Analysis of the Adherence of mHealth Applications to HIPAA Technical Safeguards

Bilash Saha

Analysis of the Adherence of mHealth Applications to HIPAA Technical Safeguards

A Thesis Presented To
The Faculty of Information Technology Department

by
Bilash Saha

In Partial Fulfillment of Requirements for the Degree
Master of Science in Information Technology

College of Computing and Software Engineering, Kennesaw State University

Summer 2023

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The proliferation of mobile health technology, or mHealth apps, has made it essential to protect individual health details. People now have easy access to digital platforms that allow them to save, share, and access their medical data and treatment information as well as easily monitor and manage health-related issues. It is crucial to make sure that protected health information (PHI) is effectively and securely transmitted, received, created, and maintained in accordance with the rules outlined by the Health Insurance Portability and Accountability Act (HIPAA), as the use of mHealth apps increases. Unfortunately, many mobile app developers, particularly those of mHealth apps, do not completely understand the HIPAA security and privacy requirements. This offers a unique opportunity for research to create an analytical framework that can help programmers maintain safe and HIPAA-compliant source code while also educating users about the security and privacy of private health information. The plan is to develop a framework which will serve as the foundation for developing an integrated development environment (IDE) plugin for mHealth app developers and a web-based interface for mHealth app consumers. This will help developers identify and address HIPAA compliance issues during the development process and provide consumers with a tool to evaluate the privacy and security of mHealth apps before downloading and using them. The goal is to encourage the development of secure and compliant mHealth apps that safeguard personal health information.

Keywords: *HIPAA, mHealth, Android Apps, Privacy & Security, IDE plugin*

CHAPTER 1
INTRODUCTION

Due to the fact that mHealth systems acquire, process, store, and transport sensitive user data as well as individual health records, the situation with regard to security vulnerabilities is particularly crucial. A study conducted in the United States has found that smartphones are becoming more common in medical environments. More than half of physicians in the US encourage their patients to use medical applications on their smartphones, with over 58% of those polled reporting that they have downloaded a mHealth app [32]. The use of medical smartphone apps has increased during emergencies such as COVID-19 lockdowns [24, 35]. Healthcare practitioners are also using these apps more frequently [34]. Patients use these apps for a variety of reasons including paying bills, scheduling appointments, sending messages to providers, accessing lab results, and viewing prescriptions and medical records [36, 40].

The mHealth community has noted the increasing demand from patients for data accessibility and app data sharing, which has raised concerns about the security and privacy of patient health information [45]. To address this issue, the Health Insurance Portability and Accountability Act (HIPAA) was enacted in the US in 1996 [46]. According to HIPAA regulations, business associates providing mHealth apps on behalf of a covered entity must implement administrative, technological, and physical safeguards to protect electronically protected health information (EPHI) [42–44]. Therefore, it is critical to ensure that patients transmit their Protected Health Information (PHI) to apps securely to comply with HIPAA requirements and maintain data security in the mHealth ecosystem [45]. According to a survey [47], 84% of FDA-approved medical health applications had security flaws that could expose sensitive data or harm the device.

Previously, healthcare data breaches resulted in changes to a patient's medical history, posing potential health risks [22, 48, 49]. To ensure patient data security and privacy, mHealth apps must ensure secure data transmission between external sensors, third-party APIs, the provider's server, the cloud environment, and the mobile app, as well as protect data during processing and provide transparency in data collection, storage, and transfer protocols [50]. With over 300,000 mHealth apps available and approximately 25% of healthcare providers experiencing data breaches that violate HIPAA policies, it is critical to perform a HIPAA Technical Safeguards assessment before using any unsecured mHealth apps [51]. The COVID-19 pandemic has significantly increased the use of technology infrastructure, resulting in an increase in the market for unsecured mHealth applications, emphasizing the importance of assessing and ensuring HIPAA compliance to protect patient data [51].

According to a recent US Government inter-agency assessment, there have been an average of 4,000 daily ransomware attacks since early 2016, representing a 300% increase over the 1,000 daily ransomware attacks reported in 2015 [30]. Healthcare institutions are particularly appealing targets for cyber attackers because they contain vast amounts of valuable data, such as EHR systems, EMR systems, billing systems, practice management systems, computerized physician order entry systems, and thousands of IoT devices [30]. Cyberattacks on healthcare networks have increased during the COVID-19 pandemic, putting patient care and private data at risk and potentially violating HIPAA's Privacy and Security Rules [52]. A ransomware attack on Dusseldorf University Hospital, for example, resulted in patient transfers and one death in 2020 [52]. Similarly, in September 2020, the Ryuk ransomware attack on Universal Healthcare Services disrupted EHR at all 400 sites for approximately three weeks, costing $67 million in lost revenue and recovery [53].Several healthcare institutions, including Brookside ENT and Hearing Center of Creek in Michigan, Wood Ranch Medical in Simi Valley in California, DCH Health Systems in Alabama, Rouen University Hospital-Charles Nicolle in northern France, Cancer Center of Hawaii, and Hackensack Meridian Health in New Jersey, were victims of ransomware attacks in 2019 [54, 26, 27].

The HIPAA Security Rule requires covered entities and business associates to have security incident procedures, response, and reporting processes in place, such as ransomware detection and analysis, eradication of ransomware instances, mitigation or remediation of vulnerabilities, data backups and recovery plans, contingency plans, and so on, to help them respond to and recover from a ransomware attack [29]. Cybersecurity threats, such as malware and ransomware, can cause significant damage to computer systems, data centers, web and mobile applications in a variety of industries and businesses [33, 38]. Because traditional anti-ransomware solutions are inadequate to combat advanced and sophisticated attacks, new cutting-edge methodologies, such as conventional and neural network-based architectures, can be used to develop advanced ransomware solutions [23, 28, 31, 37, 41].Previous research has primarily focused on mobile device security rather than thoroughly investigating how apps securely store or transfer data, especially when used by remote healthcare providers or users. HIPAA security and privacy regulations are unfamiliar to many mHealth app developers. This presents an opportunity to develop static and dynamic code analysis algorithms and techniques to assist mHealth app developers in ensuring their products comply with HIPAA security and privacy guidelines. There is currently no available analysis framework for assessing mHealth app security and privacy risks in accordance with applicable HIPAA technical security and privacy guidelines.

To address this issue, we propose the "HIPAAChecker" HIPAA Technical Safeguard assessment framework, which employs various analysis algorithms and techniques to statically and dynamically analyze mHealth source code (Android). Furthermore, we conducted a comparative study of 285 mHealth apps obtained from the Google Play store in the Medical and Health & Fitness categories. Qualitative evaluation methods were used to examine the results of the HIPAA report generated by the investigated apps and identify app vulnerabilities. Our findings demonstrate that the HIPAAChecker framework is extremely effective in addressing potential data breaches through mHealth apps and provides valuable assistance to the developer community[39].

The aim of this research is to promote the development of secure and compliant mHealth apps that protect the confidentiality of personal health information. So the main objectives are as follows:

- Develop a source code analysis framework to evaluate HIPAA Technical Safeguards for determining the compliance of mHealth applications.
- Incorporate API-level checking in accordance with secure data communication between third-party mHealth apps and electronic health record systems.
- Implementation of meta-analysis to identify potential risks and safety features, and in detecting HIPAA violations.
- Develop an IDE plugin tool that provides developers with analysis and feedback on their codebase to address potential security and privacy issues early in the development process.

CHAPTER 2
BACKGROUND AND LITERATURE WORK

2.1 Background:

Administrative, Physical, and Technical security requirements are outlined in the Health Insurance Portability and Accountability Act (HIPAA). Administrative safeguards are rules and guidelines that control the choice, creation, application, and upkeep of security measures.

Reference	**Rule Name**	**Technical Safeguards**
164.312(a)(1)	Authorization	Implement technological policies and procedures to restrict access to individuals or software programs that have been given access privileges for electronic information systems that maintain EPHI.
164.312(a)(2)(i)	Unique Id	Assign a unique name or number to each patient in order to identify and monitor their identification.
164.312(a)(2)(ii)	Emergency EPHI Access	Create and use processes for acquiring required digitally protected health information in an emergency.
164.312(a)(2)(iii)	Automatic Session timeout	Implement software procedures that end a session after a certain period of inactivity.
164.312(a)(2)(iv)	EPHI Encryption and Decryption	Implement a system for encrypting and decrypting EPHI.
164.312(b)	EPHI Audit Control	Implement methods for recording and examining activities in information systems that use or include EPHI.
164.312(c)(1)	EPHI Data Integrity	Implement regulations and procedures to prevent unauthorized manipulation or destruction of EPHI.
164.312(c)(2)	EPHI Integrity Verification	Utilize technological tools to verify that electronically stored protected health information has not been tampered with or deleted without authorization.
164.312(d)	EPHI Authentication	Establish processes to confirm that the individual or organization requesting access to EPHI is who is being identified.
164.312(e)(1)	EPHI Transmission Security	Implement technological security measures to prevent unauthorized access to digitally protected health information that is being sent through a network of electronic communications.
164.312(e)(2)(i)	EPHI Transmission Integrity	Implement security measures to guarantee that electronically transmitted protected health information is not improperly altered up to disposal without being noticed.
164.312(e)(2)(ii)	Appropriate EPHI Encryption	Implement a mechanism to encrypt EPHI whenever deemed appropriate.

