MSF:A comprehensive security framework for mHealth applications

Medha Srivastava
Computing & Software Systems
University of Washington Bothell
Bothell, USA
medhas31@uw.edu

Geethapriya Thamilarasu
Computing & Software Systems
University of Washington Bothell
Bothell, USA
geetha@uw.edu

***Abstract*—Mobile health (mHealth) applications are growing in popularity due to their effectiveness in delivering healthcare services and improving patient health outcomes. mHealth apps are also increasingly targeted by attackers as they handle sensitive and private medical and personal data. Existing research indicates that lack of security cognizance among app developers is the leading cause of security vulnerabilities in mHealth apps. In this paper, we propose a security framework that provides authentication, authorization, secure storage and transmission of medical data and workflows for mHealth applications. The proposed framework can be easily imported into new and existing mHealth apps to mitigate its security and privacy vulnerabilities. To prove the framework's effectiveness, we use a dummy mHealth app built on top of it and confirm via our findings that the framework can be easily integrated into any mHealth app and reduce its security and privacy risks without compromising user experience.**

***Index Terms*—mHealth, Security, Privacy, Android, Security framework, Mobile Security**

I. Introduction

In recent years we have witnessed an unparalleled growth of mobile applications particularly more prominent in healthcare domain [1]–[3]. The term mobile health or mHealth refers to the practice of medical and public health using mobile devices such as smartphones, personal digital assistants (PDAs), patient monitoring devices and other wireless devices. Using mHealth applications, doctors are able to easily access medical data and patient information, prescribe medications and effectively monitor patient health conditions [2]. mHealth apps also enable patients to educate themselves about their health, attain fitness goals and track their own wellness [3]. Patients and caregivers can easily view their prescriptions and access or update their medical records via their smartphones. It is evident that mHealth apps benefit all stakeholders doctors, patients, caregivers, and hospitals by offering quick, resilient and convenient care tools. This has resulted in improved decision making, better diagnosis & treatment and eventually superior patient outcomes.

Despite the continued growth and range of benefits offered by mHealth apps, they face numerous security and privacy challenges [5]–[8]. mHealth apps deal with extremely sensitive data such as patients personally identifiable information, financial and insurance information and health records. Storage of such medical data on the mobile devices or its transmission to and from the devices raises significant security and privacy concerns [10], [11]. The sensitive nature of this data coupled with the fact that mHealth apps have extremely naive security implementations [13] cause these apps to face higher security and privacy risks compared to other domains. Security challenges faced by mHealth apps such as leakage of information, storing of the data in unencrypted form on the phones physical storage, sharing of information with third party services including social networks [4], [8] are also prevalent in other mobile application domains such as retail or banking. However, the critical nature of such threats is compounded for mHealth apps because of the higher sensitivity of the data as compared to apps of other categories. mHealth apps for instance, tend to collect extremely private data such as patients activities, location, lifestyle, dietary and eating habits apart from their personally identifiable information for a prolonged period of time [4]. Furthermore, since these apps often allow remote access of patient data to healthcare professionals, they are also subject to increased attack vectors [9].

Vulnerability analysis of top 15 Android based mHealth apps conducted in 2015 yielded almost 250 security vulnerabilities [13]. This research demonstrated that the vulnerabilities are mostly due to software development mistakes such as improper permissions, invalid or weak SSL/TLS protocol usage, bad choice of cipher suite with no integrity and storing sensitive data in external storage among others. Such inadequate coding and security engineering practices form the entry point for attackers to cause identity theft, disclosure threats, privilege escalation attacks and side channel attacks [9], [13].

This lack of security cognizance among developers of mHealth apps can become a severe bottleneck in its widespread adoption and usage. To this end, a security framework that app developers can use and rely upon to fill the gap caused by their lack of awareness of security principles can prove vital in considerably reducing the vulnerabilities that exist in mHealth apps today.

