Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.4, August 2025, pp. 1272~1279
e-ISSN: 2548-6861

Static Analysis-Based Security Enhancement for Mobile Applications Using Mobile Security Framework (MOBSF)

Putri Nur Izzati [1]*, Kasmawi [2]*
* Program Studi Keamanan Sistem Informasi, Teknik Informatika, Politeknik Negeri Bengkalis
izzatiputri50@gmail.com [1], kasmawi@polbeng.ac.id [2]

Article Info

Article history:

Received 2025-04-28
Revised 2025-06-02
Accepted 2025-07-03

Keyword:

Mobile App Security,
Mobile Security Framework,
Static Analysis Mobsf,
Security Analysis,
Vulnerability Repair.

ABSTRACT

Mobile application security is crucial to protect users' personal data and maintain trust in the application. Without proper security testing, an app becomes vulnerable to threats such as data theft and cyber attacks. This study aims to identify and fix security vulnerabilities in the XYZ mobile application, a social platform used to report domestic violence and child sexual abuse cases. The analysis was conducted using static analysis with the Mobile Security Framework (MOBSF). The XYZ app was developed using Flutter and falls under the hybrid application category. Since it handles sensitive information from victims and reporters, ensuring its security is essential. The analysis revealed four major vulnerabilities with high risk levels, mainly related to misconfiguration and weak security settings. After addressing these issues, the app's security score improved from 37/100 (high risk) to 61/100 (low risk). These improvements were implemented in the final development phase before the app was released to users. MOBSF helped developers detect potential vulnerabilities early through static analysis, serving as a security baseline. This approach ensured the app no longer contained risks such as debug certificates, enabled debug mode, or support for outdated Android versions. The findings show that MOBSF-based security analysis is effective in detecting and reducing application security weaknesses, making the XYZ app more secure in protecting user data.

I. PENDAHULUAN

Analisis keamanan aplikasi mobile merupakan aspek penting dalam mengidentifikasi dan mengatasi potensi ancaman siber. Data yang dikumpulkan dalam proses ini dapat berupa log aktivitas pengguna, lalu lintas jaringan, serta catatan akses. Ancaman keamanan terus berkembang, sehingga metode tradisional yang hanya mengandalkan tanda-tanda ancaman yang sudah dikenal menjadi kurang efektif. Dengan analisis keamanan berbasis pola perilaku dan anomali, ancaman baru dapat diidentifikasi lebih cepat [1].

Keamanan aplikasi mobile menjadi krusial seiring meningkatnya penggunaan perangkat seluler, yang mencapai 90% populasi internet global. Hal ini menjadikan perangkat mobile sebagai target utama bagi peretas. Oleh karena itu, pengamanan aplikasi mobile harus ditingkatkan untuk melindungi data pengguna [2]. Mobile Security Framework (MOBSF) adalah tools open-source yang mampu melakukan uji penetrasi, analisis malware, serta penilaian keamanan aplikasi melalui analisis statis dan dinamis [3].

Aplikasi XYZ dikembangkan sebagai platform pelaporan tindak kekerasan dalam rumah tangga dan kekerasan seksual pada anak di Kabupaten Bengkalis. Keamanan aplikasi ini menjadi perhatian utama karena berkaitan dengan perlindungan identitas pelapor dan kepercayaan masyarakat terhadap sistem. MOBSF digunakan untuk mengidentifikasi kerentanan pada aplikasi XYZ, melakukan perbaikan, serta mengukur peningkatan tingkat keamanannya setelah dilakukannya perbaikan. Aplikasi XYZ termasuk dalam kategori aplikasi sosial yang berfungsi sebagai platform pelaporan kasus kekerasan dalam rumah tangga dan kekerasan seksual terhadap anak. Aplikasi ini memiliki fungsi krusial dalam perlindungan identitas korban dan pelapor, sehingga aspek keamanannya menjadi sangat penting demi

menjaga kerahasiaan informasi sensitif yang dikirimkan oleh pengguna.

Sejumlah penelitian sebelumnya juga menunjukkan penggunaan berbagai alat analisis keamanan aplikasi mobile selain MOBSF. AndroBugs digunakan untuk mengidentifikasi kelemahan berbasis kode statik dan konfigurasi aplikasi Android secara cepat [4]. QARK (Quick Android Review Kit) mempunyai kemampuan dalam mengidentifikasi celah terkait permission, intent, dan penggunaan API [5]. SonarQube sering dimanfaatkan untuk analisis kualitas kode dan kerentanan yang tersembunyi [6]. Meski demikian, dalam penelitian ini digunakanlah MOBSF karena memiliki kemampuan integratif untuk analisis statik dan dinamis secara lokal tanpa perlu mengunggah ke server eksternal, serta kompatibel dengan format APK yang dihasilkan Flutter.

