Care	A
Fibri Check	A-
Health2Sync	A+
Heart Rate Monitor	N/A
HoMedics	A
iHealth Gluco-Smart	T
iHealth MyVitals	T
Mate	C
MedM Health	A+
Missi Remote Health	T
MobECG	A+
MyChart	A+
mySugr	B
PeriCoach	A
Qardio Heart Health	A
SaniQ Asthma	A+
TACTIO HEALTH	A
TempTraq	A
UDI Check	A
Valedo	F
Walgreens Connect	A

TABLE III
CIPHER SUITES OF TLS SERVER

App	Highest and Lowest Cipher Suite
ACCU-CHEK	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA WEAK
Air Smart Spirometer	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
Alert	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_3DES_EDE_CBC_SHA WEAK
Blood Glucose Tracker	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_CAMELLIA_128_CBC_SHA WEAK
Cornerstones4Care	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_3DES_EDE_CBC_SHA WEAK
Fibri Check	No preference on cipher suites
Health2Sync	No preference on cipher suites
Heart Rate Monitor	N/A
HoMedics	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA WEAK
iHealth Gluco-Smart	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_128_CBC_SHA WEAK
iHealth MyVitals	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_128_CBC_SHA WEAK
Mate	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_3DES_EDE_CBC_SHA WEAK
MedM Health	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_DHE_RSA_WITH_AES_128_CBC_SHA
Missi Remote Health	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_256_CBC_SHA WEAK
MobECG	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 TLS_RSA_WITH_3DES_EDE_CBC_SHA WEAK
MyChart	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_128_CBC_SHA WEAK
mySugr	No preference on cipher suites
PeriCoach	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
Qardio Heart Health	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_3DES_EDE_CBC_SHA WEAK
SaniQ Asthma	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_CAMELLIA_128_CBC_SHA WEAK
TACTIO HEALTH	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA WEAK
TempTraq	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 TLS_RSA_WITH_3DES_EDE_CBC_SHA WEAK
UDI Check	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA WEAK
Valedo	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_DHE_RSA_WITH_AES_256_CBC_SHA WEAK
Walgreens Connect	No preference on cipher suites

configuration. Therefore, only four out of twenty-five of the tested applications (16%) pass this test.

Only one application (PeriCoach) made API calls to a server that supported TLS version 1.3. Twenty-four of the application servers supported TLS v1.2. Only one application server (Air Smart Spirometer) refrained from supporting TLS 1.1. Only two application servers (Air Smart Spirometer and Walgreens Connect) refrained from supporting TLS 1.0. None of the application servers supported SSL 3 nor SSL 2. In March of 2018, Internet Engineering Task Force (IETF) approved the final draft for TLS v1.3 ([23]). It is recommended that developers utilize this new standard along with TLS 1.2 instead of any previous TLS versions.

In addition, only three of the applications used a 4096 bit RSA key while twenty-one of the applications used a 2048 bit RSA key. Twenty-four applications used SHA256 with RSA for their signing algorithm. The higher key length, the exponentially higher degree of security an RSA system can offer. However, with increased key length comes decreased performance. It is understood that in today's world, 2048 bits is the minimum RSA key length [24].

Four of the applications communicated with servers that had no preference on the order of accepted cipher suites. This is potentially dangerous because an adversary could force the server to use a weak cipher suite. Only one of the applications (MedM Health) used no weak cipher suites. An adversary can force weak cipher methods for the remaining twenty-four application and potentially defeat any encryption protection. The strongest and weakest cipher suits for each application is shown in Table III.

Only three of the application servers (MedM Health, MobECG, SaniQ Asthma) supported HSTS. One of the applications (TACTIO HEALTH) had the ability to support HSTS but had this feature disabled. The remaining applications did not support HSTS. Therefore, twenty-two of the applications suffer from HTTP Downgrade Attack.

Fiddler Results

Upon investigation by the Fiddler server, five applications (iHealth MyVitals, MedM Health, MobECG, Valedo, Heart Rate Monitor) did not send any unprotected user data to a server. The remaining twenty applications collected and transmitted user data that was intercepted by a MitM attack and read in plain text.

Twelve of the applications directly sent the patient's password, while sixteen sent the patient's name, and eighteen sent the patient's email to a server. Every application that directly sent the user's password without protection also transmitted the user's email. Given weak TLS configuration by these applications, an adversary could capture users emails and passwords that are potentially re-used across other services.