The rest of the paper is organized as follows: In Section II, we discuss the background and related work for existing security frameworks for Android mHelath apps. In Section III, we describe the design and architecture of our proposed mHealth security framework. Section IV lists the features offered by our proposed framework. In Section V, we describe the effectiveness of the framework in a simulated environment. Finally, Section VI provides the concluding remarks on this research.

II. Background & Related Work

The rise in cyber-attacks targeting healthcare systems has been the topic of numerous studies and analysis with focus on the collection, storage and transmission of patient medical data [14]–[16]. Users of mHealth apps are often unaware of the substantial risk on their privacy and security of their data associated with using the app [6], [17]. Most popular mHealth apps dont have basic security mechanisms, where sometimes more than 70% of these apps store or transfer sensitive data without any encryption [8], [13].

Ahmed *et al.* addressed the privacy and security concerns in mHealth apps and proposed the tagging of health information to identify sensitive information [18]. Mitchell *et al.* proposed a one-policy framework to provide guidelines for developing mHealth apps [19]. Murad *et al.* proposed a mechanism to provide security to mHealth applications running on devices used by paramedics [20], however, their solution was limited to a single device. Simplicio *et al.* proposed SecourHealth security framework focused on securing the medical data collected by mHealth applications while on the devices storage or in transit [21]. This was also one of the first attempts to build an application framework that could be leveraged by mHealth apps to improve their security. However, it severely restricted the user experience for the consumers of those mHealth apps. Despite being great solutions, existing literature could not fully empower the app developers in securing their mHealth apps. Liu *et al.* proposed a security framework that enables secure transmission of electronic medical records and personal healthcare information [22]. This solution however expects digital health stakeholders to significantly change their networking infrastructure which might not be feasible. Also, it only ensures the security of data during transit and not when it is being collected at the client's device or monitor.

Lakin *et al.* demonstrated that majority of the vulnerabilities found in top 15 Android and iOS mHealth applications, including the top three vulnerabilities, were a result of bad coding and improper security engineering practices [13]. To that end, they proposed a framework to mitigate some of those reported vulnerabilities such as authentication and encryption of medical data by preventing app developers from making bad choices during the development process. However, one of the biggest disadvantages of their proposed framework is that it was implemented as a prototype rather than an actual framework that can be used by app developers. Also, some of the implemented features such as device rooting detection no longer apply to the Android OS since Google made significant improvements to the Android OS and Android app development processes in 2017. Finally, the framework fails to address some major vulnerabilities reported in the paper such as weak SSL/TLS.

Hussain *et al.* proposed the mHealth apps security framework (MASF) focusing on securing the medical data associated with Android mHealth apps as well as protecting the privacy of the users using those apps [4]. This framework too proved extremely effective in addressing concerns such as authentication and encryption of the medical data being accessed, leakage of information, device mis-bonding attacks and privilege escalation attacks. It was also able to thwart attacks that were a direct result of the flaws existing in the base OS itself. However, a key issue that seriously hampers the adoptability of the framework is that it doesnt work on Android OS that comes with everyday mobile phones. Since the framework adds hooks to OS internals, it can only work if one modifies the default Android OS or if the framework comes pre-installed with every phone. This can prove a big challenge because the creators of various mobile platforms might not be willing to modify their OS or pre-install non-approved apps and frameworks with the default offering. Furthermore, it doesnt prevent the app developers from making bad choices during their development process anyway. MASF was also presented as only a prototype and not a functional implementation.

As evident, most of the aforementioned proposals focused only on certain security issues plaguing mHealth apps. These solutions are difficult to integrate with everyday apps as they expect the app developers to either significantly change their development process or dont run on default mobile OS platforms. Furthermore, most of these solutions are only implemented as a prototype. Currently, there exists no framework that can be easily incorporated into mHealth apps. Our work aims to address these limitations by providing a security framework for mHealth apps that addresses security and privacy issues by reducing the cybersecurity burden from app developers. Also, our goal here is to not build a prototype, but an actual framework, that is available for the app developers to start integrating into their software.