Table 1: HIPAA Rules of Technical Safeguards

Physical safeguards are techniques, guidelines, and practices created to protect equipment from environmental and natural threats as well as unlawful access. Technical safeguards, on the other hand, refer to the policy- and technology-related procedures that defend against unauthorized access to electronically protected health information (EPHI) [16], [17], [18], [19]. Proposed source code analysis approaches for mHealth apps specifically address problems with Technical Safeguards (Table I) [19] [20]. It will be possible to meet other administrative and operational safeguards, such as providing tools and applications to review and monitor administrative security features, and prevent negative incidents, such as non-compliance with physical safeguards, by ensuring technical safeguard compliance. For instance, obtaining encrypted PHI data from a lost or stolen cell phone's mHealth app would be quite challenging.

2.2 Related Work:

In order to guarantee the confidentiality, integrity, and accessibility of electronic health information that is kept or transferred electronically, the HIPAA [1] security rule, which went into effect in April 2005, imposes administrative, physical, and technical measures [2], [3]. To safeguard patient and healthcare professional data, mHealth apps must be secured. According to a recent study in this area, security threats for mobile health applications might be divided into three categories: high (apps for monitoring, diagnosis, and care), medium (calculators, localizers, and alarms), and low (informative and educational apps) [4]. The American Health Information Management Association (AHIMA) provided advice [5] on how to deal with mobile health data breaches, including reviewing privacy settings on both apps and mobile devices, looking for certification signs, using password and encryption, and refraining from texting others with private or sensitive health information. To lessen security and privacy issues, the majority of vulnerabilities in the mobile health app should be addressed and resolved. Before permitting the usage of the mHealth applications, such initiatives need help to evaluate the source code and test them in accordance with the HIPAA's subsequent security and privacy criteria. According to [6], the privacy and security of personal health records are key issues. The absence of standardized mHealth applications and security concerns are a major impediment to their broad deployment. A comparative study [5] [7] of the top 20 mHealth applications found that while just two apps needed user authentication before logging in, 65% of the apps asked users to submit personal information such as name, address, email, and DOB. Data breaches and the privacy of users' personal information are seriously threatened by the 50% of applications that store data on the cloud. Additionally, more than 65% of applications shared user data with advertising or third parties without getting user permission, which is against the law. Only 20% of applications provided users with information regarding data privacy and security measures. Authors provided a static security analysis method using the free and open-source FindSecurityBugs [8] IDE plugin for Android Studio. They showed how integrating the plugin helps developers to safeguard mobile applications and lessen security threats as they are being implemented. According to a thorough review of the literature and Internet searches done as of March 2023 [9], [10], [11], [12], [13] [14], there aren't any tools or frameworks that verify the security of mHealth apps using the HIPAA security standards for EPHI. With the use of supplementary code analysis tools like FindBugs [8], IntelliJ IDE [15], and Eclipse IDE [9], developers may maintain and tidy up their code. These tools are designed to find possible flaws like inconsistencies, aid in improving the structure of the code, conform the source code to standards, and offer rapid fixes. Their primary responsibility is not to check security risks based on HIPAA technical security criteria [1]. A complete list of Android app analysis tools is provided in a study [11], however none of them concentrate on mHealth app security and privacy analysis in accordance with HIPAA technical security and privacy standards. Recent examples of mobile security analysis tools that do not concentrate on HIPAA violations are DexGuard [10] and TrustKit [12].

CHAPTER 3
HIPAACHECKER FRAMEWORK

In this study, a methodology for examining Android mHealth apps that are intended to accept patient data as input and follow HIPAA technical security criteria for data storage and transmission is proposed. The goal of this framework is to automatically evaluate application source code and discover security and privacy patterns prevalent in mHealth apps, in contrast to certain existing analysis tools that concentrate on Java-specific security checks. The framework's overall architecture is depicted in Figure 1, and its properties are contrasted with those of other tools already in use in Figure 2. Before publishing apps through Android Studio to the market, the plugin tool enables mHealth developers to find and fix flaws that can affect HIPAA technical security and privacy requirements. Additionally, by submitting APK files, users of common mHealth applications can use the tool to look for security flaws. The meta analysis flow in relation to regular web users and developers is depicted in Figure 3.

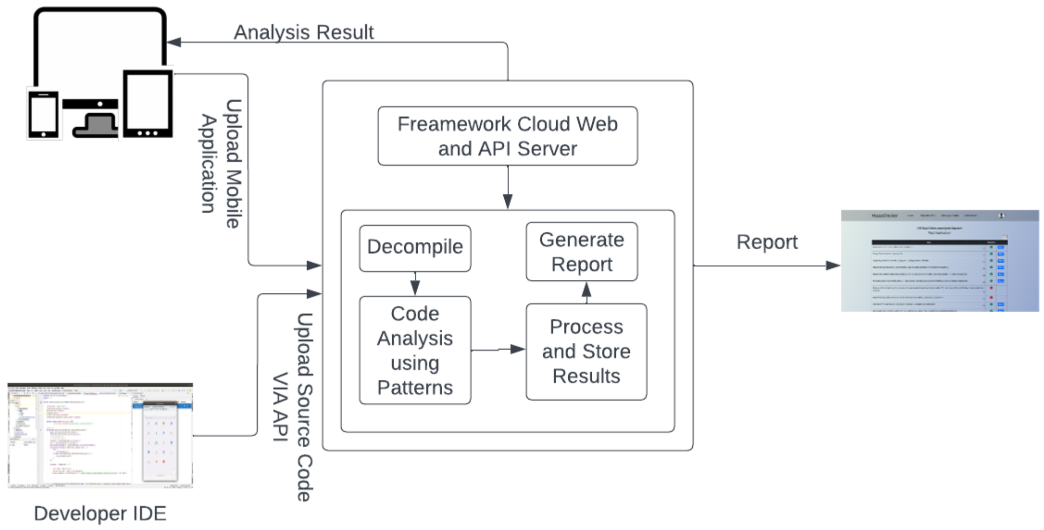

Analysis Result
Upload Mobile Application
Freamework Cloud Web and API Server
Decompile
Generate Report
Code Analysis using Patterns
Process and Store Results
Report
Upload Source Code VIA API
Developer IDE

Figure 1: HIPAAchecker Framework Architecture

The framework uses JADX [21] to decode manifest and other resources and decompile the dex file into Java classes because Android applications are often developed in Java, compiled into dex format files, and executed in instances of the Android virtual machine. A source code analyzer employing patterns illustrated at Table 2 is built and deployed into the framework cloud server machine. It looks for certain HIPAA-related code-level vulnerabilities and processes the results to produce reports for the targeted users. The report details the precise lines of source code that need to be refactored to make the application HIPAA compliant.

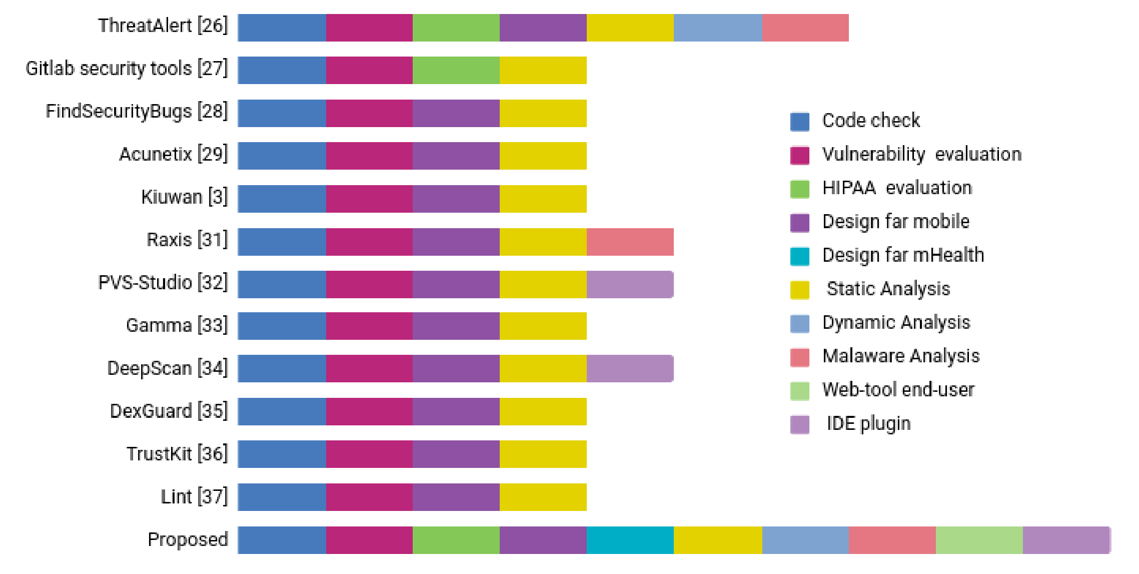


Figure 2: Comparison of the features of the proposed framework with those of similar products on the market

Overall, the proposed framework [55] will give mHealth app developers a methodical way to check that their applications meet HIPAA's technical security criteria, thereby improving the privacy and security of patient data.

