Secure iOS mHealth Apps Development: An IDE-Embedded Framework for HIPAA-Aware Coding

Md Bajlur Rashid[1*], **MD Abdul Barek**[2†], **Md Mostafizur Rahman**[2†], **Sharmin Yeasmin**[3‡], **Hossain Shahriar**[4§], **Sheikh Iqbal Ahamed**[4§]

[*] Department of Cybersecurity and Information Technology, University of West Florida, USA
[†] Department of Intelligent Systems and Robotics, University of West Florida, USA
[†] Department of Cybersecurity and Information Technology, University of West Florida, USA
[‡] Department of Computer Science and Engineering, South East University, Bangladesh
[§] Center for Cybersecurity, University of West Florida, USA
[§] Department of Computer Science, Marquette University, Milwaukee, WI, United States
{ mr248@students.uwf.edu*, mb381@students.uwf.edu †, mr240@students.uwf.edu †, sharmin10seu@gmail.com ‡, hshahriar@uwf.edu §, sheikh.ahamed@mu.edu§}

***Abstract*—With the rapid growth of technology, accessing digital health records has become increasingly easier. Especially mobile health technology like mHealth apps help users to manage their health information, as well as store, share and access medical records and treatment information. Along with this huge advancement, mHealth apps are increasingly at risk of exposing protected health information (PHI) when security measures are not adequately implemented. The Health Insurance Portability and Accountability Act (HIPAA) ensures the secure handling of PHI, and mHealth applications are required to comply with its standards. But it is unfortunate to note that many mobile and mHealth app developers, along with their security teams, lack sufficient awareness of HIPAA regulations, leading to inadequate implementation of compliance measures. Moreover, the implementation of HIPAA security should be integrated into applications from the earliest stages of development to ensure data security and regulatory adherence throughout the software lifecycle. This highlights the need for a comprehensive framework that supports developers from the initial stages of mHealth app development and fosters HIPAA compliance awareness among security teams and end users. An iOS framework has been designed for integration into the Integrated Development Environment(IDE), accompanied by a web application to visualize HIPAA security concerns in mHealth app development. The web application is intended to guide both developers and security teams on HIPAA compliance, offering insights on incorporating regulations into source code, with the IDE framework enabling the identification and resolution of compliance violations during development. The aim is to encourage the design of secure and compliant mHealth applications that effectively safeguard personal health information.**

***Index Terms*—HIPAA Compliance, mHealth Applications, iOS Development, Xcode Integration, Swift Package Manager(SPM), HIPAA Checker, Developer Awareness, Security Framework.**

I. Introduction

As technology continues to improve, especially in mobile application development, health information has become increasingly accessible, manageable, and transferable. A recent study[1] reported that over 60% of mHealth app users regularly accessed their electronic health records, scheduled appointments, and reviewed medication or treatment plans through these platforms. Also a cross-sectional survey study published in 2025, found that 85.4%, across 20 countries, healthcare professionals can conduct virtual consultations with patients [2], provide digital prescriptions and treatment plans, and both patients and doctors can access medical test results in real-time. Furthermore, pharmacies can digitally access prescriptions and fulfill them by delivering the prescribed medication directly to the patient. This technological progression has greatly enhanced the usability and convenience of healthcare services, streamlining communication and care delivery.

However, while these advancements simplify the healthcare process, they also introduce significant security risks to PHI [3]. A comprehensive security assessment of mHealth applications found that 47% of iOS apps utilized backend servers with suboptimal security configurations, and 33% employed unsecured connections, posing risks to patient data confidentiality, authenticity, and integrity [4–6]. Medical and treatment data, such as medical conditions, treatment plans, diagnostic results, and prescriptions, constitute highly sensitive health information. Additionally, personally identifiable information (PII), including a patient's name, date of birth, and gender, is equally private and must be protected from unauthorized access. Safeguarding this information is essential for protecting patient privacy and keeping trust in healthcare systems.

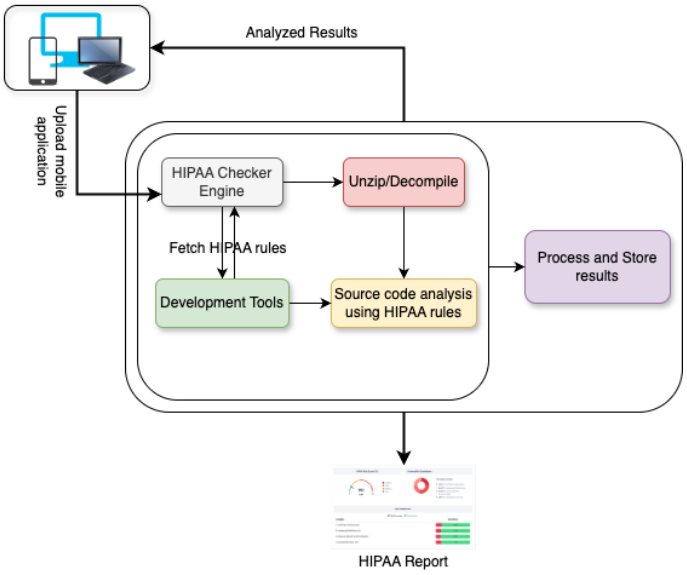


Fig. 1: HIPAA Checker Framework Architecture

The Health Insurance Portability and Accountability Act (HIPAA) defines rules for the secure handling of PHI through technical safeguards such as access control, audit logs, encryption, transmission security, and authentication[7, 8]. PHI must be encrypted during storage and transfer, and access should be strictly controlled through proper authorization and authentication mechanisms. Failure to imple-