III. Design & Architecture

In this section, we present the design and high-level architecture of the proposed mHealth Security Framework (MSF). While we demonstrate the framework on Android OS, the proposed design is not restricted to Android OS. The high-level architecture and the layers can be easily adapted to iOS and other mobile platforms. Our proposed framework shown in Figure 1 forms a bridge between the mHealth application and Android Platform Application Programming Interface (API) for any workflows within the app that has a security footprint (example: storing or retrieving personally identifiable information and communicating to third party web servers). At the same time, our framework also allows the mobile application to directly interface with the Android Platform API, to ensure that app developers are not restricted in providing desired user experience for consumers. In essence, app developers can easily import the framework into their mHealth apps and integrate necessary security and privacy features provided by the library into their code. The framework

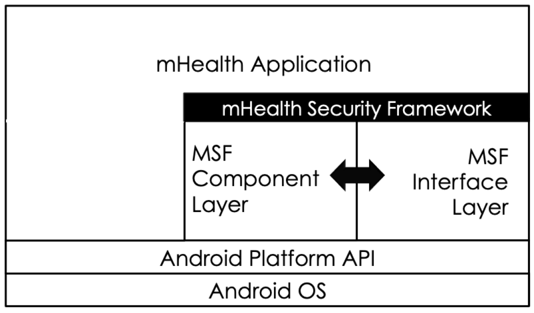


Fig. 1: Architecture of mHealth Security Framework (MSF)

design consists of four layers including Android OS, Android Platform API, and two layers namely MSF component layer (MSF-CL) and MSF interface layer (MSF-IL) that are built on top of the Android Platform API. We now present further details on each of these layers below:

A. Android Operating System

We chose Android platform for our framework, as it is the most popular operating system for smartphones. MSF built on Android requires no modifications during development, testing, release or during integration with mHealth apps. MSF uses the security and non-security functionalities that come natively with the default operating system. However, it is important to note that the framework doesnt interface directly with the Android OS. Instead, it uses the Android Platform API provided by the makers of Android.

B. Android Platform Application Programming Interface

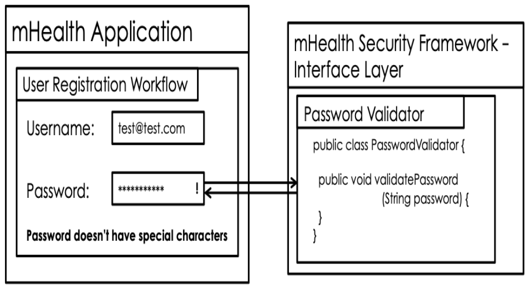
Android API is used by applications to interact with the underlying operating system. The proposed framework acts as a wrapper around the Android Platform API exposing functionalities common for mHealth apps with the benefit of ensuring that sound security engineering practices are adopted in their implementation.

C. MSF Component Layer

This layer consists of Android Components such as activities, fragments, layouts, views and intents specific to functionalities or workflows commonly used in mHealth apps. The framework implements sound security practices into existing Android components (see Section IV and V). It is important to note that the component layer relies on the frameworks interface layer for security functionalities such as authentication, authorization and encryption (Figure 2).

D. MSF Interface Layer

This layer is responsible for enforcing privacy and security standards in handling of medical data and various other workflows common in mHealth apps. It is built on top of the Android Platform API and uses it to interact with various hardware and software components of Android OS such as sensors, pin/password authenticator, external storage, key store and database (Figure 3). The interface layer ensures that appropriate security standards are enforced at all times by


mHealth Application
User Registration Workflow
Username:
test@test.com
Password:
****
!
Password doesn't have special characters
mHealth Security Framework - Interface Layer
Password Validator

```
public class PasswordValidator {

  public void validatePassword
            (String password) {
  }
}
```

Fig. 2: MSF CL using MSF IL for password validation

isolating the confidentiality, integrity and availability checks. For instance, protocols, algorithms and policy enforcement used in authentication, cryptography, authorization and other security and privacy principles are embedded in this layer. The static nature of these checks ensures that app developers cannot override fundamental security requirements or design security mechanisms that provide insufficient cryptography.