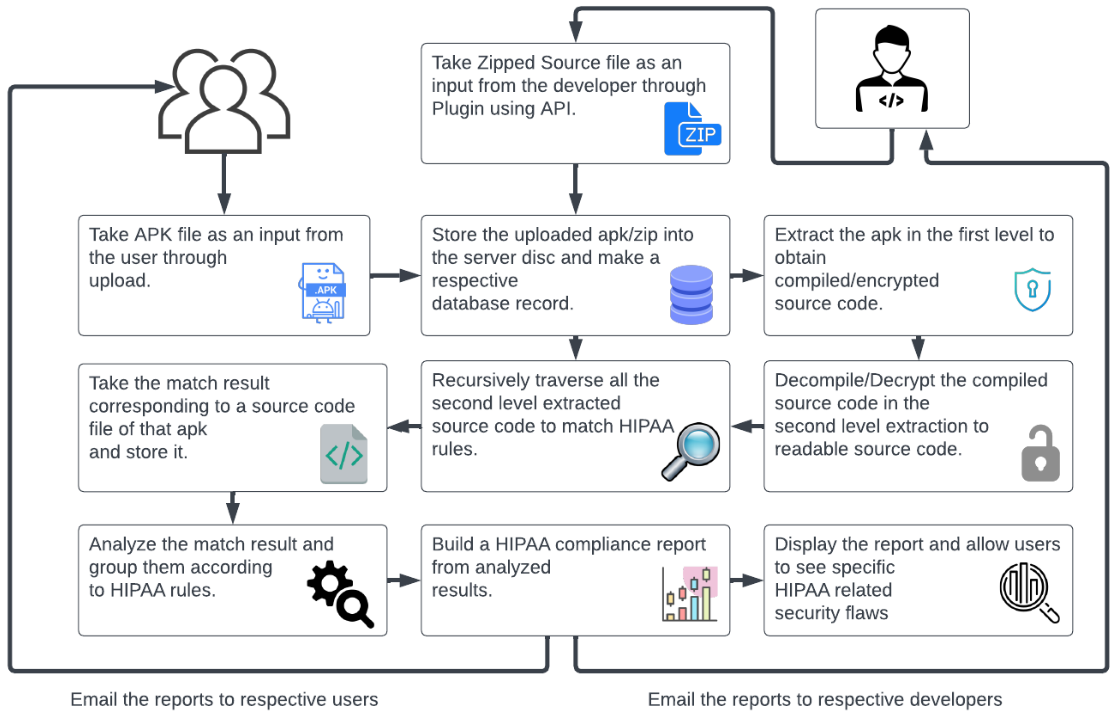

Take Zipped Source file as an input from the developer through Plugin using API.
ZIP
Take APK file as an input from the user through upload.
.APK
Store the uploaded apk/zip into the server disc and make a respective database record.
Extract the apk in the first level to obtain compiled/encrypted source code.
Take the match result corresponding to a source code file of that apk and store it.
Recursively traverse all the second level extracted source code to match HIPAA rules.
Decompile/Decrypt the compiled source code in the second level extraction to readable source code.
Analyze the match result and group them according to HIPAA rules.
Build a HIPAA compliance report from analyzed results.
Display the report and allow users to see specific HIPAA related security flaws
Email the reports to respective users
Email the reports to respective developers

Figure 3: HIPAA Analysis Flow

Rule ID	Sub Rule ID	Detection Code Patterns
EPHI_encryption_decryption	EN-DE	➢ import java.util Base64
	AES	➢ import org.springframework.security.crypto ➢ import java.security.Security ➢ Cipher.getInstance("AES/ECB/ ➢ Cipher.getInstance("AES") ➢ Cipher.getInstance(AES_MODE ➢ new SecretKeySpec(keyBytes, "AES" ➢ Cipher.getInstance("AES/CBC/
	DES	➢ Cipher.getInstance(.*DES ➢ Cipher.getInstance(.*des
	RSA	➢ Cipher.getInstance("RSA
	BLOWFISH	➢ .getInstance(.*BLOWFISH
	RC	➢ .getInstance(.*RC2 ➢ .getInstance(.*rc4 ➢ .getInstance(.*RC4, .getInstance(.*rc2

	Message Digest	➢ MessageDigest ➢ import java.security.MessageDigest ➢ .getInstance(.*MD5 ➢ .getInstance(.*md5 ➢ DigestUtils.md5(➢ import org.apache.commons.codec.digest.DigestUtils;
	SHA	➢ .getInstance(.*SHA-1 ➢ .getInstance(.*SHA1 ➢ DigestUtils.sha(
	ECB	➢ Cipher.getInstance(\s*"\s*AES\/ECB
	HMAC	➢ import org.apache.commons.codec.digest.HmacAlgorithms import org.apache.commons.codec.digest.HmacUtils
EPHI_Transmission_integrity	TRANS-NET	➢ javax.net.ssl.TrustManager ➢ TrustManagerFactory.getInstance(
Appropriate_ EPHI_Encryption	DE	➢ android.util.Base64 ➢ .decodeToString ➢ .decode
	EN	➢ android.util.Base64 ➢ .encodeToString, .encode
	ENCRYPT	➢ io.realm.Realm ➢ .encryptionKey(
	Chiper	➢ Net.sqlcipher. ➢ AS encrypted KEY
Authorization	Authorization Control	➢ AuthorizationException
	Access Control	➢ IllegalAccessException
EPHI_Transmission_Security	API	➢ addRequestProperty(\"Authorization
	PKIX	➢ PKIXRevocationChecker
	TRANS-Data	➢ HttpsURLConnection new
Unique_Id	PK	➢ PRIMARY KEY
EPHI_authentication	FireBaseAuth	➢ FirebaseUser ➢ sendFirebasePropertyRegisteredUser ➢ FirebaseUserPropertiesSender ➢ Com.google.firebase\:firebase-auth ➢ FirebaseAuth
	aAuth	➢ android.accounts.AccountManager ➢ AccountManager.get(, .currentUser
Automatic_Session_Timeout	Inactivity	➢ public void onUserInteraction() ➢ .reset()

		➢ .clear() ➢ .commit()
EPHI_Audit_Control	Audit	➢ AppOpsManager.OnOpNotedCallback
EPHI_integrity_verification	authorization_exception_on_destroy	➢ AuthorizationException
	illegal_destruction_restriction	➢ IllegalAccessException
EPHI_data_integrity	authorization_exception	➢ AuthorizationException
	illegal_access	➢ IllegalAccessException
	user_authentication_oauth	➢ android.accounts.AccountManager ➢ AccountManager.get(
	user_authentication_firebase	➢ FirebaseUser ➢ sendFirebasePropertyRegisteredUser ➢ FirebaseUserPropertiesSender Com.google.firebase\:firebase-auth ➢ FirebaseAuth

Table 2: HIPAA rules based meta-analysis techniques

Formal language is utilized to describe the vulnerability pattern that corresponds to HIPAA requirements instead of natural language, and the HIPAA technical safeguard patterns are formally described as Equation 1. The complete list of pattern-based matching for meta-analysis is described in Table 3.