Reference	Rule Id	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_access	Create and use processes for acquiring required digitally protected health information in an emergency.
164.312(a)(2)(iii)	user_inactivity	Implement software procedures that end a session after a certain period of inactivity.
164.312(a)(2)(iv)	encryption_decryption	Implement a system for encrypting and decrypting EPHI.
164.312(b)	audit	Implement methods for recording and examining activities in information systems that use or include EPHI.
164.312(c)(1)	data_integrity	Implement regulations and procedures to prevent unauthorized manipulation or destruction of EPHI.
164.312(c)(2)	authorization_for_destruction	Utilize technological tools to verify that electronically stored protected health information has not been tampered with or deleted without authorization.
164.312(d)	user_authentication	Establish processes to confirm that the individual or organization requesting access to EPHI is who is being identified.
164.312(e)(1)	transmition_secuirity	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)	guard_against_com_network	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)	phi_encryption	Implement a mechanism to encrypt EPHI whenever deemed appropriate.

TABLE I: HIPAA Technical Safeguards

ment these measures exposes sensitive data—like medical conditions, diagnostic results, or treatment plans—to risks such as unauthorized access, data breaches, identity theft, and potential fraud[9, 10].

These security lapses not only threaten patient privacy and care quality, but also pose legal and financial risks to developers and healthcare providers[11]. In 2024, Montefiore Medical Center paid a $4.75 million HIPAA settlement after an employee accessed over 12,000 patient records without authorization, highlighting the severe legal and financial consequences of inadequate compliance[12, 13]. Therefore, rigorous adherence to HIPAA safeguards is essential for ensuring data integrity, maintaining trust, and achieving regulatory compliance in health applications.

Developing a HIPAA-compliant health application, particularly a mHealth app, requires a clear understanding of the technical safeguards that must be integrated into the software. A Security Journey Study reveals that 20% of organizations are confident in detecting vulnerabilities before release, while over 60% face challenges in remediating them, highlighting the critical need to address vulnerabilities during the development phase to ensure secure

applications [14, 15]. However, developers and security teams may not be fully aware of how HIPAA measures should be implemented in many cases. Furthermore, the technical safeguards outlined by HIPAA are often broadly defined, which may lead to misinterpretation or inconsistent implementation. While the development team may attempt to incorporate privacy and security measures at later stages, such as during final development or just prior to deployment, this reactive approach may not be sufficient[16, 17]. Effective security measures need to be planned at the software architecture design level and should be integrated throughout the entire development lifecycle to ensure comprehensive protection of sensitive health data.

To support HIPAA compliance in mHealth development, we introduce xPlugin—an iOS package integrated via Swift Package Manager in Xcode. It performs static code analysis, generates a HIPAA compliance report with risk scores and line-level remediation guidance, and sends results to the HIPAA Checker portal. Compatible with Objective-C and Swift (iOS 11+, Swift 5.2+), it offers broad applicability. Figure 4 illustrates its integration in Xcode. A web dashboard further helps teams visualize compliance metrics and track safeguard implementation throughout development, ensuring continuous security and regulatory alignment.

The aim of this research is to promote secure and compliant mHealth apps that protect personal health information. The main objectives are:

- Develop a source code analysis framework to assess mHealth apps for HIPAA Technical Safeguard compliance.
- Integrate an analytical dashboard to visualize HIPAA risk scores and vulnerabilities with detailed reports.
- Conduct a meta-analysis to identify risk factors, evaluate safety mechanisms, and detect HIPAA non-compliance.
- Design and implement a Swift Package for Xcode to provide real-time static analysis and feedback for early security and privacy issue detection.

The remainder of this paper is organized as follows: Section II outlines the research methods employed in this study. Section III presents the architecture of the HIPAA Checker frameworks. Section IV details the framework development process. Section V discusses the testing procedures and evaluation results. Section VI provides key recommendations. Section VII addresses limitations and proposes directions for future research. Finally, Section VIII concludes the paper.