Penelitian sebelumnya menunjukkan bahwa MOBSF dapat digunakan untuk analisis risiko malware dan deteksi karakteristik ancaman dalam aplikasi Android [7,8]. Berbeda dengan penelitian-penelitian sebelumnya yang umumnya hanya berfokus pada proses analisis keamanan aplikasi menggunakan MOBSF, penelitian ini tidak hanya melakukan identifikasi terhadap kerentanan keamanan, tetapi juga secara langsung melakukan proses perbaikan terhadap kerentanan yang ditemukan. Dengan demikian, penelitian ini memberikan kontribusi lebih lanjut berupa peningkatan nyata terhadap tingkat keamanan aplikasi yang diuji, dari kategori risiko tinggi menjadi risiko rendah. Pendekatan ini menunjukkan bahwa penggunaan MOBSF tidak hanya sebatas sebagai alat analisis, tetapi juga sebagai acuan teknis dalam proses hardening aplikasi sebelum dirilis ke publik, penelitian ini akan lebih fokus pada analisis dan peningkatan keamanan aplikasi XYZ setelah proses perbaikan kerentanan dilakukan. Hasil penelitian ini menunjukkan efektivitas MOBSF dalam menilai tingkat keamanan aplikasi setelah dilakukan perbaikan.

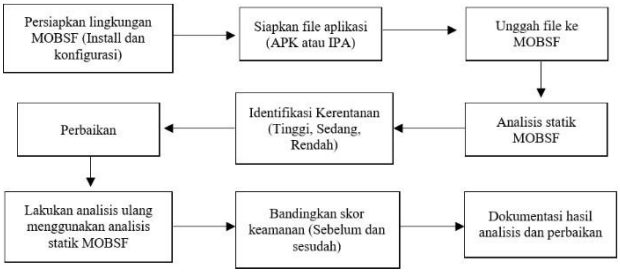
II. METODE

Tahapan dalam menganalisis keamanan aplikasi XYZ menggunakan Mobile Security Framework dapat dilihat pada Gambar 1.

Gambar 1. Tahap Penelitian

Proses analisis keamanan aplikasi menggunakan Mobile Security Framework (MOBSF) dimulai dengan instalasi dan konfigurasi lingkungan kerja. Setelah itu, file XYZ.apk diunggah ke MOBSF untuk analisis statik guna mengevaluasi izin, konfigurasi, dan potensi kerentanan keamanan dalam kode. Aplikasi XYZ dikembangkan menggunakan Flutter dan menghasilkan berkas APK dalam format release APK hybrid, yaitu gabungan dari elemen native dan web-based yang dijalankan pada satu container aplikasi. Format ini dipilih karena memungkinkan pengembangan lintas platform dengan satu basis kode, namun tetap memerlukan pengujian keamanan secara menyeluruh karena memuat komponen native yang bisa dieksploitasi apabila tidak dikonfigurasi dengan benar.

Kerentanan yang ditemukan dikategorikan berdasarkan tingkat keparahannya, lalu dilakukan perbaikan. Setelah perbaikan, aplikasi dianalisis ulang untuk menilai efektivitas langkah-langkah perbaikan yang diambil. Hasil analisis dibandingkan sebelum dan sesudah perbaikan, kemudian didokumentasikan untuk pelaporan. Penggunaan MOBSF pada penelitian dapat dilihat pada gambar 2.

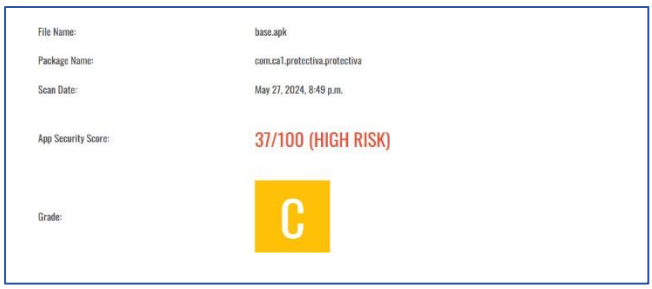


Gambar 2. Tahapan Mobile Security Framework MOBSF pada penelitian yang dilakukan

A. *Lingkungan Pengujian*

Dalam pengujian aplikasi XYZ, lingkungan pengujian yang digunakan melibatkan beberapa perangkat lunak dan perangkat keras yang mendukung proses analisis kerentanan. Pengujian aplikasi XYZ dilakukan menggunakan perangkat lunak Visual Studio Code sebagai platform pengeditan dan pengembangan kode, aplikasi XYZ sebagai objek pengujian keamanan. Analisis kerentanan dilakukan secara statik menggunakan Mobile Security Framework (MOBSF). MOBSF menghasilkan laporan analisis keamanan yang digunakan sebagai acuan untuk mempermudah perbaikan terhadap kerentanan yang ditemukan dalam aplikasi.

B. *Analisis statik MOBSF*



Gambar 3. Hasil Analisis Statik MOBSF

Gambar 3 menunjukkan hasil analisis aplikasi XYZ menggunakan Mobile Security Framework (MOBSF). Tingkat kerentanan aplikasi XYZ menunjukkan risiko tinggi (high risk), yang memungkinkan penyerang untuk melakukan serangan. Temuan kerentanan yang didapat dari analisis statik MOBSF termasuk dalam kategori Misconfiguration dan Weak Security Settings.