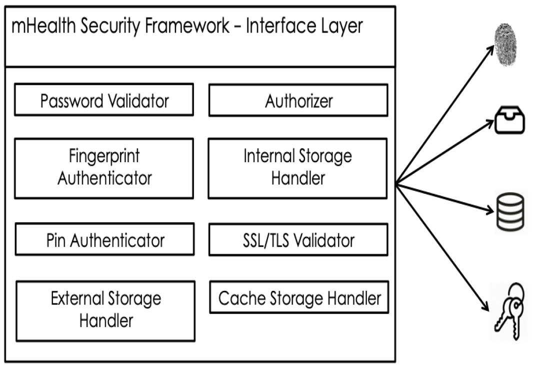


Fig. 3: MSF IL interacting with Android Platform internals.

IV. Features of mHealth Security Framework

In this section, we provide the list of features implemented in our security framework. The goal is to understand the workflows for mHealth apps and identify implementation flaws that lead to these security vulnerabilities. Table I provides a list of the various features and their assigned priority level. We assign priority levels to each feature to help decide features to include in the Minimum Viable Product (MVP) version of the framework.

A. Password Android component that automatically validates strength of password (F1)

This feature provides a user interface component for password authentication. It automatically checks strength of the password and enforces user to choose strong passwords.

B. Code flow to securely store/retrieve sensitive data to/from Androids internal/cache/external storage (F2/F3/F4)

This feature allows mHealth apps to store and retrieve sensitive data such as Personally Identifiable Information (PII) to and from the Android phones internal, cache or external storage. Irrespective of the type of storage option chosen, the

TABLE I: List of Features implemented in MSF

Feature	Feature Name	Cyber Security Aspect	Priority
F1	Password Android component that automatically validates strength of the password	Authentication	Must have
F2	Code flow to securely store/retrieve sensitive data to/from Androids internal storage	Storage	Should have
F3	Code flow to securely store/retrieve sensitive data to/from Androids internal cache	Storage	Should have
F4	Code flow to securely store/retrieve sensitive data to/from Androids external storage	Storage	Must have
F5	Code flow to securely store/retrieve sensitive data to/from Androids database	Storage	Should have
F6	Integration for Facebook or Gmail Login workflow	Authentication.	Could have
F7	Require fingerprint authentication when storing/retrieving sensitive data from Androids storage options	Authentication	Must have
F8	Code flow to prevent connection to 3rd party servers or APIs without SSL/TLS or with invalid SSL/TLS certificates	Communication.	Must have
F9	Role based Access Control	Authorization	Must have

framework triggers authentication via fingerprint or pin and encrypts the data before physically storing it on the phone.

C. Code flow to securely store/retrieve sensitive data to/from Androids database (F5)

This feature provides for storing sensitive data in databases, triggers authentication and encryption/decryption of sensitive data when storing/retrieving it from the database.

D. Integration for Facebook or Gmail Login workflow (F6)

This feature enables users of mHealth apps to register using their Gmail or Facebook Login credentials. The goal of the framework is to abstract the integration since interfacing with these services isn't easy and can prove to be a roadblock for mHealth app develoopers from providing sound authentication mechanisms.

E. Require fingerprint authentication when storing/retrieving sensitive data from Androids storage options (F7)

This feature enforces the user to perform fingerprint authentication if the user's phone has appropriate hardware installed to extract fingerprints. If not, the framework must default to using password or pin authentication mechanisms. This feature is triggered in all cases of storing, retrieving and transmission of sensitive data to/from the mHealth app.

F. Code flow to prevent connection to 3rd party servers or APIs without SSL/TLS or with invalid SSL/TLS certificates (F8)

This feature provides a mechanism for the mHealth app to connect to third party servers and APIs. This feature detects domains with no SSL/TLS certificates, domains with expired or self-signed certificates or domains with untrusted certificate authorities and ensures that connections to non SSL/TLS servers are rejected.