II. RESEARCH METHODS

The Health Insurance Portability and Accountability Act (HIPAA) outlines three core categories of security requirements: administrative, physical, and technical safeguards. Table I presents the technical

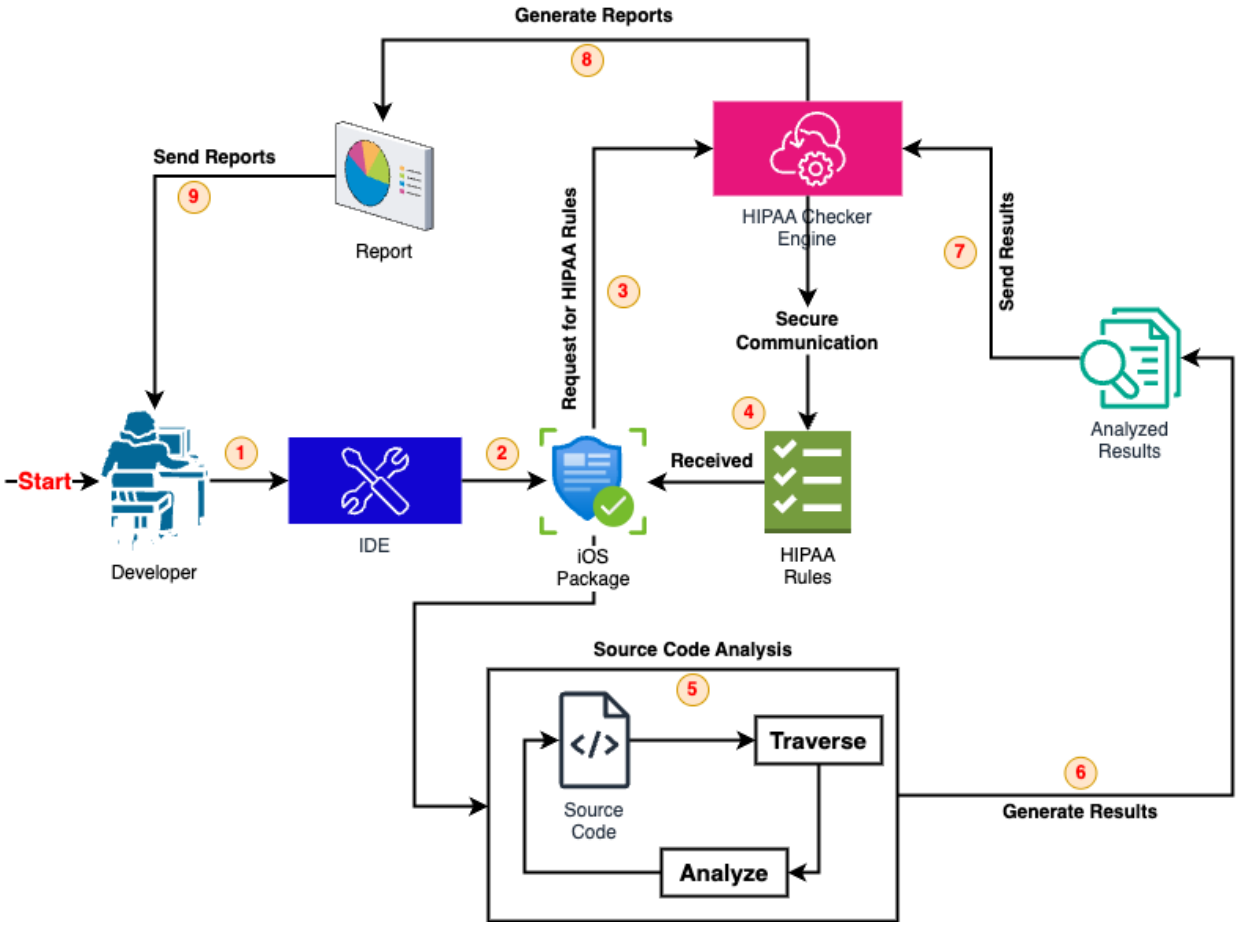


Fig. 2: HIPAA Checker xPlugin workflow

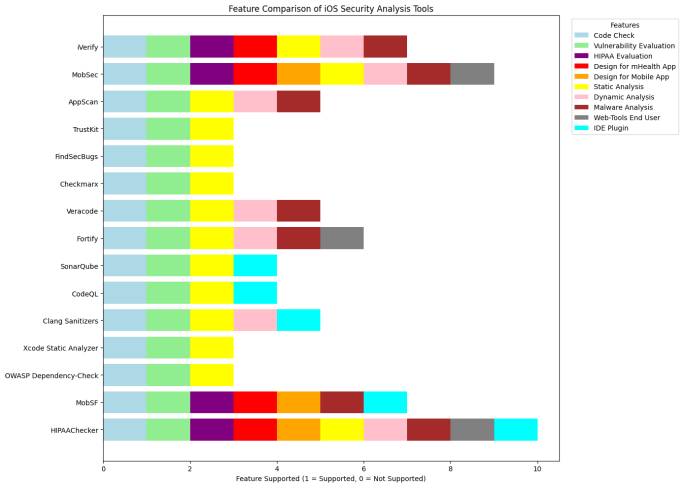


Fig. 3: Comparison of the features of the proposed framework with those of similar products on the market

safeguard requirements of HIPAA as defined by the US Department of Health and Human Services (HHS), outlining the essential security standards and implementation specifications necessary to ePHI in digital health systems.