TABEL I
TEMUAN KERENTANAN DARI ANALISIS STATIK MOBSF

Temuan	**Tingkat Keparahan**	**Deskripsi**
Debug Configuration Enabled Production Bulids Must Note Be Debuggable	High	aplikasi masih menggunakan pengaturan debug, yang biasanya digunakan untuk pengujian saat pengembangan
Application Signed With debug certificate	High	Aplikasi ditandatangani menggunakan sertifikat debug, yang biasanya digunakan untuk pengujian selama pengembangan.
app can be installed on a vulnerable upatched android version android 5.0-5.0.2, [minSDK=21]	High	Aplikasi ini dapat diinstal pada android versi lama yang memiliki beberapa unfixed kerentanan.
Debug enabled for app [Android:debuggable=true]	High	peretas dapat memanfaatkan fitur ini untuk mengeksploitasi atau memodifikasi aplikasi.
Aplication Vulnerable To Janus Vulnerability	Warning	aplikasi rentan terhadap celah keamanan yang dikenal sebagai Janus Vulnerability.
Certificate Algorithm Might Be Vulnerable To Hash Collision	Warning	Algoritma yang digunakan dalam sertifikat aplikasi (seperti MD5 atau SHA-1) memiliki kelemahan yang memungkinkan dua data berbeda menghasilkan hash yang sama (hash collision).
application data can be backed up [android:allowBack up] flag is missing.	Warning	Aplikasi tidak memiliki pengaturan allowBackup dalam file konfigurasinya
Broadcast receiver (android.profileinstaller.profileinstallReceiver) is protected by a permission, but the protection level of the permission should be checked. permission: android.permision. DUMP [android:exported= true]	Warning	aplikasi memiliki Broadcast Receiver yang dapat menerima pesan dari sistem atau aplikasi lain.
files may contain hardcoded sensitive information like usernames, passwords, keys etc.	Warning	Aplikasi mungkin menyimpan informasi penting seperti nama pengguna, kata sandi, atau kunci API secara langsung dalam kode.
signed application	Info	aplikasitelah ditandatangani dengan sertifikat digital.
this app copies data to clipboard, sensitive data should note be coppied to clipboard as other applications can acess it.	Info	aplikasi menyalin data ke clipboard, yang dapat diakses oleh aplikasi lain yang berjalan di perangkat.
Malware Permissions	Izin Teratas Yang Banyak Disalah Gunakan Oleh Malware Terkenal.	Merujuk pada izin (permissions) yang diminta oleh aplikasi yang dapat digunakan untuk tujuan jahat, seperti malware.

Tabel 1 menunjukan kerentanan-kerentanan yang ditemukan dari proses analisis aplikasi XYZ menggunakan Mobile Security Framework. Temuan kerentanan yang didapatkan sebanyak 12 kerentanan diantaranya 4 dengan tingkat kerentanan high, 5 dengan kerentanan warning, 2 info dan 1 malware permissions.

C. Perbaikan Kerentanan

Perbaikan kerentanan dilakukan secara bertahap berdasarkan hasil analisis statik menggunakan Mobile Security Framework (MOBSF). Tahap pertama adalah mengidentifikasi jenis kerentanan yang memiliki tingkat keparahan tinggi (high risk), lalu dilakukan penelusuran terhadap sumber kerentanan dalam konfigurasi aplikasi, seperti pengaturan AndroidManifest.xml, build.gradle, serta sertifikasi penandatanganan aplikasi. Setelah ditemukan, perbaikan dilakukan melalui penyesuaian parameter dan konfigurasi yang relevan, seperti menonaktifkan mode debug, meningkatkan minimum SDK, serta menandatangani aplikasi dengan sertifikat rilis. Perbaikan diuji ulang menggunakan MOBSF untuk memastikan bahwa perbaikan tersebut berhasil mengurangi tingkat risiko.

Perbaikan kerentanan difokuskan pada temuan dengan tingkat keparahan High, di mana terdapat empat kerentanan utama yang berhasil diidentifikasi. Keempat kerentanan tersebut termasuk dalam kategori misconfiguration dan weak security settings, yang secara umum dapat menjadi pintu masuk awal bagi penyerang untuk mengeksploitasi sistem lebih lanjut.

1) Debug Configuration Enabled Production Bulids Must Note Be Debuggable

Pada kerentanan ini peretas dapat memanfaatkan fitur debug untuk mendapatkan informasi sensitif dan memanipulasi aplikasi.

Sebelum : Konfigurasi Debuggable Aktif pada AndroidManifest.xml

```
android:debuggable=”true”/>
```

Sesudah: Konfigurasi Debuggable nonaktif pada AndroidManifest.xml

```
Android:debuggable=”false”/>
```

Untuk menghindari peretas mendapatkan informasi sensitif dan memanipulasi aplikasi, maka perbaikan yang dilakukan adalah debug pada aplikasi harus di matikan pada file AndroidManifest.xml.