G. Role based Access Control (F9)

This feature enables mHealth app developers to perform authorization checks in the form of a role based access control system. This is to support various workflows common in mHealth apps such as determining if a given user (with the role of Patient) has privileges to perform a given action (Create) on a given resource (Prescriptions). The framework must also provide basic setup for the mHealth app to create/delete new users and new roles, assign those roles to various users and finally assign privileges to roles.

V. Results

In this section, we present the implementation results of the proposed mHealth security framework. We created a dummy mHealth app built for Android OS that acts as a simulation environment for real world Android mHealth apps. This app has the mHealth Security Frameworks (MSF) Android Archive (AAR) library file integrated into its codebase. It consists of workflows leveraging the various features exposed in the MVP version of MSF, thereby, allowing us to prove the effectiveness of the framework in mitigating security vulnerabilities reported in popular mHealth apps today. We used Android Developer Tools (ADT), part of the Android Studio Integrated Development Environment, to run simulations and perform various result validations. Specifically, ADT provides an Android phone emulator to run the dummy mHealth app. It also allowed us to select various Android phones and run various Android OS versions to fully test our framework and validate the results against expected outcomes.

A. Password Android component that automatically validates strength of password (F1)

The component appropriately raises errors when weak passwords are entered (Figure 4) in the input. The user is forced to meet the expected standards for passwords in order to proceed forward in the dummy app.

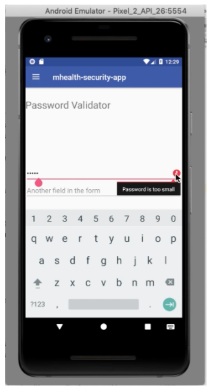
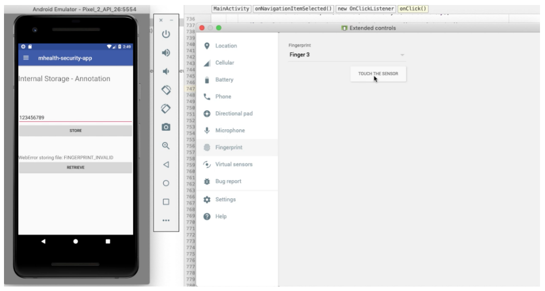

Android Emulator - Pixel_2_API_26:5554
mhealth-security-app
Password Validator
Another field in the form
Password is too small

Fig. 4: PasswordEditText component to auto validate password.

B. Code flow to securely store/retrieve sensitive data to/from Androids internal/cache/external storage (F2/F3/F4)

Figure 5 demonstrates an attempt made by an invalid user to use the app to store sensitive data. The framework triggered the authentication workflow which failed because of the user's invalid fingerprint. On the other hand, Figure 6 demonstrates that upon successful authentication, the data (123456789 in Figure 5) is encrypted using AES/CBC/PKCS7Padding algorithm and stored in one of the three aforementioned Android storage options. Figure 6 confirms that the data is stored in encrypted



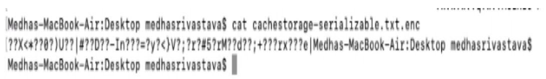
123456789

WebError storing file: FINGERPRINT_INVALID

Fingerprint: Finger 3

Fig. 5: Internal Storage workflow Store with Wrong Fingerprint.

format and cant be deciphered. Finally, the Retrieve also works like Store, however, after a successful authentication, the framework decrypts the encrypted data read from the file and then returns it back to the mHealth app to use it.