It was discovered that applications collected and transmitted patients' sensitive data such as weight, height, birthday, gender, their address, and their citizenship status along with

Fig. 1. Fiddler Results: Number of apps that collected and transmitted data in each category

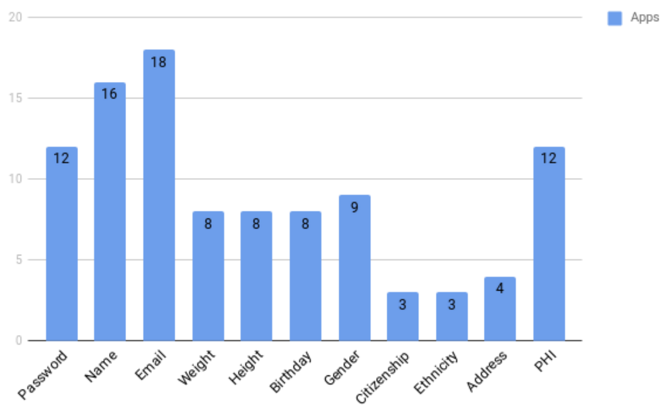
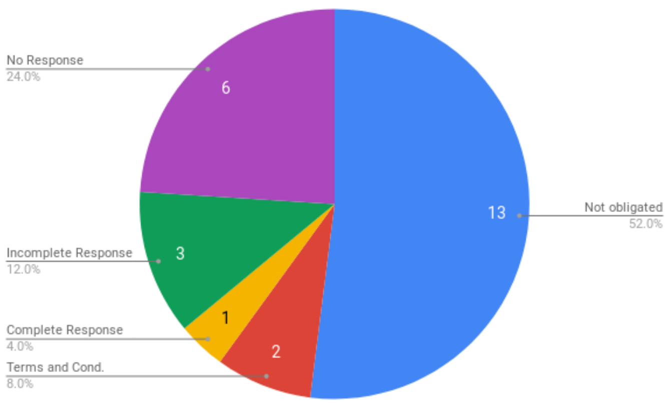


Fig. 2. HIPAA compliance



ethnicity data. This information could be utilized by an adversary for social engineering, bribery, identity theft, and guessing answers to a user's private login security questions. Figure 1 visualizes the above information and the number of applications that collected and transmitted such information, albeit insecurely.

Furthermore, out of the twenty five tested applications, twelve also collected and transmitted PHI, making them subject to HIPAA's compliance laws. Six collected and transmitted a patient's measurements including glucose and blood sugar levels. Two sent patients' medical history to their servers. Six transmitted identification of patients' clinicians, and three transmitted prescriptions of their users. Three transmitted users' conditions, including whether or not the user was pregnant, had epilepsy, or had seizures. Lastly, three applications transmitted information about the user's medical devices including device types and its location.

Of the twelve applications that collected and transmitted PHI, only one passed the TLS configuration test (Health2Sync). The remaining eleven are not providing appropriate measures of data security when transmitting PHI to their servers. We recommend such applications to improve their TLS configurations.

HIPAA Compliance Results

Of twelve applications likely to be subject to HIPAA compliance as determined by Fiddler results, only two (TACTIO HEALTH, Walgreens Connect) discussed HIPAA in their terms and conditions. To investigate further, an email was sent to each of the applications' development teams inquiring on the acknowledgement of HIPAA compliance.

Only one application (Cornerstones4Care) responded to us and clearly defined how their application handled each of the HIPAA specifications outlined in our emails. Three more applications (Qardio, Alert, Mate) responded to our emails claiming to be HIPAA compliant, however, they did not disclose how this was achieved. The remaining applications that were found to collect and transmit PHI in our Fiddler test did not respond to our inquiries.

The Glooko support team behind Cornerstones4Care replied with complete response to our emails with a link to a white paper discussing their HIPAA compliance policy. The paper states that Glooko provides reasonable access control, unique user identification, automatic logoff, and uses end to end encryption with AES 256 to satisfy the security of health related data transmission. In addition, the white paper mentions Glooko's policies on secure data storage, notification of data breaches, and its application auditing procedure. All teams are encouraged to include a clear policy on how they achieve HIPAA compliance within their applications' terms and conditions. In addition, it is recommended to include a link to such policies within an application.

In addition the development teams of the thirteen applications that did not appear to transmit any PHI based on our Fiddler test observations were contacted. If an application stores PHI, they still are subject to HIPAA compliancy. Only two such applications (MedM, MySugr) responded to our emails and explained why they are not expected to be HIPAA compliant. The others did not respond giving us no way to investiage whether they are obliged to and are compliant to HIPAA specifications. Figure 2 provides a visual representation of the above information.

VI. CONCLUSION

In this work, the security and privacy of the top twenty-five mHealth Android applications with capability to connect to medical devices are examined. Tests were conducted to examine their use of secure Transport Layer Security protocol (TLS). In addition, a Man-in-the-Middle attack was run to investigate what data was collected and transmitted to servers. Furthermore, HIPAA compliance was also investigated. Twenty-one of the twenty-five applications have weak TLS configurations, twelve revealed passwords in a Man-in-the-Middle attack, and twelve applications collect and transmit PHI data. Of the twenty-five applications, only one was completely transparent in how it is HIPAA compliant, and only two explicitly mention HIPAA in their terms and agreements in the application. There was no application that had ideal TLS configuration while implementing remote password login and being transparent in their HIPPA compliance. Such security and privacy weaknesses are alarming to mHealth applications that deal with private PHI.