2) Debug enabled for app [Android:debuggable=true]

Kerentanan ini memungkinkan peretas dapat meretas aplikasi dengan alat debugging seperti android debug bridge

Sebelum : Konfigurasi Debuggable Aktif pada Build Tipe Release

```
    builtTypes {
    release {
    signingConfig signingConfigs.debug debuggable false
    }
    }
}
```

Sesudah: Konfigurasi Debuggable nonaktif pada Build Tipe Release

```
    builtTypes {
    release {
    signingConfig signingConfigs.debug debuggable false
    }
    }
}
```

Untuk menghindari kerentanan ini perbaikan yang dilakukan menggunakan build release yang tidak dapat di-debug dengan memastikan mode debug dinonaktifkan melalui pengaturan file build.gradle.

3) App can be installed on a vulnerable upatched android version android 5.0-5.0.2, [minSDK=21]

Kerentanan ini berkaitan dengan mendukung perangkat Android lawas yang sudah tidak aman. Kerentanan ini membuat penyerang dapat menggunakan kerentanan dalam sistem operasi untuk menyerang aplikasi.

Sebelum: Application ID Default pada File build.gradle

```
    defaultConfig {
    applicationID “com.example.protectiva” versionCode
flutterVersionCode.toInteger() versionName “1.0.2”
}
```

Sesudah : Konfigurasi Default Aplikasi dalam File build.gradle

```
    defaultConfig {
    applicationID “com.example.protectiva”
minSdkVersioan 29
    targetSdkVersion 34
    versionCode flutterVersionCode.toInteger()
versionName “1.0.2”
}
```

Perbaikan yang dilakukan adalah meningkatkan minSDKVersion ke versi yang lebih baru dengan mengkonfigurasi di file build.gradle.

4) Application Signed With debug certificate

Sebelum :

Tanpa Release Key

Sesudah :

Perbaikan yang dilakukan adalah membuat release keystore. aplikasi yang ditanda tangani dengan keystore unik dapat melindungi data pengguna dari manipulasi pihak ketiga. Release key juga dapat mencegah aplikasi di-repackage oleh pihak yang tidak berwenang dan hanya aplikasi yang ditandatangani dengan release keystore yang dapat diterima oleh platform distribusi aplikasi.

impor key.properties dalam file build.gradle:

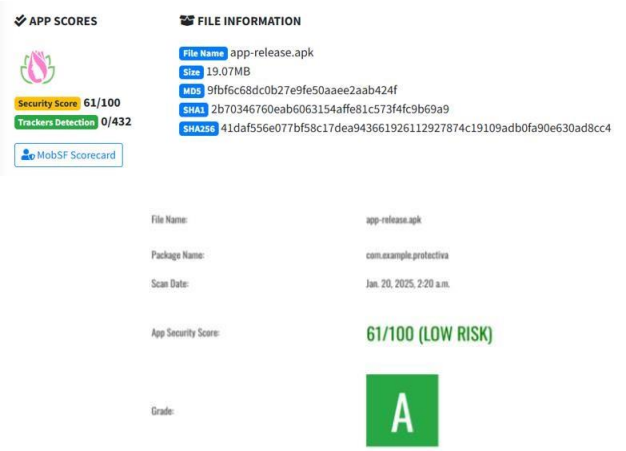
Konfigurasi File key.properties untuk Signing Android App

```
    def keystoreProperties = new
        Properties() def keystorePropertiesFile
=rootProject.file
("key.properties")if
        (keystorePropertiesFile.exists())
        { keystoreProperties.load(new
FileInputStream(keystorePropertiesFile)) }
```

Perbaikan yang dilakukan dari analisis menggunakan Mobile Security Framework (MOBSF) ditujukan untuk diterapkan pada tahap akhir pengembangan, yaitu tepat sebelum aplikasi diterbitkan atau didistribusikan kepada pengguna. Dengan menggunakan MOBSF sebagai detektor awal potensi kerentanan, pengembang aplikasi memiliki panduan untuk melakukan pemeriksaan keamanan yang menyeluruh melalui analisis statis. Perbaikan ini digunakan sebagai penerapan penilaian baseline keamanan pra-rilis (pre-release security baseline) yang sangat penting bagi pengembang, terutama untuk memastikan bahwa aplikasi tidak lagi mengandung risiko konfigurasi seperti penggunaan sertifikat debug, mode debug yang diaktifkan, dan dukungan untuk versi Android yang lebih lama dan rentan. Dengan perbaikan ini, jelas bahwa MOBSF tidak hanya berfungsi sebagai alat deteksi, tetapi juga membantu dalam pengambilan keputusan selama siklus pengembangan perangkat lunak yang aman (secured software development lifecycle).