Administrative safeguards include policies and procedures that guide the selection, development, implementation, and maintenance of security measures designed to protect health information[3, 18, 19]. Physical safeguards refer to mechanisms, rules, and practices that protect electronic systems, related equipment, and data from environmental hazards and unauthorized physical access[20]. In contrast, technical safeguards refer to the technological tools and policies put in place to defend electronically protected health information (EPHI) from unauthorized access.

This research focuses on the development and application of source code analysis methods focusing on technical safeguards within mHealth applications. By ensuring compliance with technical safeguards, it becomes feasible to indirectly support broader administrative and physical security objectives. For example, tools developed to detect technical non-compliance can also aid in monitoring administrative controls and preventing incidents such as breaches resulting from inadequate physical protection. A practical case is the difficulty of retrieving encrypted PHI from a lost or stolen mobile device running a compliant mHealth application, thereby reinforcing both technical and physical security postures.

III. HIPAACHECKER FRAMEWORK

This study introduces a methodology for analyzing iOS mHealth applications to ensure HIPAA compliance in data storage and transmission. Unlike general iOS security tools, the proposed framework scans

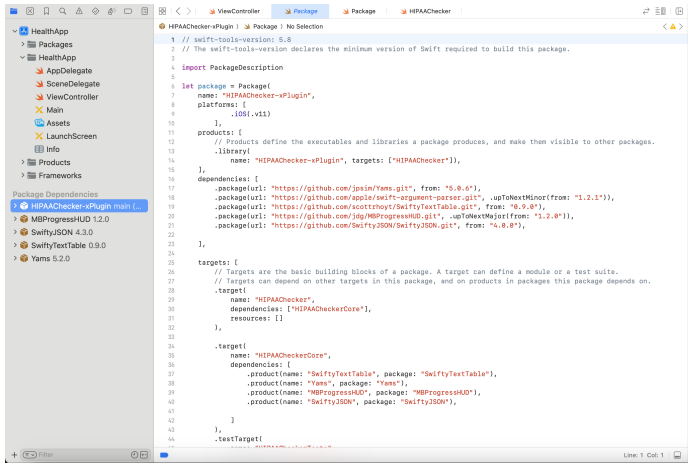
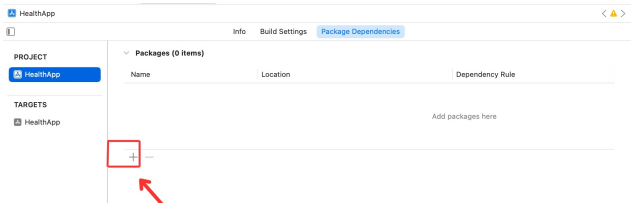
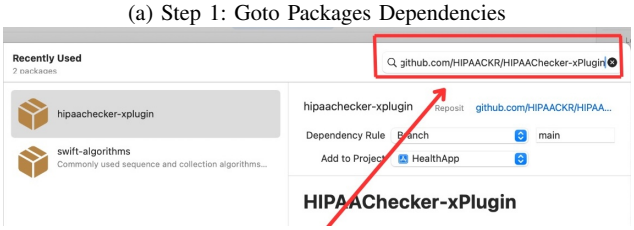