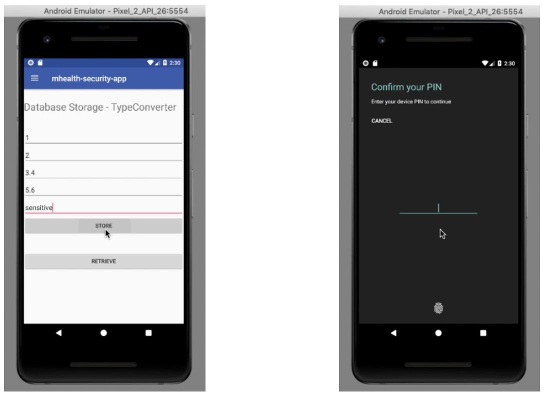


```
Medhas-MacBook-Air:Desktop medhasrivastava$ cat cachestorage-serializable.txt.enc
??X<*??0?)U??|#??D??-In???=?y?<}V?;?r?#5?rM??d??;+???rx???e|Medhas-MacBook-Air:Desktop medhasrivastava$
Medhas-MacBook-Air:Desktop medhasrivastava$
```

Fig. 6: Internal Cache workflow Encrypted data.

C. Code flow to securely store/retrieve sensitive data to/from Androids database (F5)

As shown in Figure 7, the user of the dummy mHealth app must pass PIN, Password or fingerprint authentication to store sensitive data of various data types such as integer (1 in figure), long (2), float (3.4), double (5.6) and string (sensitive) into the database. Figure 8 confirms that the framework encrypts each


Android Emulator - Pixel_2_API_26:5554
mhealth-security-app
Database Storage - TypeConverter
STORE
RETRIEVE
Android Emulator - Pixel_2_API_26:5554
Confirm your PIN
Enter your device PIN to continue
CANCEL

1
2
3.4
5.6
sensitive

Fig. 7: Database storage workflow Store.

of the sensitive fields before inserting/updating the database. Similarly, upon retrieval, the framework once again triggers the authentication process and if successful, decrypts those fields before passing it on to the mHealth app to use them.

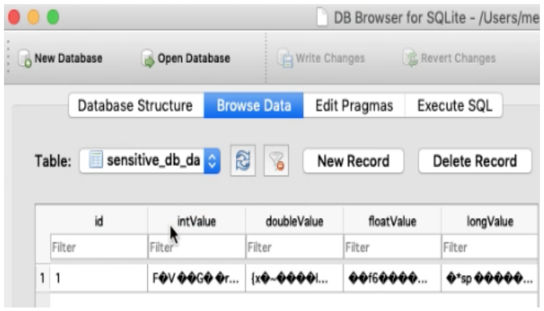


Table: sensitive_db_da

	id	intValue	doubleValue	floatValue	longValue
	Filter	Filter	Filter	Filter	Filter
1	1	F�V��G� �r...	{x�~����l...	��f6����...	�*sp �����...

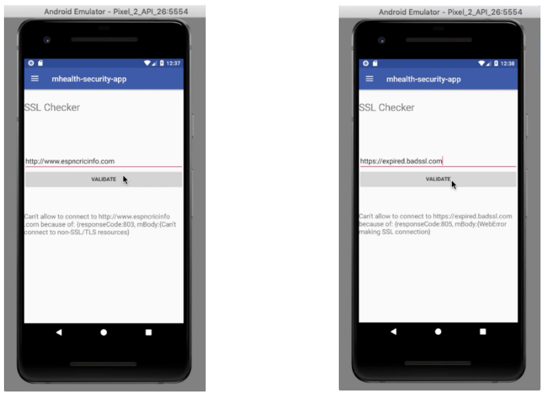
Fig. 8: Database storage workflow Encrypted Row.

D. Require fingerprint authentication when storing/retrieving sensitive data from Androids storage options (F7)

Implementation results of features F2, F3, F4 and F5 (current section) confirm that the dummy mHealth app forces the user to provide his/her fingerprint for authentication purposes. Also, the app fallsback to password or pin authentication if the phone doesn't have appropriate fingerprint hardware installed.