D. Lanjutan analisis statik MOBSF

Setelah aplikasi diperbaiki berdasarkan kerentanan yang ditemukan melalui analisis statik menggunakan MOBSF, aplikasi dianalisis kembali dengan tool yang sama untuk memastikan bahwa perbaikan yang dilakukan efektif dan mampu meningkatkan tingkat keamanan aplikasi.

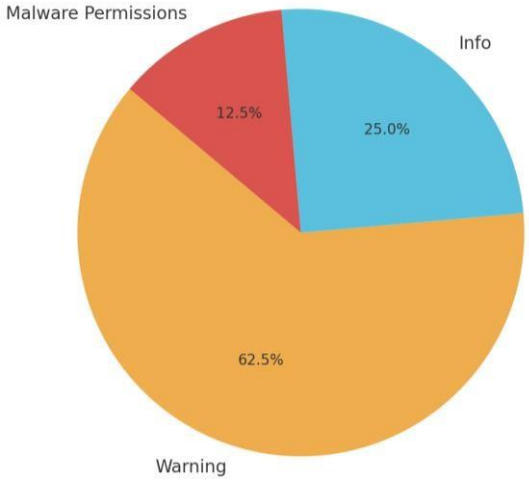


Gambar 4. Hasil Lanjutan Analisis Statik MOBSF

Gambar 4 menunjukan hasil analisis lanjutan aplikasi XYZ menggunakan mobile security framework (MOBSF) yang menunjukan keamanan pada aplikasi telah mencapai Tingkat keamanan low risk atau beresiko rendah. Ini berarti sebagian besar kerentanan termasuk dalam kategori misconfiguration dan weak security settings telah diperbaiki atau dianggap tidak terlalu berbahaya dalam konteks penggunaan aplikasi.

III. HASIL DAN PEMBAHASAN

Perbaikan yang dilakukan berfokus pada tingkat temuan High. Pada analisis lanjutan, skor keamanan aplikasi XYZ mencapai 61 dari 100, dengan tingkat risiko menurun menjadi Low.



Gambar 5. Distribusi Tingkat Kerentanan Setelah Perbaikan

Gambar 5 memperlihatkan distribusi tingkat kerentanan aplikasi setelah perbaikan dilakukan. Sebanyak 62,5% kerentanan berada pada kategori warning, sementara 25% berupa info, dan hanya 12,5% termasuk malware permissions. Tidak ditemukannya kerentanan high risk menandakan bahwa perbaikan berhasil menghilangkan celah kritis yang sebelumnya ditemukan.

Tingkat Low Risk ini menunjukkan bahwa meskipun ada beberapa kerentanan yang masih ditemukan, kerentanan tersebut tidak lagi dianggap kritis setelah beberapa kali perbaikan. Risiko Low ini berarti bahwa kemungkinan eksploitasi oleh pihak yang tidak bertanggung jawab sudah berkurang secara signifikan, walaupun belum sepenuhnya hilang.

Dampak dari perbaikan yang dilakukan menunjukkan peningkatan signifikan terhadap keamanan aplikasi XYZ. Sebelum dilakukan perbaikan, aplikasi berada pada tingkat risiko tinggi (high risk) dengan skor keamanan hanya 37/100. Setelah dilakukan penyesuaian konfigurasi dan perbaikan pada empat kerentanan utama, hasil analisis ulang menggunakan MOBSF menunjukkan bahwa tingkat risiko menurun menjadi low risk dengan skor 61/100. Perbaikan ini tidak hanya mengurangi kemungkinan eksploitasi terhadap komponen sensitif aplikasi, tetapi juga meningkatkan kepercayaan pengguna dan kesiapan aplikasi untuk dipublikasikan. Dengan demikian, perbaikan yang dilakukan memberikan dampak langsung terhadap peningkatan kualitas dan keamanan sistem, serta mendukung penerapan praktik pengembangan aplikasi yang lebih aman sejak tahap pra-rilis.

TABEL II
PERBANDINGAN SEBELUM DAN SESUDAH PERBAIKAN

Kerentanan	Kondisi Sebelum Perbaikan	Kondisi Setelah Perbaikan	Hasil Perbaikan
Debug Configuration Enabled Production Builds	Konfigurasi debug masih diaktifkan pada aplikasi produksi, meningkatkan risiko reverse engineering dan injeksi kode berbahaya.	Debug mode dinonaktifkan pada versi produksi untuk mencegah eksploitasi dan meminimalkan risiko keamanan.	Debugging pada aplikasi produksi berhasil dinonaktifkanse hingga mengurangi kemungkinan serangan berbasis reverse engineering.
Debug Enabled for App (debuggable= true)	Debugging diaktifkan sehingga memungkinkanpenyerang untuk menghubungkan debugger, melihat stack trace, dan mengakses data sensitif melalui debugging tools.	Properti debuggable diatur ke false untuk mencegah eksploitasi debugging dan melindungi data sensitif dalam aplikasi.	Debugging Tools tidak Lagi dapat mengakses aplikasi, meningkatkan keamanan terhadap eksploitasi dan analisis kode berbahaya.
Minimum SDK Version Rendah	Aplikasi dapat dijalankan pada perangkat dengan versi Android lama, yang rentan terhadap eksploitasi keamanan.	Minimum SDK versi diperbarui ke versi yang lebih tinggi untuk meningkatk an kompatibilit as dan mengurangi risiko eksploitasi pada perangkat lama.	Aplikasi kini hanya dapat diinstal pada perangkat dengan versi Android yang lebih aman, mengurangi risiko eksploitasi kerentanan sistem lama.
Signed with	Aplikasi ditandatangani	Aplikasi ditandatang ani	Penandatang anan dengan