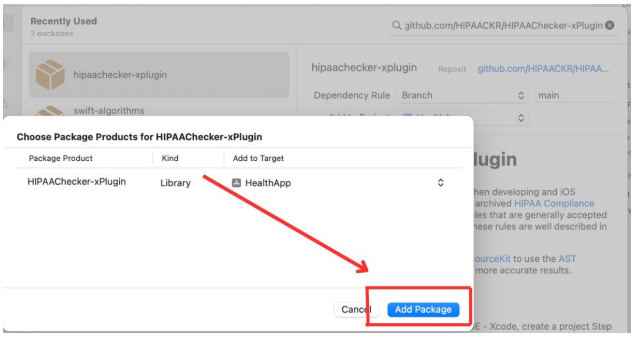
Fig. 4: HIPAA Checker xPlugin



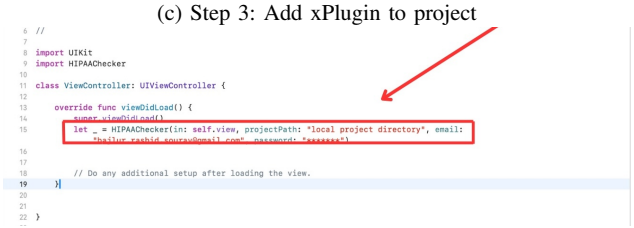
(a) Step 1: Goto Packages Dependencies



(b) Step 2: Search for xPlugin

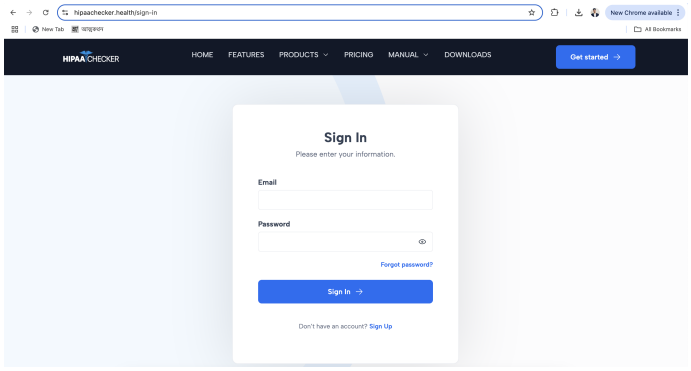


(c) Step 3: Add xPlugin to project

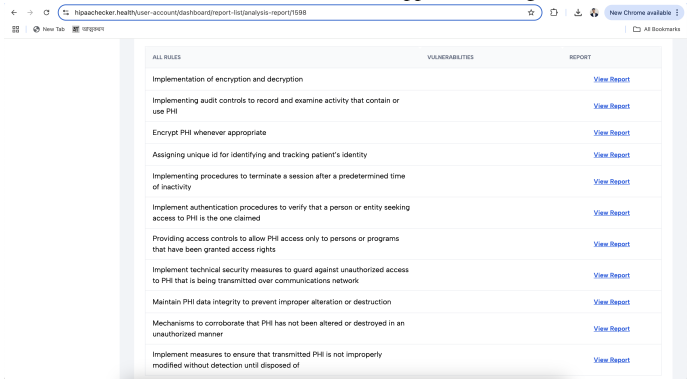


(d) Step 4: Initialize xPlugin

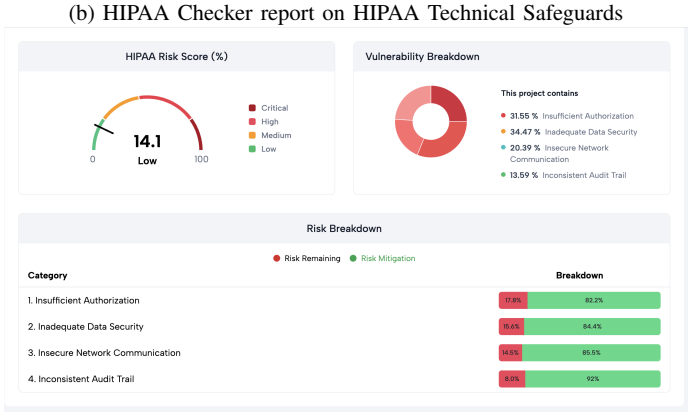
Fig. 5: Procedure for Integrating xPlugin during the Development.



(a) HIPAA Checker Web app onboarding



(b) HIPAA Checker report on HIPAA Technical Safeguards



(c) HIPAA Risk score summarization and CVSS score on vulnerabilities

Fig. 6: HIPAA Checker Web application and HIPAA risk assessment

source code for HIPAA-specific security and privacy patterns. The system architecture is shown in Figure 1, and its feature comparison with existing tools appears in Figure 3. Developers can use the Xcode-integrated xPlugin before deployment, while others may upload source files via the HIPAA Checker web platform, as outlined in Figure 2. xPlugin fetches HIPAA rule sets—including the ”authorization” rules in Table II—to analyze Swift and Objective-C code and generate compliance reports highlighting specific lines needing attention.

Overall, the proposed framework offers mHealth app developers a systematic approach to verifying that their applications align with HIPAA's technical safeguards, ultimately enhancing patient data privacy and security.

Rule Id	Subrule Id	Code Patterns
authorization	authorization_exception	- LAErrorAuthorizationFailed - LAErrorNotInteractive - NSError.domain == LAErrorDomain - case (.accessDenied|.notAuthorized)
	illegal_access	- SecAccessControlCreateFlags - LAErrorUserCancel - LAErrorSystemCancel - kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly
	authorization_enforcement	- SecItemCopyMatching.*kSecUseAuthenticationUI = kSecUseAuthenticationUIFail - evaluatePolicy(LAPolicyDeviceOwnerAuthentication - SecAccessControlCreateWithFlags.*kSecAccessControlApplicationPassword - requireUserPresence
	access_control_exception	- kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly - SecAccessControlCreateWithFlags.*(biometryCurrentSet|devicePasscode) - userInteractionAllowed = false - validateAuthentication(UIViewController(
	biometric_authentication	- LAPolicyDeviceOwnerAuthenticationWithBiometrics - LAContext.canEvaluatePolicy(.deviceOwnerAuthenticationWithBiometrics) - LocalizedReason = ”Access health records” - kSecAccessControlBiometryCurrentSet - LAErrorBiometryLockout - addConstraint(.touchIDAny)

TABLE II: iOS Code Patterns corresponding to one HIPAA rule ID and Sub Rule IDs

IV. A. Framework Development:

To facilitate automated HIPAA compliance verification in mHealth application development, we propose a comprehensive framework comprising a web-based analysis platform and an IDE-integrated tool—xPlugin—targeted specifically for iOS development environments such as Xcode.