E. Code flow to prevent connection to 3rd party servers or APIs without SSL/TLS or with invalid SSL/TLS certificates (F8)

Figure 9 demonstrates the dummy mHealth app trying to connect to two 3rd party websites, one with non SSL/TLS and the other with an expired SSL certificate. In both cases, the framework correctly throws an unrecoverable exception with an appropriate reason for the failure. This can be trapped by the mHealth app developer to take an appropriate action in their app.



http://www.espncricinfo.com

Can't allow to connect to http://www.espncricinfo.com because of: {responseCode:803, mBody:{Can't connect to non-SSL/TLS resources}

https://expired.badssl.com

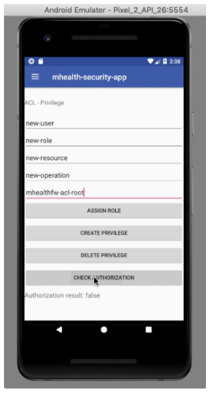
Can't allow to connect to https://expired.badssl.com because of: {responseCode:805, mBody:{WebError making SSL connection}

Fig. 9: SSL Checker connecting to various 3rd party urls and APIs.

F. Role based Access Control (F9)

Figure 10 demonstrates the dummy mHealth app performing authorization check for a sample workflow. The framework accurately fails the authorization check for a new-user to perform a new-operation on a new-resource because new-role that is currently assigned to new-user does not have the required privilege.

MSF is a fully functional framework that relies on app developers to understand how to use various features offered by the framework. For instance, if the app developer forgets



new-user
new-role
new-resource
new-operation
mhealthfw-acl-root

Authorization result: false

Fig. 10: Authorization Check.

to mark sensitive data as sensitive, then the framework wont trigger any of the security features such as authentication and encryption when storing/retrieving it.

VI. Conclusion

With the growing popularity of smartphones and mHealth apps, security and privacy are of paramount importance. In this work, we proposed a mHealth Security Framework (MSF) to mitigate security vulnerabilities that occur due to insecure practices followed by app developers. The proposed mHealth security framework does exceedingly well in addressing the security vulnerabilities that exist in mHealth apps today that are a direct result of flaws in app development processes. Our proposed framework is built on top of default mobile OS and doesn't require any modifications to mobile platforms.

The separation of concerns paradigm followed in the framework enables developers of those apps to iteratively integrate various features provided by the framework. Most importantly, it resolves popular security vulnerabilities found in mHealth apps today and also addresses the lack of security cognizance among mHealth app developers by hiding the cyber security aspects of common workflows present in current mHealth apps and ensuring that the best standards are followed in their implementation. Also, as validated by the dummy mHealth app, the framework adds little to no cost to the user experience factor of mHealth apps adopting it. Finally, making the frameworks codebase available to the research community enables iterative improvements through peer reviews and also enables the entire community to collectively keep the solution up to date with latest security patches and new mobile OS versions. Future work involves adapting this framework for other mobile platforms such as iOS.

References

[1] M. Abdulnabi, A. Al-Haiqi, M.L. Mat Kiah, A.A. Zaidan, B.B. Zaidan, and M. Hussain, A distributed framework for health information exchange using smartphone technologies, Journal of biomedical informatics, vol. 69, pp. 230250, 2017.

[2] D.D. Luxton, R.A. McCann, N.E. Bush, M.C. Mishkind, and G.M. Reger, mHealth for mental health: integrating smartphone technology in behavioral healthcare, Professional Psychology: Research and Practice, vol. 42, no. 6, p. 505, 2011.

[3] D.R. Bateman, B. Srinivas, T.W. Emmett, T.K. Schleyer, R.J. Holden, H.C. Hendrie, and C. Callahan, Categorizing health outcomes and efficacy of mHealth apps for persons with cognitive impairment: a systematic review, Journal of Medical Internet Research, vol. 19, no. 8, 2017.

[4] M. Hussain, A. Al-Haiqi, A.A. Zaidan, B.B. Zaidan, M. Kiah, S. Iqbal, S. Iqbal, M. Abdulnabi, A security framework for mHealth apps on Android platform, Computers & Security, vol. 75, pp. 191-217, 2018.

[5] B. Martnez-Perez, I. de la Torre-Dez, and M. Lopez-Coronado, Privacy and security in mobile health apps: A review and recommendations, Journal of Medical Systems, vol. 39, no. 1, pp. 18, 2014.