$H = (r, d, s, v, p)$

Here,

"r" represents the rule reference

"d" represents the detection process

"s" represents sub rules of HIPAA Technical Safeguard

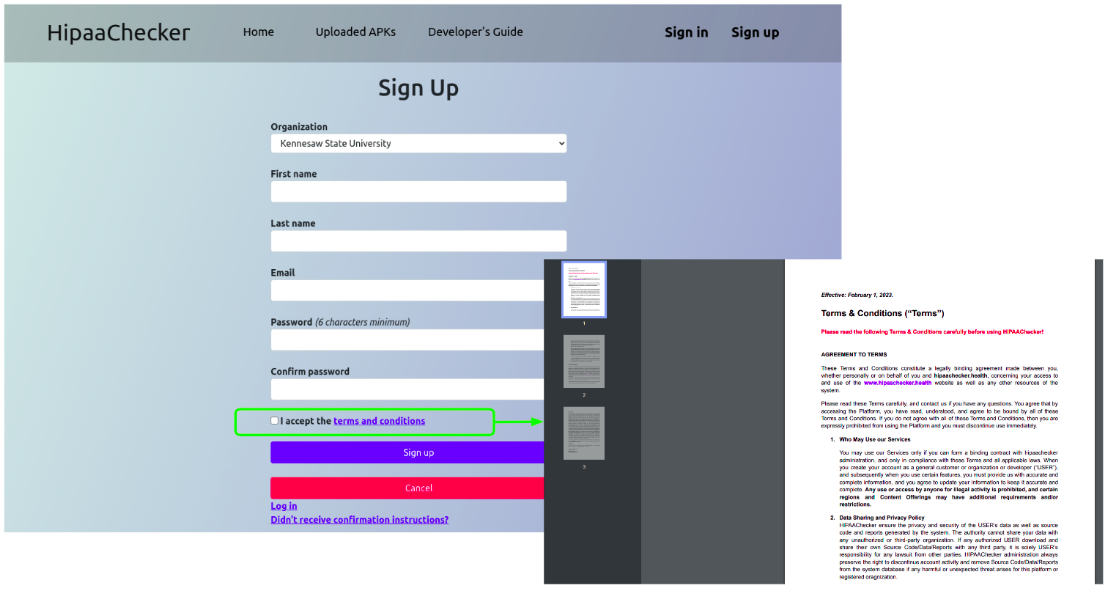
"v" represents vulnerability information or vulnerability evidence

"p" represents the patterns of HIPAA compliance.

CHAPTER 4
FRAMEWORK DEVELOPMENT

We developed a web-based platform for conducting HIPAA compliance experiments. We used agile methodology to create the HIPAAChecker application, which allowed us to create it quickly and with the most up-to-date technologies. We carefully selected the technologies we used to ensure that the web application and plugin were dependable, efficient, and secure. We chose Ruby as the backend programming language for the web application and Ruby on Rails as the web application framework. Ruby on Rails is well-known for its ability to speed up the development process and enable developers to write clean, maintainable code. For the cloud server, we also used Amazon Web Services (AWS) Elastic Compute Cloud (EC2), which provides scalable and reliable cloud computing resources. We used PostgreSQL, a powerful and dependable open-source database management system, for database management. To manage changes to the source code, the popular version control system Git was used.

We used HTTPS and two-factor authentication (2FA) for additional security to ensure secure communication. While 2FA adds an extra layer of security to user authentication by requiring a second form of authentication in addition to the password, HTTPS encrypts data in transit to prevent unauthorized access. User authentication and authorization were handled by Devise, a versatile and adaptable authentication solution. To ensure the application's quality, we used RSpec and Capybara for unit and integration testing, which allowed us to find and address problems early in the development process. The application was automated for deployment using Capistrano and Docker, which offered a reliable and effective way to do so with zero downtime.On the front end of the web application, we used Bootstrap for responsive design, creating a user-friendly and aesthetically pleasing interface. With the aid of Webpack, we were able to effectively manage and bundle our JavaScript and CSS files. ActionCable was used for real-time websockets, enabling the web application to react to user actions instantly. Additionally, we developed RESTful APIs with token-based authentication and API authentication using JSON Web Tokens (JWT) for third-party integration. We used Jbuilder in the API to efficiently serialize data so that it could be used in the API. Our team integrated the plugin with Android Studio using the IntelliJ Platform SDK. Git was also used for version control, and RESTful APIs were used to communicate with the plugin while JSON Web Tokens (JWT) were used for authentication and authorization. We tested the plugin in various scenarios using an Android emulator or a physical Android device, and we used OkHttp to make HTTP requests.



Effective: February 1, 2023.

Terms & Conditions ("Terms")

Please read the following Terms & Conditions carefully before using HIPAAChecker!

AGREEMENT TO TERMS

These Terms and Conditions constitute a legally binding agreement made between you, whether personally or on behalf of you and **hipaachecker.health**, concerning your access to and use of the **www.hipaachecker.health** website as well as any other resources of the system.

Please read these Terms carefully, and contact us if you have any questions. You agree that by accessing the Platform, you have read, understood, and agree to be bound by all of these Terms and Conditions. If you do not agree with all of these Terms and Conditions, then you are expressly prohibited from using the Platform and you must discontinue use immediately.

1. **Who May Use our Services**

   You may use our Services only if you can form a binding contract with hipaachecker administration, and only in compliance with these Terms and all applicable laws. When you create your account as a general customer or organization or developer ("USER"), and subsequently when you use certain features, you must provide us with accurate and complete information, and you agree to update your information to keep it accurate and complete. **Any use or access by anyone for illegal activity is prohibited, and certain regions and Content Offerings may have additional requirements and/or restrictions.**

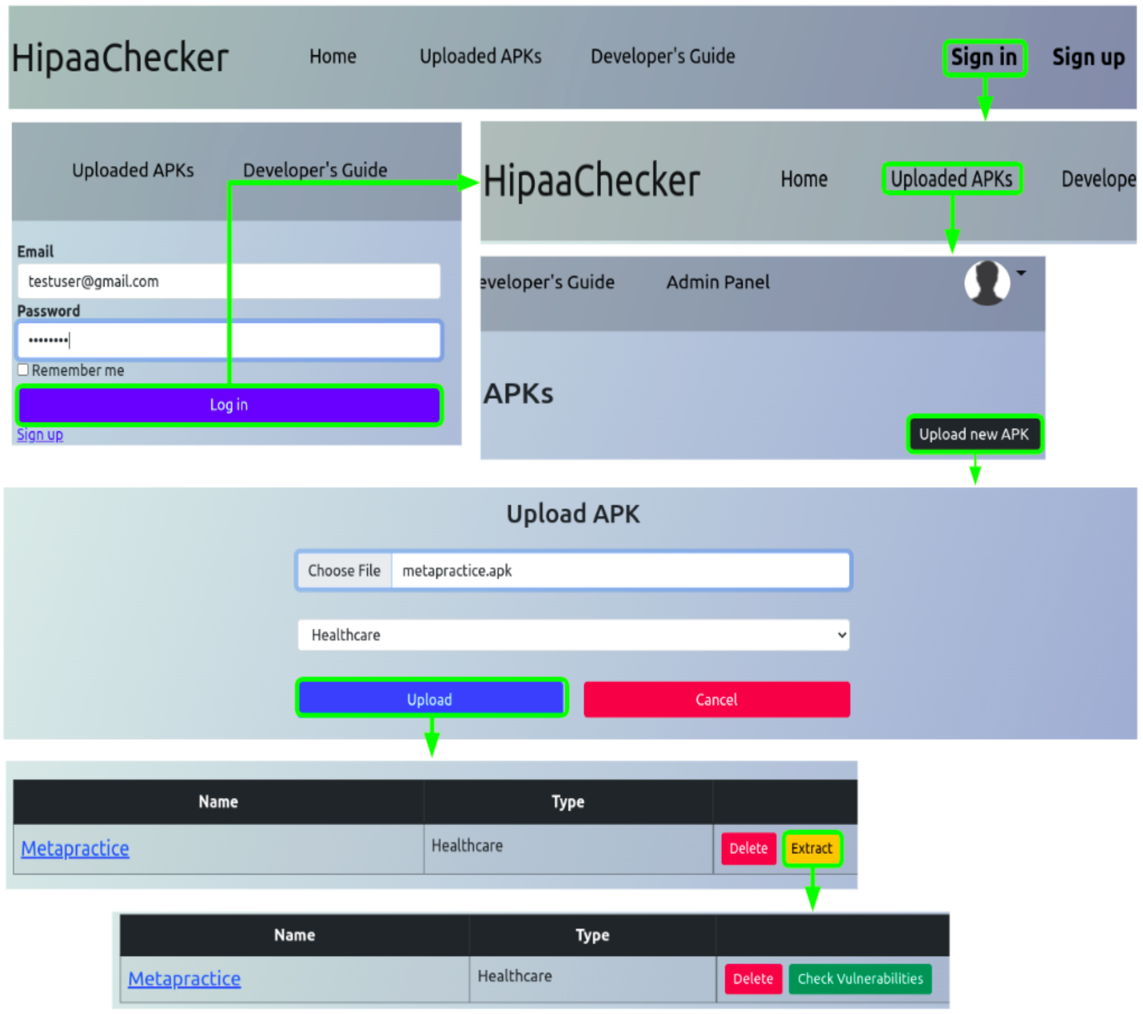
2. **Data Sharing and Privacy Policy**
   HIPAAChecker ensure the privacy and security of the USER's data as well as source code and reports generated by the system. The authority cannot share your data with any unauthorized or third-party organization. If any authorized USER download and share their own Source Code/Data/Reports with any third party, it is solely USER's responsibility for any lawsuit from other parties. HIPAAChecker administration always preserve the right to discontinue account activity and remove Source Code/Data/Reports from the system database if any harmful or unexpected threat arises for this platform or registered oragnization.