Debug Certificate	dengan sertifikat debug, yang tidak aman untuk rilis produksi karena kemungkinka n modifikasi dan pemalsuan aplikasi.	dengan sertifikat rilis resmi untuk memastikan integritas aplikasi dan kepercayaan pengguna.	sertifikat rilis resmi memastikan aplikasi lebih sulit dimodifikasi atau dipalsukan.

Gambar 6. Peningkatan Skor Keamanan Aplikasi

Gambar 6 menunjukkan peningkatan skor keamanan aplikasi XYZ dari 37 (kategori high risk) menjadi 61 (kategori low risk) setelah dilakukan perbaikan terhadap konfigurasi dan sertifikasi aplikasi. Kenaikan ini menunjukkan efektivitas proses perbaikan dalam mengurangi potensi risiko keamanan.

Untuk mengklasifikasikan kerentanan yang ditemukan berdasarkan standar yang diakui secara internasional, hasil analisis dari MOBSF diinterpretasikan dengan mengacu pada OWASP Mobile Top Ten. Tujuannya adalah untuk memetakan temuan ke dalam sepuluh kategori utama risiko keamanan aplikasi mobile yang telah ditetapkan oleh OWASP, sehingga dapat memberikan pemahaman yang lebih terstruktur dan mendalam mengenai tingkat serta jenis risiko yang dihadapi oleh aplikasi.

TABEL III
INTERPRETASI HASIL ANALSIS BERDASARKAN OWASP MOBILE TOP TEN

OWASP Mobile Top Ten	Temuan Kerentanan dari MOBSF	Risiko	Rekomendasi Perbaikan
M1: Improper Platform Usage	Debug configuration enabled, application signed with debug certificate	Aplikasi dapat dieksploitasi untuk mengungkapkan informasi sensitif melalui fitur debug.	Nonaktifkan konfigurasi debug dalam aplikasi produksi, gunakan sertifikat rilis yang valid saat menandatanga ni aplikasi.
M2: Insecure Data Storage	App can be installed on a vulnerable, unpatched Android	Data pengguna berisiko berekspos pada perangkat yang tidak aman.	Tingkatkan versi minimum SDK untuk mendukung perangkat yang lebih amandan
	version with Min SDK 21		tambal kerentanan yang diketahui.
M3: Insecure Communica tion	Tidak ada temuan terkait	Tidak ada temuan terkait	Tidak ada temuan terkait
M4: Insecure Authenticati on	Tidak ada temuan terkait	Tidak ada temuan terkait	Tidak ada temuan terkait
M5: Insufficient Cryptograp hy	Tidak ada temuan terkait	Tidak ada temuan terkait	Tidak ada temuan terkait
M6: Insecure Authorizatio n	Tidak ada temuan terkait	Tidak ada temuan terkait	Tidak ada temuan terkait
M7: Client Code Quality	Debug configuration enabled, production builds must not be debuggable	Potensi manipulasi kode klien untuk memanfaatkan celah keamanan.	Nonaktifkan pengaturan debugging pada aplikasi produksi dan gunakan pengamanan tambahan seperti ProGuard atau R8 untuk mengaburkan kode aplikasi.
M8: Code Tampering	Tidak ada temuan terkait	Tidak ada temuan terkait	Tidak ada temuan terkait
M9: Reverse Engineering	Debug for enabled app	Aplikasi dapat dibongkar untuk mengakses kode sumber dan logika bisnis.	Terapkan obfuscation kode menggunakan ProGuard atau R8 untuk menyulitkan pembongkara n dan analisis aplikasi.
M10: Extraneous Functionalit y	App can be installed on a vulnerable, unpatched Android version	Fungsi yang tidak dimaksudkan untuk rilis produksi dapat terekspos.	Lakukan audit keamanan sebelum rilis untuk menghapus fungsi yang tidak diperlukan pada produksi.