The results were shared with the developers of the tested mHealth applications. It is strongly suggested that mHealth developers be more aware of TLS security for servers, remote login and proper protection for user passwords and PHI, as well as HIPAA compliance where it is necessary.

Additionally, it is imperative that application developers consistently audit their teams. Applications liable for HIPAA compliance need to enforce audits in particular because Section 164.312(b) of the act requires it. It is recommended that transmitted PHI is aliased as code words and the values are encrypted to alleviate the consequences of a MitM attack. Finally, users of mHealth applications need to be aware of security threats and should research and vet an mHealth application prior to using it.

In the future, a larger pool of mHealth applications can be analyzed. In fact, developing a user-friendly, easy-to-understand security rating meachnism for mHealth applications can help users make more informed decisions. In addition, secure data storage needs to be included in such a security benchmark. Researching how PHI is stored in mHealth servers could also give us a more complete understanding of additional pitfalls and vulnerabilities in the mHealth market. Lastly, investigatation of the vulnerabilities among sensors and hardware devices that connect to such mHealth applications is left as a future work.

REFERENCES

[1] P. Krebs and D. T. Duncan, "Health app use among us mobile phone owners: a national survey," *JMIR mHealth and uHealth*, vol. 3, no. 4, 2015.

[2] "Notice of data event: Primary health care inc." https://www.prnewswire.com/news-releases/notice-of-data-event-primary-health-care-inc-300615413.html, 2018.

[3] S. Arora, J. Yttri, and W. Nilsen, "Privacy and security in mobile health (mhealth) research," *Alcohol research: current reviews*, vol. 36, no. 1, p. 143, 2014.

[4] "The hipaa privacy rule." https://www.hhs.gov/hipaa/for-professionals/privacy/index.html, 2018.

[5] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The internet of things for health care: a comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[6] B. Lawrence, "5 vital tips for developing hipaa compliant mobile apps: A checklist." https://www.nowsecure.com/blog/2017/03/23/5-vital-tips-developing-hipaa-compliant-mobile-apps-checklist/, 2018.

[7] L. Onwuzurike and E. De Cristofaro, "Danger is my middle name: experimenting with ssl vulnerabilities in android apps," in *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2015, p. 15.

[8] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," *Wireless Personal Communications*, vol. 58, no. 1, pp. 49–69, 2011.

[9] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel, "Security and privacy for implantable medical devices," *IEEE pervasive computing*, no. 1, pp. 30–39, 2008.

[10] S. Turner, "Transport layer security," *IEEE Internet Computing*, vol. 18, no. 6, pp. 60–63, 2014.

[11] H. Krawczyk, K. G. Paterson, and H. Wee, "On the security of the tls protocol: A systematic analysis," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 429–448.

[12] J. Hodges, C. Jackson, and A. Barth, "Http strict transport security (hsts)," Tech. Rep., 2012.

[13] Y. Sheffer, R. Holz, and P. Saint-Andre, "Summarizing known attacks on transport layer security (tls) and datagram tls (dtls)," Tech. Rep., 2015.

[14] A. Popov, "Prohibiting rc4 cipher suites," Tech. Rep., 2015.

[15] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta *et al.*, "Imperfect forward secrecy: How diffie-hellman fails in practice," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 5–17.

[16] "Hipaa protected health information': What does phi include?" https://www.hipaa.com/hipaa-protected-health-information-what-does-phi-include/, accessed in spring 2018.

[17] "Hipaa violations and enforcement." https://www.ama-assn.org/practice-management/hipaa-violations-enforcement, accessed in spring 2018.

[18] "Wireshark," https://www.wireshark.org/, accessed in spring 2018.

[19] "Ssl labs," https://www.ssllabs.com/, accessed in spring 2018.

[20] "Telerik fiddler," https://www.telerik.com/fiddler, accessed in spring 2018.

[21] K. Araujo, R. Best, D. Heitmueller, and D. Tikhonov, "Network access using reverse proxy," Nov. 24 2005, uS Patent App. 11/078,001.

[22] "Summary of the hipaa security rule." https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html, accessed in spring 2018.

[23] I. E. T. F. Network Working Group, "The transport layer security (tls) protocol version 1.3," https://tools.ietf.org/html/draft-ietf-tls-tls13-28, 2018.

[24] R. Sinha, H. K. Srivastava, and S. Gupta, "Performance based comparison study of rsa and elliptic curve cryptography," *International Journal of Scientific & Engineering Research*, vol. 4, no. 5, pp. 720–725, 2013.