Figure 4: User Acceptance of terms and conditions during Sign Up

As part of the development process, we created comprehensive test cases for the HIPAAChecker application and used automation testing to ensure the application's quality and reliability. We were able to perform repetitive tests more efficiently and accurately thanks to automation testing, which reduced the time and effort required for testing. We were able to identify and fix issues early in the development process by developing test cases and utilizing automation testing, ensuring that the application met the high standards required for healthcare applications. We were also able to test the application in different scenarios, ensuring that it performed as expected under various conditions, thanks to the use of automation testing.

Our platform allows users to verify HIPAA compliance via APK file URLs and generate reports. The platform features two layers of access control to ensure security: Layer 1 involves username and password authentication, and Layer 2 involves user account and authentication security using two-factor authentication (2FA). In order to ensure the security and privacy of sensitive codebase being analyzed on our web-based platform, we have developed terms and conditions that users must agree to before using our service. Figure 4 shows the sign up process and before Sign up all users must agree to these developed terms and conditions.

As shown in Figure 5 and 6, a user may sign up for our online application, and when their email has been verified, they can log in. Once a user has successfully uploaded an APK file to our online application, they will have the option to extract the source code of the application using our "Extract" button. The extraction process will occur in two steps: first, the program will extract the compiled source code, and second, it will extract the readable source code. This allows us to analyze the code in depth and check for any HIPAA compliance violations.



Email
testuser@gmail.com
Password
••••••••

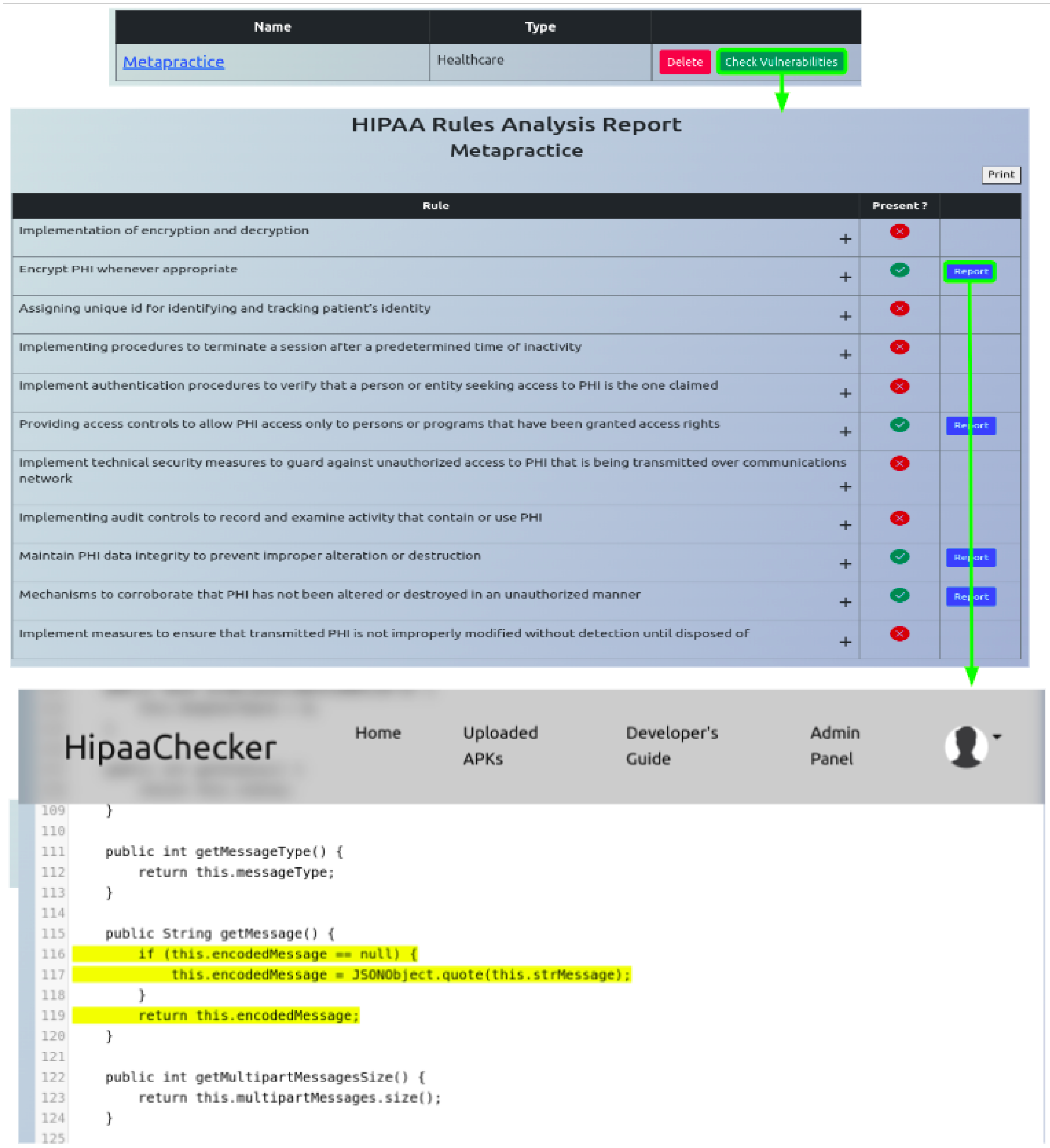
metapractice.apk
Healthcare

Name	Type	
Metapractice	Healthcare	Delete Extract

Name	Type	
Metapractice	Healthcare	Delete Check Vulnerabilities

Figure 5: Developed Web application Flow for login and and uploading APK

After the extraction process is complete, a new button with the caption "Check Vulnerabilities" will appear. When users click this button, our system will recursively search through the retrieved source code to check for any patterns that match those shown in Table 2. The table contains a list of patterns that are indicative of HIPAA compliance violations. If a match is found, the match findings will be stored in the database for further analysis.



Name	Type	
Metapractice	Healthcare	Delete Check Vulnerabilities

HIPAA Rules Analysis Report

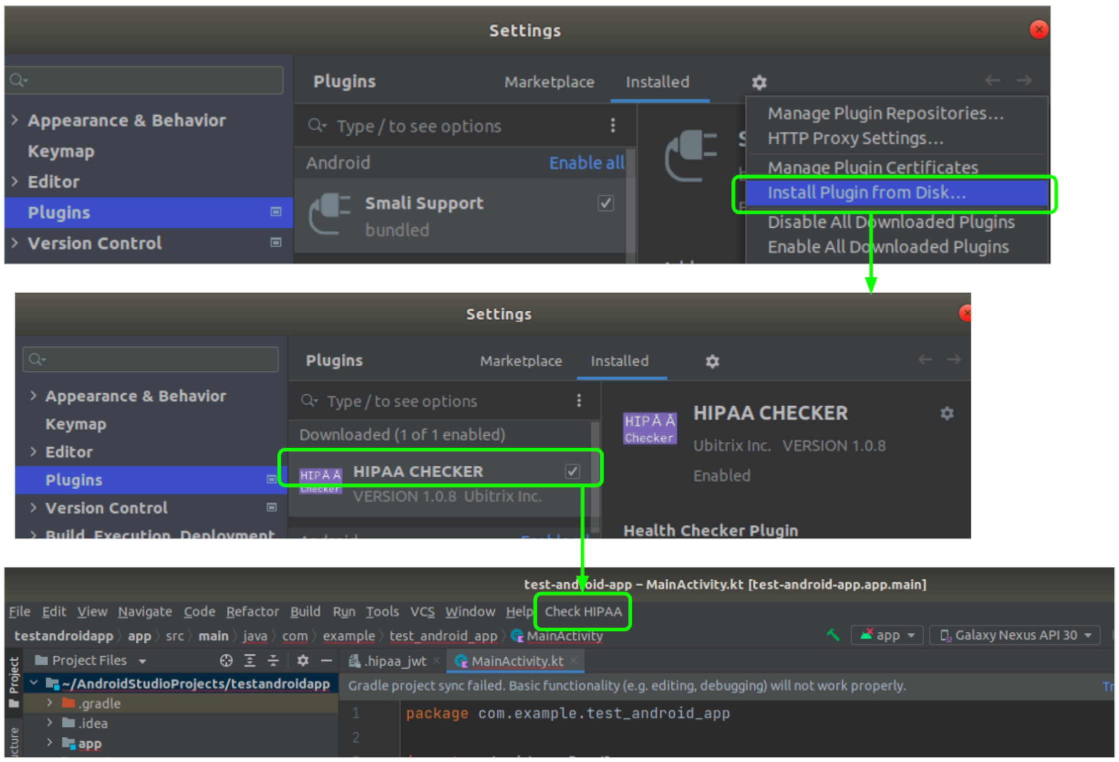
Metapractice

Rule	Present ?	
Implementation of encryption and decryption +	✗	
Encrypt PHI whenever appropriate +	✓	Report
Assigning unique id for identifying and tracking patient's identity +	✗	
Implementing procedures to terminate a session after a predetermined time of inactivity +	✗	
Implement authentication procedures to verify that a person or entity seeking access to PHI is the one claimed +	✗	
Providing access controls to allow PHI access only to persons or programs that have been granted access rights +	✓	Report
Implement technical security measures to guard against unauthorized access to PHI that is being transmitted over communications network +	✗	
Implementing audit controls to record and examine activity that contain or use PHI +	✗	
Maintain PHI data integrity to prevent improper alteration or destruction +	✓	Report
Mechanisms to corroborate that PHI has not been altered or destroyed in an unauthorized manner +	✓	Report
Implement measures to ensure that transmitted PHI is not improperly modified without detection until disposed of +	✗	

```
    }

    public int getMessageType() {
        return this.messageType;
    }

    public String getMessage() {
        if (this.encodedMessage == null) {
            this.encodedMessage = JSONObject.quote(this.strMessage);
        }
        return this.encodedMessage;
    }

    public int getMultipartMessagesSize() {
        return this.multipartMessages.size();
    }
```