[6] R. Adhikari, D. Richards, and K. Scott, Security and Privacy Issues Related to the Use of Mobile Health Apps, Australasian Conference on Information Systems (ACIS), Auckland, New Zealand, 2014.

[7] T. Dehling, F. Gao, S. Schneider, and A. Sunyaev, Exploring the far side of mobile health: information security and privacy of mobile health apps on iOS and Android, JMIR mHealth Uhealth, vol. 3, no. 1, 2015.

[8] D. He, M. Naveed, C. Gunter, and K. Nahrstedt, Security Concerns in Android mHealth Apps, Annual Symposium proceedings. AMIA Symposium, 2014, 645654.

[9] M. Plachkinova, S. Andrs, and S. Chatterjee, A Taxonomy of mHealth AppsSecurity and Privacy Concerns, Hawaii International Conference on System Sciences, 2015.

[10] H.O. Alanazi, G.M. Alam, B.B. Zaidan, and A.A. Zaidan, Securing electronic medical records transmissions over unsecured communications: an overview for better medical governance, Journal of Medicinal Plants Research, vol. 4, no. 19, pp. 2059-2074, 2010.

[11] S. Gejibo, F. Mancini, K. A. Mughal, R. A. B. Valvik, and J. Klungsyr, Secure data storage for mobile data collection systems, in Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ser. MEDES 12. New York, NY, USA: ACM, 2012, pp. 131144.

[12] J. Müthing, T. Jäschke, and CM. Friedrich, Client-focused security assessment of mHealth apps and recommended practices to prevent or mitigate transport security issues, JMIR mHealth Uhealth, vol. 5, no. 10, 2017.

[13] C. Lakin, and G. Thamilarasu, A Security Framework for Mobile Health Application, 5th International Conference on Future Internet of Things and Cloud Workshops, 2017.

[14] F. Gonalves, J. Macedo, M. J. Nicolau, and A. Santos, Security architecture for mobile e-health applications in medication control, in 2013 21st International Conference on Software, Telecommunications and Computer Networks - (SoftCOM 2013), Sep. 2013, pp. 18.

[15] D. Sethia, D. Gupta, T. Mittal, U. Arora, and H. Saran, NFC based secure mobile healthcare system, in 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS), Jan. 2014, pp. 16.

[16] W. D. Yu, L. Davuluri, M. Radhakrishnan, and M. Runiassy, A Security Oriented Design (SOD) Framework for eHealth Systems, in Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International, Jul. 2014, pp. 122127.

[17] Federal Trade Commission. FTC Releases New Guidance for Developers of Mobile Health Apps; 2016. Available from: https://www.ftc.gov/news-events/press-releases/2016/04/ftc-releases-new-guidance-developers-mobile-health-apps.

[18] M. Ahmed and M. Ahamad, Protecting Health Information on Mobile Devices, in Proceedings of the Second ACM Conference on Data and Application Security and Privacy, ser. CODASPY 12. New York, NY, USA: ACM, 2012, pp. 229240.

[19] S. Mitchell, S. Ridley, C. Tharenos, U. Varshney, R. Vetter, and U. Yaylacicegi, Investigating Privacy and Security Challenges of mHealth Applications, Americas Conference on Information Systems (AMCIS), Chicago, Illinois, USA, 2013.

[20] A. Murad, B. Schooley, and Y. Abed, A Secure mHealth Application for EMS: Design and Implementation, in Proceedings of the 4th Conference on Wireless Health, ser. WH 13. New York, NY, USA: ACM, 2013, pp. 15:115:2.

[21] M. A. Simplicio, L. H. Iwaya, B. M. Barros, T. C. M. B. Carvalho, and M. Nslund, SecourHealth: A Delay-Tolerant Security Framework for Mobile Health Data Collection, IEEE Journal of Biomedical and Health Informatics, vol. 19, no. 2, pp. 761772, Mar. 2015.

[22] W. Liu and E. K. Park, "e-Healthcare Security Solution Framework," 2012 21st International Conference on Computer Communications and Networks (ICCCN), Munich, 2012, pp. 1-6.