1) xPlugin: IDE-Level Compliance Enforcement: The core component of the proposed methodology is xPlugin, a Swift Package Manager (SPM)–distributable plugin designed for seamless integration with the Xcode IDE. It can be downloaded by Xcode from this GitHub Repository. Figure 5 shows the steps to integrate it into Xcode. xPlugin connects directly to the HIPAA Checker analytical engine through secure API communication channels. Upon initialization, developers are prompted to authenticate using their credentials from the HIPAA Checker web platform and specify the root directory of the project. It is advisable to perform this initialization at the inception of the development process to ensure continuous compliance tracking.

Once configured, xPlugin fetches the latest HIPAA compliance rules from the remote engine. These rules consist of technical safeguards defined by HIPAA, including subrules and source code patterns tailored to the privacy and security of PHI. The plugin then initiates a static analysis pass over the source code, detecting compliance violations by matching against the predefined rule set.

Following the local analysis, the plugin securely transmits the results to the HIPAA Checker engine, which performs a comprehensive evaluation. The engine identifies code-level vulnerabilities in four key categories aligned with HIPAA technical safeguards:

- Insufficient Authorization.
- Inadequate Data Security.
- Insecure Network Communication.
- Inconsistent Audit Trail.

Each detected issue is quantified using the Common Vulnerability Scoring System (CVSS), a widely recognized framework for evaluating the severity of software vulnerabilities. The resulting compliance report provides a granular overview of the application's HIPAA adherence, highlighting specific line numbers, affected files, and rule violations. This report also includes a risk percentage and mitigation recommendation, allowing developers to prioritize remediation efforts effectively.

By integrating directly into the development workflow, xPlugin minimizes the overhead associated with external audits and enables real-time compliance validation. This approach significantly enhances development efficiency and reduces the likelihood of non-compliance prior to deployment.

2) Web-Based HIPAA Checker Platform: Complementing the IDE tool, the web-based HIPAA Checker platform (link) allows broader accessibility for compliance analysis across diverse development frameworks. As depicted in Figure 2, the platform supports secure file upload, metadata extraction, and recursive source code scanning. Users undergo a two-step authentication process: standard credentials followed by two-factor authentication (2FA). After successful upload, the system extracts compiled and readable source code, performs a deep pattern-matching scan, and stores results in a structured database for further processing. The process is shown in Figure 6.

The final report, accessible via the platform's user interface, mirrors the xPlugin output—detailing matched subrules, categorized risk scores, and direct links to code segments requiring modification. Users may navigate to specific HIPAA rules, view matched lines in context,

and take immediate corrective action.

Together, the xPlugin and web platform form a robust, developer-centric ecosystem for ensuring HIPAA compliance in mHealth applications. By embedding compliance checks into the software development lifecycle, the framework empowers developers to produce secure, regulation-adherent applications with reduced manual intervention and higher assurance of patient data protection.

V. B. Testing and Evaluation

The source code for various iOS mHealth applications, developed in Swift and Objective-C, was manually collected from open-source repositories such as GitHub, GitLab, and other cloud-based platforms. A strict filtering process excluded inactive, incomplete, deprecated, or non-health-related projects. The selection emphasized apps handling ePHI, particularly those involving covered entities and business associates under HIPAA. Additional factors included geographical diversity, as well as privacy policies, data handling methods, and ePHI transmission practices.

After collecting the project source codes, we tested them using xPlugin. The results reveal a heterogeneous compliance landscape: High compliance levels were observed in eight of the implemented eleven safeguards. These include user_authentication, audit, data_integrity, user_inactivity, unique_id, encryption_decryption, transmition_secuirity, and guard_against_com_network. The prevalence of these safeguards suggests that developers are utilizing built-in iOS security features—such as Apple's Keychain, audit logging tools, and inactivity timeout mechanisms—to meet standard security expectations.