Hasil penelitian ini menunjukkan bahwa proses perbaikan kerentanan berdasarkan analisis MOBSF mampu menurunkan tingkat risiko aplikasi dari kategori high risk menjadi low risk. Temuan ini sejalan dengan penelitian yang dilakukan oleh [7], yang juga menunjukkan efektivitas penggunaan MOBSF dalam mendeteksi celah keamanan pada aplikasi mobile. Namun, berbeda dengan penelitian tersebut yang hanya melakukan analisis tanpa tindak lanjut perbaikan, penelitian ini melangkah lebih jauh dengan menerapkan perbaikan pada konfigurasi teknis aplikasi. Hal ini juga melengkapi studi [8], yang menggarisbawahi pentingnya pengujian static untuk identifikasi awal malware, namun tidak menekankan pada proses hardening aplikasi. Oleh karena itu, kontribusi utama dari penelitian ini terletak pada implementasi langsung hasil analisis sebagai bagian dari strategi pengembangan aplikasi yang lebih aman sebelum rilis.

V. KESIMPULAN

Analisis terhadap aplikasi XYZ menggunakan mobile security framework dan perbaikan yang dilakukan terbukti mampu meningkatkan tingkat keamanan aplikasi, dari skor awal 37/100 (risiko tinggi) menjadi 61/100 (risiko rendah). Hasil ini menunjukkan bahwa perbaikan yang dilakukan efektif dalam mengatasi kerentanan yang ditemukan. Selain itu, Mobile Security Framework (MOBSF) terbukti menjadi alat pengujian yang efektif dalam mengidentifikasi kerentanan pada aplikasi XYZ. Berdasarkan perbaikan yang telah dilakukan, analisis keamanan menggunakan analisis statik Mobile Security Framework (MOBSF) memiliki potensi untuk dikembangkan lebih lanjut. Penelitian selanjutnya diharapkan dapat memanfaatkan metode analisis dinamis MOBSF untuk mengamati pola perilaku aplikasi serta memahami bagaimana aplikasi beroperasi dalam berbagai kondisi lingkungan, sehingga dapat memperkuat perlindungan terhadap potensi serangan secara lebih menyeluruh.

DAFTAR PUSTAKA

[1] F. Al Fajar, "Analisis Keamanan Aplikasi Web Prodi Teknik Informatika UIKA Menggunakan Acunetix Web Vulnerability", INOVA-TIF, vol. 3, no. 2, pp. 110–120, Dec. 2020.

[2] Fortinet (2023). What is mobile security? Mobile app security definition. https://www.fortinet.com/resources/cyberglossary/mobile-app-security

[3] Kurniawan, C., Trianto, N., Rekayasa,), Siber, K., Siber, P., & Negara, S. (2021). Security Assessment pada Aplikasi Mobile Android XYZ dengan Mengacu pada Kerentanan OWASP Mobile Top Ten 2016.

[4] Given. (N.D.). Upgrading And Expanding Androbugs To Address Emerging Vulnerabilities. https://ssrn.com/abstract=4600829

[5] Haris, M., Jadoon, B., Yousaf, M., & Hassan Khan, F. (2017). Evolution Of Android Operating System: A Review. www.apiar.org.au

[6] Lomio, F., Moreschini, S., & Lenarduzzi, V. (2021). Fault Prediction based on Software Metrics and SonarQube Rules. Machine or Deep Learning? http://arxiv.org/abs/2103.11321

[7] Erbeliza, S. (2023). Analisis Keamanan Aplikasi Mobile Commerce Menggunakan Mobile Security Framework (MOBSF) Dan OWASP Mobile Application Security Testing Guide (Mastg).

[8] Himawan, I., Septianzah, K., & Setiadi, I. (2023). Analisa Resiko Malware Dengan Static Mobsf Terhadap Aplikasi Android Apk. Technologia: Jurnal Ilmiah, 14(4), 364. https://doi.org/10.31602/tji.v14i4.11460

[9] OWASP, Top 10 Mobile Risks - OWASP Mobile Top 10 2024 - Final Release, [online] accessed on: 5 Juni 2024 dari https://owasp.org/www-project-mobile-top-10/2023-risks/

[10] Aan Kartono, Anang Sularsa, S. J. I. I. (2019). Membangun Sistem Pengujian Keamanan Aplikasi Android Menggunakan Mobsf. E-Proceeding of Applied Science, 5(1), 146.

[11] Tansen, E., & Wahyu Nurdiarto, D. (2020). Analisis Dan Deteksi Malware Dengan Metode Hybrid Analysis Menggunakan Framework Mobsf. Jurnal Teknologi Informasi, 4(2).