Figure 6: Developed Web application Flow for checking HIPAA compliance

Once the traversal is complete and all of the match results have been combined, the user will be presented with a report. The report will include a legible high label that indicates whether the uploaded application complies with relevant HIPAA requirements or not. If the requirement is satisfied, the report will display a green check mark. If not, the report will display a cross.


Settings
Plugins
Marketplace
Installed
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Type / to see options
Android
Enable all
Smali Support
bundled
Manage Plugin Repositories…
HTTP Proxy Settings…
Manage Plugin Certificates
Install Plugin from Disk…
Disable All Downloaded Plugins
Enable All Downloaded Plugins
Settings
Plugins
Marketplace
Installed
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Type / to see options
Downloaded (1 of 1 enabled)
HIPAA CHECKER
VERSION 1.0.8 Ubitrix Inc.
HIPAA CHECKER
Ubitrix Inc. VERSION 1.0.8
Enabled
Health Checker Plugin
test-android-app – MainActivity.kt [test-android-app.app.main]
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Check HIPAA
testandroidapp › app › src › main › java › com › example › test_android_app › MainActivity
app
Galaxy Nexus API 30
Project Files
.hipaa_jwt
MainActivity.kt
~/AndroidStudioProjects/testandroidapp
.gradle
.idea
app

Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not work properly.

```
package com.example.test_android_app
```

Figure 7: Developed IDE plugin flow for checking HIPAA compliance for Developers

In addition to the high-level report, users can also view a code-level report that includes the line numbers of the relevant lines of code and a matching section of code that is connected to the relevant HIPAA regulation. Users can click on a specific rule's name to learn more about its sub-rules match. If the user clicks on a specific link in the report, they can view the full source code file with the specific matched line highlighted. Our platform provides a comprehensive and easy-to-use way for developers to ensure HIPAA compliance in their applications. By following these steps and using our service, developers can confidently deploy their applications knowing they comply with HIPAA regulations.

We developed an analytical API that can be connected to Integrated Development Environments (IDEs), such as Android Studio, to help developers ensure that their healthcare applications are HIPAA-compliant. Installing our developed IDE plugin, developers can easily check their code for compliance with HIPAA regulations by clicking a "Check HIPAA" button. When the button is pressed, the source code is compressed and uploaded to a centralized server, where it is processed. The server then emails the developer a report on the code's HIPAA compliance status, providing a granular level of detailed analysis that highlights specific line numbers where HIPAA compliance may be lacking. Our HIPAAChecker report greatly minimizes the work required of developers to ensure that their healthcare applications adhere to HIPAA regulations. By facilitating developers' ability to resolve HIPAA-related security issues throughout the development process in their preferred development environment, our tool enhances the overall quality and compliance of the code, increasing the effectiveness and efficiency of development.

CHAPTER 5
TESTING AND EVALUATION

We conducted a thorough investigation to assess the potential risk of HIPAA violations by downloading 285 mHealth apps from the Google Play Store and Github's Medical and Health & Fitness categories. The apps were chosen based on specific features and functionalities related to the storage, processing, management, and transfer of Electronic Protected Health Information (EPHI). We also considered the privacy policies, terms of service, and collection techniques of the selected apps from various geographical locations.

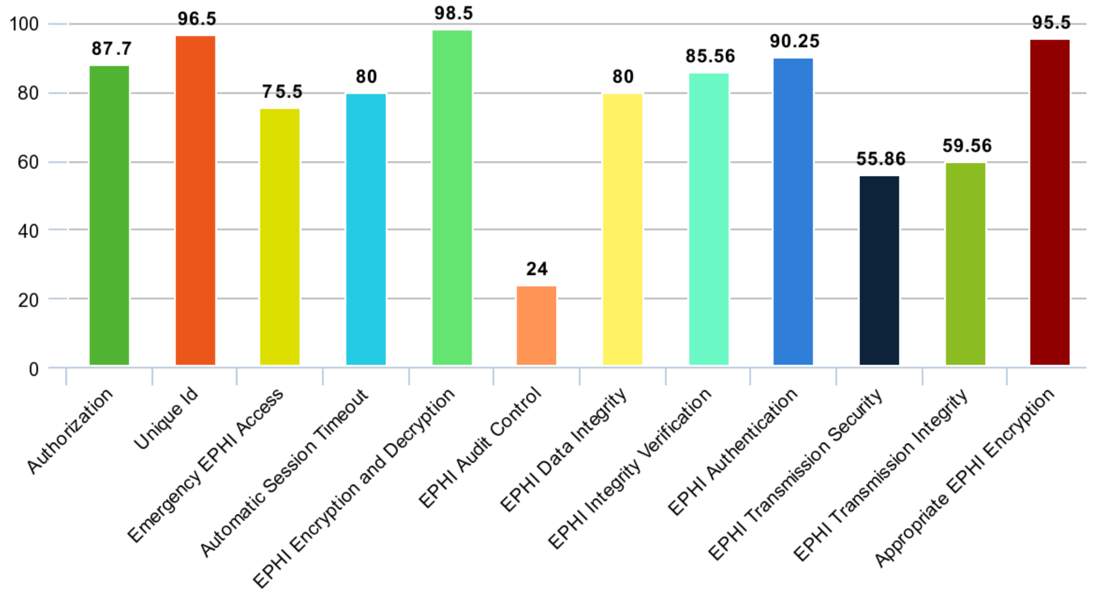


Fig. 8: Percentage of apps that met the HIPAA technical safeguards

Our primary goal was to assess the potential risk of HIPAA violations and determine whether the selected mHealth apps adhere to the act's standards and regulations. To ensure a thorough analysis, we chose both top-rated downloaded apps (10M+ downloads) and low-rated apps (100+ downloads). The Google Play Store applications were tested using our developed web application, and the Github open-source repositories were tested using our developed Integrated Development Environment (IDE) plugin.

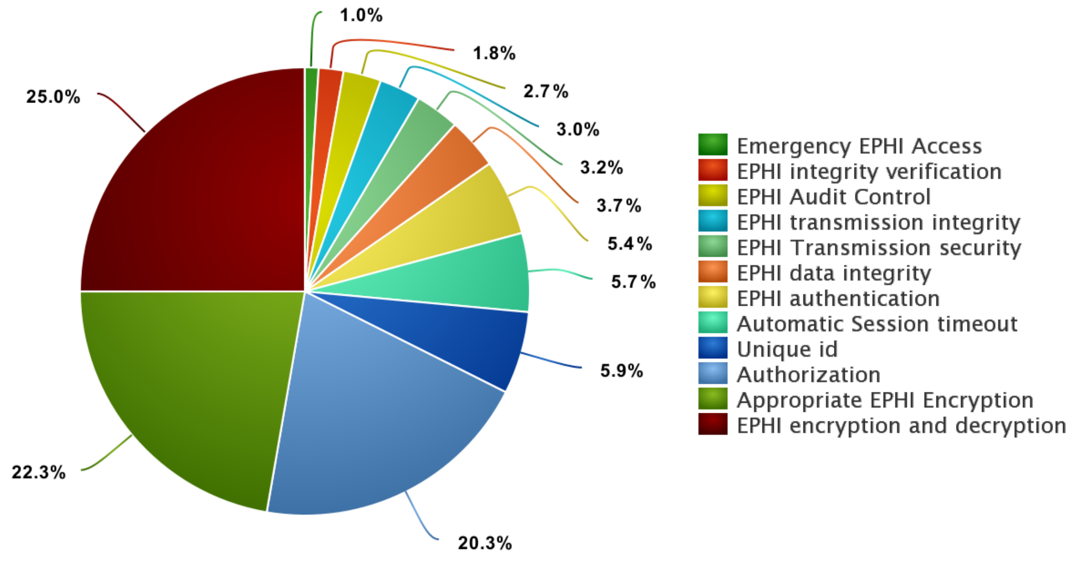


Fig. 9: Percentage of code segments detected in all downloaded apps that comply to HIPAA regulations

Our investigation revealed that a significant percentage of the downloaded apps lacked appropriate audit control or transmission security measures, potentially resulting in HIPAA violations. We discovered, in particular, that nearly half of the mHealth apps lacked appropriate audit controls to track and monitor user access to EPHI. Furthermore, approximately 40% of the apps lacked appropriate transmission security measures to protect EPHI from unauthorized access while in transit.