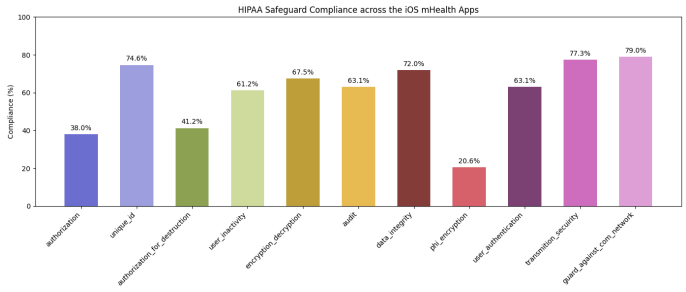


Fig. 7: HIPAA safeguards in iOS mHealth apps

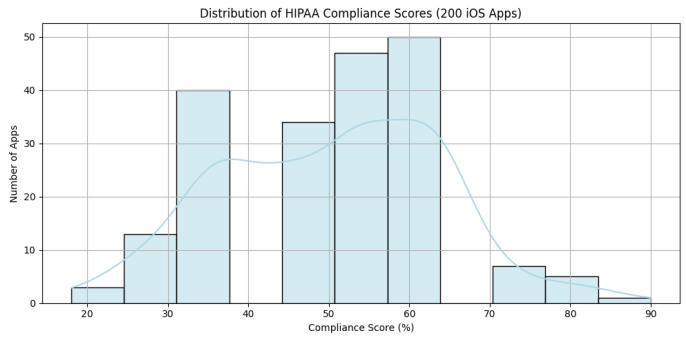


Fig. 8: HIPAA rules distributions in different iOS mHealth apps

In contrast, low compliance levels were detected for three critical safeguards: phi_encryption, authorization_for_destruction, and authorization. These areas represent core pillars of HIPAA's privacy and access control mandates. The weak enforcement of PHI-specific encryption mechanisms (phi_encryption) indicates that generic encryption is often applied without regard for data classification. Similarly, the limited presence of secure data deletion processes (authorization_for_destruction) and inconsistent implementation of access control logic (authorization) raise concerns about the ability of many applications to maintain proper data lifecycle management and user-specific data access restrictions.

These observations are illustrated in Figure 7 and 8. These findings suggest that while general security measures are partially addressed in open-source iOS applications, critical elements specifically tied to HIPAA's intent—particularly those concerning data governance and user-specific access permissions—are significantly underimplemented.

VI. Recommendations

To address HIPAA compliance gaps in iOS mHealth apps, developers should incorporate HIPAA considerations from the early stages of development. This includes conducting threat modeling, aligning design with regulatory requirements, and involving security teams throughout the software lifecycle for code audits and safeguard validation. Table III outlines actionable measures to support this process.

TABLE III: Key Recommendations for Improving HIPAA Compliance in iOS mHealth Applications

No.	Recommendation	Target Group	Purpose
1	Integrate a HIPAA compliance framework (e.g., HIPAA Checker) into the IDE	Developers	Enables real-time identification and remediation of security violations during development.
2	Implement PHI-specific encryption and secure deletion mechanisms	Developers	Ensures sensitive health information is protected and securely discarded in compliance with HIPAA.
3	Use static code analysis tools to detect and fix safeguard violations	Developers & Security	Facilitates automated detection of compliance issues early in the software lifecycle.
4	Conduct role-based access control and authorization logic design	Developers	Supports proper enforcement of access rights to sensitive data.
5	Provide HIPAA compliance training and documentation for all technical stakeholders	Developers & Security Teams	Enhances understanding of regulatory expectations and promotes consistent application of safeguards.

Furthermore, effective collaboration between legal, compliance, and technical teams is essential for translating regulatory language into actionable software practices. As health technology continues to evolve, maintaining agile compliance strategies through iterative testing, user feedback, and continuous security updates is critical to ensuring the confidentiality, integrity, and availability of ePHI.

VII. Limitations and Future Research Directions

This study is subject to several limitations that present opportunities for further investigation. First, the analysis relied primarily on open-source iOS mHealth applications available in Swift and Objective-C, which may not fully represent the broader landscape of commercial or proprietary health apps in the App Store. Consequently, the findings

may not capture the complete spectrum of HIPAA compliance practices in widely used, closed-source applications. Additionally, while the HIPAA Checker framework and accompanying web tool were evaluated against a curated dataset, the scope of safeguards assessed was limited to technical rules, excluding administrative and physical safeguards due to the nature of source code-based analysis.

Moreover, the framework's performance in real-time integrated development environments (IDEs) across diverse developer workflows was not extensively validated in longitudinal studies. This introduces potential variability in how effectively developers may adopt and benefit from the framework.

Future research should broaden compliance verification by integrating machine learning-based static and dynamic analysis techniques capable of detecting context-aware violations and inferring missing controls. Incorporating natural language processing (NLP) to analyze app privacy policies and terms of service may further enhance the framework's ability to evaluate compliance beyond source code. Additionally, large-scale empirical studies involving diverse industry practitioners are needed to

VIII. Conclusion

As the adoption of mHealth applications continues to accelerate, ensuring the security and privacy of ePHI has become a critical concern. This study highlights the significant gaps in HIPAA technical safeguard implementation across iOS-based mHealth applications. Through empirical analysis of publicly available source code and the development of a HIPAA compliance framework, this work provides a structured approach to identifying, visualizing, and addressing security vulnerabilities during the software development lifecycle.

The proposed xPlugin and HIPAA Checker framework , designed for integration within development environments, empowers developers and security teams to proactively detect and remediate compliance violations at the code level. Complemented by a web-based interface, the tool enhances transparency and offers actionable guidance aligned with regulatory standards. Furthermore, the study highlights the importance of embedding HIPAA awareness from the early stages of development and fostering close collaboration between development and security teams.

By advancing automated, accessible solutions for regulatory adherence, this work contributes to the broader goal of building secure, privacy-respecting health technologies. Continued refinement, expanded datasets, and broader community engagement will further enhance the framework's utility and impact within the evolving mHealth ecosystem.