[12] Kadi, D. (2017). Pengembangan Aplikasi Mobile Objek Wisata Secara Real Time Dengan Augmented Reality Di Kabupaten Sumba Barat Daya. Uajy, 17–39. http://eprints.stainkudus.ac.id/192/5/5. BAB II.pdf

[13] AWS, Apa itu Analitik Keamanan?, [online] accessed on: 5 Juni 2024 dari https://aws.amazon.com/id/what-is/security-analytics/

[14] Digital Solusi Grup, Apa itu Application Security? Pengertian, Maksud, dan Pembahasannya!, accessed on: 5 Juni 2024 dari https://digitalsolusigrup.co.id/application-security-adalah/

[15] Wibowo, E. Y. A. (2019). Evaluasi Tata Kelola Keamanan Teknologi Informasi Menggunakan Framework Cobit 5 Dan Iso 27002" (Studi Kasus: Pusat Jaringan Komunikasi Badan Meteorologi Klimatologi Dan Geofisika. Repository.Uinjkt.Ac.Id, 1–585. http://repository.uinjkt.ac.id/dspace/handle/123456789/48133

[16] OWASP, Top 10 Mobile Risks - OWASP Mobile Top 10 2024 - Final Release, accessed on: 5 Juni 2024 dari https://owasp.org/www-project-mobile-top-10/2023-risks/

[17] Tansen, E., & Wahyu Nurdiarto, D. (2020). Analisis Dan Deteksi Malware Dengan Metode Hybrid Analysis Menggunakan Framework Mobsf. Jurnal Teknologi Informasi, 4(2).

[18] Putranda Muhammad Arrysatrya Yusuf Putrandaa1, I. K. A. M. (2024). Analisis Keamanan pada Aplikasi Udayana Mobile Mengacu pada OWASP Mobile Top 10 2016. Jurnal Elektronik Ilmu Komputer Udayana, 12(3).

[19] Aan Kartono, Anang Sularsa, S. J. I. I. (2019). Membangun Sistem Pengujian Keamanan Aplikasi Android Menggunakan Mobsf. E-Proceeding of Applied Science, 5(1), 146.

[20] Febriyan, D., #1, P., Hasbi, M., Surya, M., 3, M., Rekayasa, #, Siber, K., Siber, P., Negara, S., (2022). Security Assessment Aplikasi Mobile E-Kinerja dengan Acuan OWASP Top 10 Mobile Risks., 8(3).

[21] Anwar, C., Herli Sumerli A, C., Rahayu, N., & Kraugusteeliana, K. (2023). The Application of Mobile Security Framework (MOBSF) and Mobile Application Security Testing Guide to Ensure the Security in Mobile Commerce Applications. 5(2), 97–102. https://doi.org/10.37034/jsisfotek.v5i1.231

[22] Archibong, E. E., Stephen, B. U.-A., & Asuquo, P. (2024). Analysis of Cybersecurity Vulnerabilities in Mobile Payment Applications. Archives of Advanced Engineering Science. https://doi.org/10.47852/bonviewaaes42022595

[23] Gunawan Indra, & Yudatama Arya Kukuh. (2023). Analisis Keamanan Aplikasi Dompet Digital Pendekatan Statis dan Dinamis. 17.

[24] Nurindahsari, F., & Zen, B. P. (2021). Analisis Statik Keamanan Aplikasi Video Streaming Berbasis Android Menggunakan Mobile Security Framework (Mobsf) Security Static Analysis Of Android-Based Video Streaming Application Using Mobile Security Framework (Mobsf) (Vol. 4, Issue 2). Https://Databoks.Katadata.Co.Id

[25] Alanda, A., Satria, D., Mooduto, H. A., & Kurniawan, B. (2020). Mobile Application Security Penetration Testing Based on OWASP. IOP Conference Series: Materials Science and Engineering, 846(1). https://doi.org/10.1088/1757-899X/846/1/012036

[26] Chiboora, T. H., Chacha, L., Byagutangaza, T., & Gueye, A. (2023). Evaluating Mobile Banking Application Security Posture Using the OWASP's MASVS Framework. COMPASS 2023 - Proceedings of the ACM SIGCAS/SIGCHI Conference on Computing and Sustainable Societies, 99–106. https://doi.org/10.1145/3588001.3609367

[27] Mykhaylova, O., Fedynyshyn, T., Datsiuk, A., Fihol, B., & Hulak, H. (N.D.). Mobile Application As A Critical Infrastructure Cyberattack Surface.

[28] Holla, S., & Katti, M. M. (N.D.). Android Based Mobile Application Development And Its Security. International Journal Of Computer Trends And Technology. Http://Www.Internationaljournalssrg.Org

[29] Sachdeva, S., Jolivot, R., & Choensawat, W. (N.D.). Android Malware Classification Based On Mobile Security Framework.

[30] Torstensson, J. (2017). Android Security An Evaluation Of Applications In Google Play.

[31] Reddy Basireddy, M. (2024). Investigations Into Security Testing Techniques, Tools, And Methodologies For Identifying And Mitigating Security Vulnerabilities, Journal Of Artificial Intelligence, Machine Learning And Data Science. J Artif Intell Mach Learn & Data Sci, 2024(1), 626. Https://Doi.Org/10.51219/Jaimld/Maheswara

[32] Uskono, B. M., Wijaya, R., Galih Pradipta, M., & Kusnadi, A. (2021). Analisis Keamanan Aplikasi Fintech Di Indonesia: Studi Kasus OVO, GoPay, ShopeePay dan Dana. Journal of Information and Information Security (JIFORTY), 2(1), 177–186. http://ejurnal.ubharajaya.ac.id/index.php/jiforty