The majority of the apps, on the other hand, included adequate authorization, unique user identification, and encryption/decryption mechanisms for EPHI. We discovered that more than 90% of the apps used unique user identification to authenticate user access to EPHI, and that more than 80% of the apps used EPHI encryption/decryption mechanisms to protect sensitive health information.

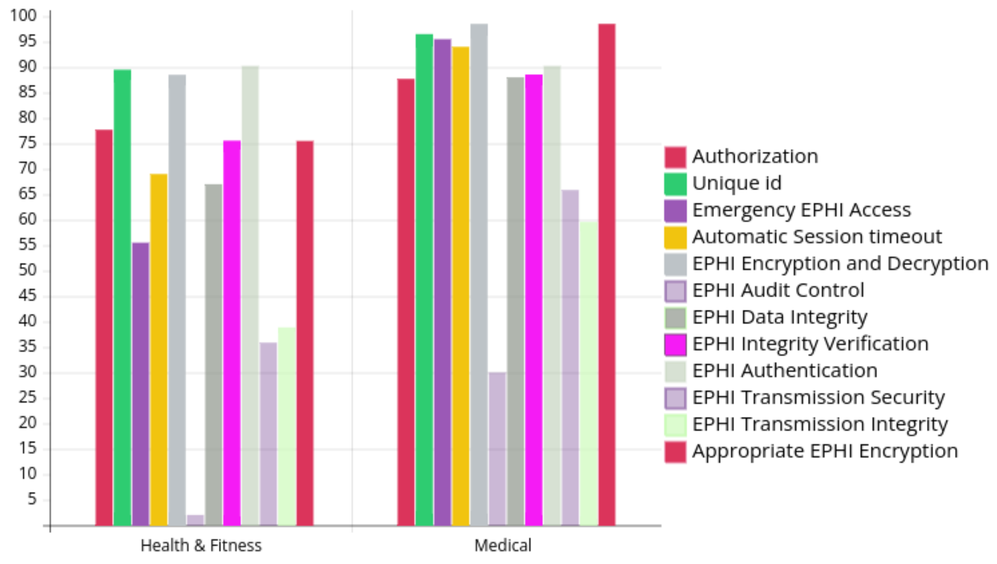


Fig. 10: Percentage of apps that met the HIPAA technical safeguards with respect to apps categories

Our findings are presented graphically, with Figures 8, 9 and 10 illustrating the findings of our investigation. Furthermore, we discovered that medical applications were more HIPAA compliant than health and fitness apps. This could be due to the nature of medical apps that handle sensitive patient information, as well as the requirement for more stringent security measures to comply with HIPAA regulations.

Our investigation sheds light on the potential risks of HIPAA violations posed by mHealth apps. According to our findings, many mHealth apps lack appropriate audit control or transmission security measures, which could lead to HIPAA violations.

CHAPTER 6
RECOMMENDATIONS

It is evident that there is a great need for advancements in the security and privacy procedures of healthcare apps. Our analysis report shows that the main hazards of these applications are lack of audit control, unsecured information sharing with third-party APIs, and unauthorized access to critical resources. We propose the following actions:

Recommendations for Application Users:

- Before sharing sensitive health information, thoroughly read the application's review and privacy policy. Look for signs of HIPAA compliance, such as adherence to privacy standards and adequate security measures.
- Before sharing personal health information, use tools to validate HIPAA compliance. This can assist guarantee that your data is handled correctly and securely.
- Examine and critique programs in order to educate others and aid researchers and developers in developing the app successfully. Giving input may assist in identifying and addressing any security issues, as well as improving overall user privacy.

Recommendations for Application Developers:

- Implement audit measures to allow for a full investigation of all incidents. This can aid in identifying and addressing any security flaws before they are exploited.
- When integrating external APIs, employ SSL to ensure safe data delivery.

- Use suitable access control mechanisms to guarantee that sensitive EPHI is only accessed by authorized personnel. This can aid in the prevention of unwanted access and the protection of user privacy.
- Use cutting-edge encryption and decryption technology to protect sensitive information from unauthorized access and ensure data security.
- To safeguard sensitive information from unwanted access and assure data security, use cutting-edge encryption and decryption technologies.

Users and developers may collaborate to enhance the security and privacy standards of healthcare applications by adopting these guidelines. It is critical that all stakeholders engaged in the development and usage of healthcare apps take responsibility for guaranteeing the safety and security of personal health information.

CHAPTER 7
CONCLUSION

The use of mHealth applications is widespread, yet many of them have security and privacy flaws and don't adhere to HIPAA Technical Safeguard requirements. Our developed HIPAAChecker can solve this issue by spotting the absence of technical security measures in both released and under-development applications. By guaranteeing that applications are in compliance with HIPAA regulations, this framework seeks to increase the trust of application users. Using our tool, developers can find and fix any security or privacy flaws in their applications. By incorporating HIPAA safety measures, the healthcare and fitness industry can improve the security of sensitive EPHI and build trust between patients and healthcare providers.

REFERENCES

[1] M. R. Mia, H. Shahriar, M. Valero, N. Sakib, B. Saha, M. A. Barek, M. J. H. Faruk, B. Goodman, R. A. Khan, and S. I. Ahamed, “A comparative study on hipaa technical safeguards assessment of android mhealth applications,” Smart Health, vol. 26, p. 100349, 2022.

[2] B. Pieper, “An overview of the hipaa security rule, part ii: Standards and specifications.” Optometry (St. Louis, Mo.), vol. 75, no. 11, pp. 728–730, 2004.

[3] F. Zubaydi, A. Saleh, F. Aloul, and A. Sagahyroon, “Security of mobile health (mhealth) systems,” in 2015 IEEE 15th international conference on bioinformatics and bioengineering (BIBE). IEEE, 2015, pp. 1–5.

[4] E. P. Morera, I. de la Torre D´ıez, B. Garcia-Zapirain, M. L´opezCoronado, and J. Arambarri, “Security recommendations for mhealth apps: Elaboration of a developer’s guide,” Journal of medical systems, vol. 40, pp. 1–13, 2016.

[5] B. Pieper, “An overview of the hipaa security rule, part ii: Standards and specifications.” Optometry (St. Louis, Mo.), vol. 75, no. 11, pp. 728–730, 2004.

[6] H. Kharrazi, R. Chisholm, D. VanNasdale, and B. Thompson, “Mobile personal health records: an evaluation of features and functionality,” International journal of medical informatics, vol. 81, no. 9, pp. 579– 593, 2012.

[7] R. Adhikari, D. Richards, and K. Scott, “Security and privacy issues related to the use of mobile health apps.” ACIS, 2014. [8] find sec bugs. (2023, Apr.) Find security bugs. [Online]. Available: https://find-sec-bugs.github.io/

[9] Eclipse. (2023, Apr.) Eclipse ide. [Online]. Available: https://www.eclipse.org/ide/

[10] guardsquare. (2023, Apr.) Dexguard. [Online]. Available: https://www.guardsquare.com/en/blog/dexguard-vs-proguard

[11] L. Li, T. F. Bissyand ́e, M. Papadakis, S. Rasthofer, A. Bartel, D. Octeau, J. Klein, and L. Traon, "Static analysis of android apps: A systematic literature review," Information and Software Technology, vol. 88, pp. 67–95, 2017.

[12] TrustKit. (2023, Apr.) Trustkit. [Online]. Available: https://github.com/ datatheorem/TrustKit

[13] J. Randolph, M. J. H. Faruk, B. Saha, H. Shahriar, M. Valero, L. Zhao, and N. Sakib, "Blockchain-based medical image sharing and automated critical-results notification: A novel framework," in 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), 2022, pp. 1756–1761.

[14] J. M. Waghmare and M. M. Chitmogrekar, "A review on malware detection methods," SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology, vol. 14, no. 01, pp. 38–43, 2023.

[15] jetbrains. (2023, Apr.) Intellij idea. [Online]. Available: https://www.jetbrains.com/idea/

[16] M. Farhadi, H. Haddad, and H. Shahriar, "Compliance of open source ehr applications with hipaa and onc security and privacy requirements," 2019.

[17] P. Ouellette, "A look at hipaa physical safeguard requirements," 2012.