IX. Acknowledgments

The work is supported by the National Science Foundation under Award $\#2433800$, $\#2421324$, $\#1946442$ and National Institutes of Health Grant $\#5R42LM014356-03$ Any opinions, findings, recommendations, expressed in this material are those of the authors and do not necessarily reflect the views of the NSF and NIH.

References

[1] Y. Zhou, H. Wang, and A. Singh, "The use of mhealth applications for self-management in chronic disease: A meta-analysis," *JMIR mHealth and uHealth*, vol. 11, no. 3, p. e43129, 2023.

[2] G. Kerr, G. Greenfield, E. Li, T. Beaney, B. W. Hayhoe, J. Car, A. Clavería, C. Collins, G. Gusso, R. D. Hoffman, *et al.*, "Factors associated with the availability of virtual consultations in primary care across 20 countries: Cross-sectional study," *Journal of Medical Internet Research*, vol. 27, p. e65147, 2025.

[3] 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.

[4] X. Jin, W. Zhang, and B. Li, "Security and privacy in mobile health applications: A comprehensive analysis," *Journal of Medical Internet Research*, vol. 21, no. 1, p. e9818, 2019.

[5] K. Rahkema and D. Pfahl, "Empirical study on code smells in ios applications," in *Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems*, pp. 61–65, 2020.

[6] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "Pios: Detecting privacy leaks in ios applications.," in *NDSS*, vol. 2011, p. 18th, 2011.

[7] B. Saha, S. Tahora, A. Barek, and H. Shahriar, "Hipaachecker: The comprehensive solution for hipaa compliance in android mhealth apps," in *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1822–1827, IEEE, 2023.

[8] M. A. Rahman, M. A. Barek, A. Riad, M. M. Rahman, M. B. Rashid, S. Ambedkar, M. R. Miaa, F. Wu, A. Cuzzocrea, and S. I. Ahamed, "Embedding with large language models for classification of hipaa safeguard compliance rules," *arXiv preprint arXiv:2410.20664*, 2024.

[9] A. K. I. Riad, M. A. Barek, M. M. Rahman, M. S. Akter, T. Islam, M. A. Rahman, M. R. Mia, H. Shahriar, F. Wu, and S. I. Ahamed, "Enhancing hipaa compliance in ai-driven mhealth devices security and privacy," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 2430–2435, IEEE, 2024.

[10] M. A. Barek, M. M. Rahman, S. Akter, A. K. Islam Riad, M. A. Rahman, H. Shahriar, A. Rahman, and F. Wu, "Mitigating insecure outputs in large language models(llms): A practical educational module," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 2424–2429, 2024.

[11] M. M. Rahman, A. S. Arshi, M. M. Hasan, S. F. Mishu, H. Shahriar, and F. Wu, "Security risk and attacks in ai: A survey of security and privacy," in *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1834–1839, IEEE, 2023.

[12] M. Perez, "Nearly $5m paid in early 2024 hipaa settlement," 2024. Accessed: 2025-05-07.

[13] M. B. Rashid, N. Islam, A. A. M. Sabuj, S. Waheed, and M. B. A. Miah, "Randomly encrypted key generation algorithm against side channel attack in cloud computing," in *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, pp. 1–5, IEEE, 2015.

[14] S. Journey, "Ponemon institute study on software vulnerabilities," 2020. Accessed: 2025-05-07.

[15] A. Forsanker, "Evaluating the code quality of ios applications generated by large language models," 2024.

[16] M. M. Rahman, M. R. Islam, M. S. Islam, and M. M. Rahman, "Mitigating software vulnerabilities through secure software development life cycle: A review," *Security and Privacy*, vol. 7, no. 1, p. e9962691, 2024.

[17] A. K. I. Riad, S. Ahmed, M. Z. Nizum, M. A. Barek, M. B. Rashid, M. M. Rahman, I. Guillermo Francia, H. Shahriar, and S. I. Ahamed, "Privacy-preserving self-supervised learning for secure image processing: A byol-based framework for mnist and chest x-ray data,"

[18] M. A. Barek, M. B. Rashid, M. M. Rahman, A. K. I. Riad, G. F. III, H. Shahriar, and S. I. Ahamed, "Vulnerability to stability: Scalable large language model in queue-based web service,"

[19] M. S. Akter, M. A. Barek, M. M. Rahman, A. K. I. Riad, M. A. Rahman, M. R. Mia, H. Shahriar, W. Chu, and S. I. Ahamed, "Hipaa technical compliance evaluation of laravel-based mhealth apps," in *2024 IEEE International Conference on Digital Health (ICDH)*, pp. 58–67, 2024.

[20] N. Islam, M. M. Shoaib Hasan, I. Hossain Shibly, M. B. Rashid, M. A. Yousuf, F. Haider, R. Ahmmed Aoni, and R. Ahmed, "Plasmonic sensor using generative adversarial networks integration," *Optics Express*, vol. 32, no. 20, pp. 34184–34198, 2024.