[18] E. Snell. (2015, Apr.) A review of common hipaa administrative safeguards. [Online]. Available: https://healthitsecurity.com/news/ a-review-of-common-hipaa-administrative-safeguards

[19] E. Staff. (2023, Apr.) Hipaa technical safeguards: A basic review. [Online]. Available: https://healthitsecurity.com/news/hipaa-technical-safeguards-basic-review

[20] U. S. Government. (2007, Apr.) Security standards: Administrative safeguards. [Online]. Available: https://www.hhs.gov/sites/default/files/ocr/privacy/hipaa/administrative/securityrule/adminsafeguards.pdf

[21] JADX. (2023, Apr.) jadx - dex to java decompiler. [Online]. Available: https://github.com/skylot/jadx

[22] Danny Allan. 2008. Web application security: automated scanning versus manual penetration testing. IBM Rational Software, Somers, White Paper (2008).

[23] Nisreen Alzahrani and Daniyal Alghazzawi. 2019. A review on android ransomware detection using deep learning techniques. In Proceedings of the 11th International Conference on Management of Digital EcoSystems. 330–335.

[24] Daniel C Baumgart. 2020. Digital advantage in the COVID-19 response: perspective from Canada's largest integrated digitalized healthcare system. NPJ digital medicine 3, 1 (2020), 1–4.

[25] Vanessa N Cooper, Hossain Shahriar, and Hisham M Haddad. 2014. A survey of android malware characterisitics and mitigation techniques. In 2014 11th International Conference on Information Technology: New Generations. IEEE, 327–332.

[26] Jessica Davis. 2019. California Provider to Close After Ransomware Attack Damages System. Retrieved April 25, 2023 from https://healthitsecurity.com/news/california-provider-to-close-after-ransomware-attack-damages-system

[27] Nathan Eddy. 2019. Alabama hospital system DCH pays to restore systems after ransomware attack. Retrieved April 25, 2023 from https://www.healthcareitnews.com/news/alabama-hospital-system-dch-pays-restore-systems-after-ransomware-attack

[28] Md Jobair Hossain Faruk, Hossain Shahriar, Maria Valero, Farhat Lamia Barsha, Shahriar Sobhan, Md Abdullah Khan, Michael Whitman, Alfredo Cuzzocrea, Dan Lo, Akond Rahman, et al. 2021. Malware Detection and Prevention using Artificial Intelligence Techniques. In 2021 IEEE International Conference on Big Data (Big Data). IEEE, 5369–5377.

[29] HSS.gov. 2016. Fact Sheet: Ransomware and HIPAA. Retrieved April 25, 2023 from https://www.hhs.gov/sites/default/files/RansomwareFactSheet.pdf

[30] Justice. 2015. United States Government Interagency Guidance Document, How to Protect Your Networks from Ransomware. Retrieved April 25, 2023 from https://www.justice.gov/criminal-ccips/file/872771/download

[31] Mina Esmail Zadeh Nojoo Kambar, Armin Esmaeilzadeh, Yoohwan Kim, and Kazem Taghva. 2022. A survey on mobile malware detection methods using machine learning. In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 0215–0221.

[32] Paul Krebs, Dustin T Duncan, et al. 2015. Health app use among US mobile phone owners: a national survey. JMIR mHealth and uHealth 3, 4 (2015), e4924.

[33] Abdullahi Mohammed Maigida, Shafi'i Muhammad Abdulhamid, Morufu Olalere, John K Alhassan, Haruna Chiroma, and Emmanuel Gbenga Dada. 2019. Systematic literature review and metadata analysis of ransomware attacks and detection

[34] S Anne Moorhead, Diane E Hazlett, Laura Harrison, Jennifer K Carroll, Anthea Irwin, and Ciska Hoving. 2013. A new dimension of health care: systematic review of the uses, benefits, and limitations of social media for health communication. Journal of medical Internet research 15, 4 (2013), e1933.

[35] Lis Neubeck, Tina Hansen, Tiny Jaarsma, Leonie Klompstra, and Robyn Gallagher. 2020. Delivering healthcare remotely to cardiovascular patients during COVID-19: a rapid review of the evidence. European Journal of Cardiovascular Nursing 19, 6 (2020), 486–494. [47] => [36] G Perna. 2018. The State of Mobile Health in Today's Practice. Retrieved April 25, 2023 from https://www.physicianspractice.com/article/state-mobile-health-todays-practice

[37] Nanda Rani, Sunita Vikrant Dhavale, Amarjit Singh, and Atul Mehra. 2022. A Survey on Machine Learning-Based Ransomware Detection. In Proceedings of the Seventh International Conference on Mathematics and Computing. Springer, 171–186.

[38] Shweta Sharma, Rakesh Kumar, and C Rama Krishna. 2021. A survey on analysis and detection of Android ransomware. Concurrency and Computation: Practice and Experience 33, 16 (2021), e6272.

[39] F. Sivilli. 2019. New OCR HIPAA Media Guidance: Apps & The Disclosure of PHI. Retrieved April 25, 2023 from https://compliancy-group.com/new-ocr-guidance-hipaa-compliant-apps-health-information/

[40] US Consumer Survey. 2019. Physicians Using Mobile Apps Seen as a Major Differentiator Amongst US Patients. Retrieved April 25, 2023 from

https://www.globenewswire.com/news-release/2019/06/06/1865254/0/en/U-S-Consumer-Survey-Physicians-Using-Mobile-Apps-Seen-as-a-Major-Differentiator-Amongst-U-S-Patients.html

[41]Jaishri M Waghmare and Mayuri M Chitmogrekar. 2022. A Review on Malware Detection Methods. SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology 14, 01 (2022), 38–43.

[42]United States Government. 2007. Code of Federal Regulations - Title 45: Public Welfare. 2007. p. 738. Retrieved April 25, 2023 from https://www.hhs.gov/ohrp/sites/default/files/ohrp/policy/ohrpregulations.pdf

[43] United States Government. 2007. Security Standards: Administrative Safeguards. Retrieved April 25, 2023 from

https://www.hhs.gov/sites/default/files/ocr/privacy/hipaa/administrative/securityrule/adminsafeguards.pdf

[44] United States Government. 2016. Health App Use Scenarios & HIPAA. retrieved April 25, 2023from https://www.hhs.gov/sites/default/files/ocr-health-App-developer-scenarios-2-2016.pdf

[45] George Grispos, Talon Flynn, William Bradley Glisson, and Kim-Kwang Raymond Choo. 2021. Investigating Protected Health Information Leakage from Android Medical Applications. In International Conference on Future Access Enablers of Ubiquitous and Intelligent Infrastructures. Springer, 311–322.

[46] Accountability Act. 1996. Health insurance portability and accountability act of 1996. Public law 104 (1996), 191.

[47] G. Contributor. 2018. Can Mobile Health Apps Be Made More Secure? Retrieved April 25, 2023 from https://www.aberdeen.com/techpro-essentials/can-mobile-health-apps-made-secure

[48] M. Oliynyk. 2016. Why is healthcare data security so important? Retrieved April 25, 2023 from https://www.protectimus.com/blog/why-is-healthcare-data-security-so-important/

[49] Ben Smith, Andrew Austin, Matt Brown, Jason T King, Jerrod Lankford, Andrew Meneely, and Laurie Williams. 2010. Challenges for protecting the privacy of health information: required certification can leave common vulnerabilities undetected. In Proceedings of the second annual workshop on Security and privacy in medical and home-care systems. 1–12.

[50] HealthIT. 2015. Guide to Privacy and Security of Electronic Health Information. Retrieved April 2, 2022 from https://www.healthit.gov/sites/default/files/pdf/privacy/privacy-and-security-guide.pdf

[51] Kenneth D Mandl and Eric D Perakslis. 2021. HIPAA and the leak of “deidentified” EHR data. N Engl J Med 384, 23 (2021), 2171–2173

[52]Patrick Howell O’Neill. 2020. A patient has died after ransomware hackers hit a German hospital. Retrieved April 2, 2022 from https://www.technologyreview.com/2020/09/18/1008582/a-patient-has-died-after-ransomware-hackers-hit-a-german-hospital/

[53] Jessica Davis. 2021. UHS Ransomware Attack Cost $67M in Lost Revenue, Recovery Efforts. Retrieved April 25, 2023 from https://healthitsecurity.com/news/uhs-ransomware-attack-cost-67-million-in-recovery-lost-revenue

[54] Sophie Carson. 2019. Hackers hold Milwaukee-based tech company’s data for ransom; nursing homes affected. Retrieved April 25, 2023 from https://www.jsonline.com/story/news/local/2019/11/23/milwaukee-firm-falls-victim-hackers-100-plus-nursing-homes-affected/4285213002/

[55] Saha, B., Tahora, S., Barek, A., & Shahriar, H. (n.d.). HIPAAChecker: The Comprehensive Solution for HIPAA Compliance in Android mHealth Apps. In Proceedings of the IEEE International Workshop on Security, Trust, and Privacy for Software Applications (p. 5703). (Under review)