JISKA (Jurnal Informatika Sunan Kalijaga)
Vol. 11, No. 1, JANUARI, 2026, Pp. 13 – 31
ISSN: 2527 – 5836 (print) | 2528 – 0074 (online)

Evaluasi Keamanan OTP Firebase pada Aplikasi Android: Perbandingan SAST dan IAST dalam Identifikasi Kerentanan

I Gede Surya Rahayuda [(1)*], Ni Putu Linda Santiari [(2)]
[1] Departemen Informatika, Universitas Udayana, Bali, Indonesia
[2] Departemen Sistem Informasi, ITB STIKOM Bali, Bali, Indonesia
e-mail : igedesuryarahayuda@unud.ac.id, linda_santiari@stikom-bali.ac.id.
* Penulis korespondensi.
Artikel ini diajukan 28 Desember 2024, direvisi 24 Maret 2025, diterima 25 Maret 2025, dan dipublikasikan 25 Januari 2025.

Abstract

Application security is crucial for protecting user data from cyber threats, particularly in Android applications that utilize One-Time Password (OTP)-based authentication. This study evaluates the security of Firebase OTP via email using a combination of Static Application Security Testing (SAST) with Mobile Security Framework (MobSF) and Interactive Application Security Testing (IAST) with AppSweep. The results show that the combination of SAST and IAST is superior to single testing methods due to its wider detection coverage. SAST detects vulnerabilities in static code, while IAST identifies exploits in runtime. The testing showed significant improvements, with high-severity vulnerabilities decreasing from 3 cases in OTP-1 to zero in OTP-5, and the security score increasing from 43 (B) to 78 (A) in MobSF. Meanwhile, the number of vulnerabilities in AppSweep decreased from 14 to 9, with all high-severity vulnerabilities resolved. However, this study still has limitations, such as limited dataset coverage and potential bias from the testing tool. For further improvement, additional research can integrate artificial intelligence to automate vulnerability detection, as well as explore biometric-based authentication to enhance system security even further.

Keywords: Firebase OTP, MobS, AppSweep, SAST, IAST

Abstrak

Keamanan aplikasi sangat penting untuk melindungi data pengguna dari ancaman siber, terutama dalam aplikasi Android yang menggunakan autentikasi berbasis One-Time Password (OTP). Penelitian ini mengevaluasi keamanan OTP Firebase melalui email menggunakan kombinasi *Static Application Security Testing* (SAST) dengan Mobile Security Framework (MobSF) dan *Interactive Application Security Testing* (IAST) dengan AppSweep. Hasil penelitian menunjukkan bahwa kombinasi SAST dan IAST lebih unggul dibandingkan metode pengujian tunggal karena cakupan deteksi yang lebih luas. SAST mendeteksi kelemahan dalam kode statis, sementara IAST mengidentifikasi eksploitasi dalam runtime. Pengujian menunjukkan perbaikan signifikan, di mana kerentanan tingkat tinggi berkurang dari 3 kasus pada OTP-1 menjadi nol pada OTP-5, dengan skor keamanan meningkat dari 43 (B) menjadi 78 (A) dalam MobSF. Sementara itu, jumlah kerentanan dalam AppSweep menurun dari 14 menjadi 9, dengan semua kerentanan tingkat tinggi terselesaikan. Namun, penelitian ini masih memiliki keterbatasan, seperti cakupan dataset yang terbatas dan potensi bias dari alat pengujian. Untuk perbaikan lebih lanjut, penelitian selanjutnya dapat mengintegrasikan kecerdasan buatan untuk otomatisasi deteksi kerentanan serta mengeksplorasi autentikasi berbasis biometrik guna meningkatkan keamanan sistem lebih lanjut.

Kata Kunci: OTP Firebase, MobSF, AppSweep, SAST, IAST

1. PENDAHULUAN

Keamanan aplikasi telah menjadi tantangan krusial dalam pengembangan perangkat lunak, terutama untuk aplikasi yang menangani data sensitif pengguna (Moon et al., 2023). Ancaman seperti pencurian data, akses tidak sah, dan manipulasi sistem terus meningkat di era digital yang semakin terhubung ini. Oleh karena itu, mekanisme autentikasi yang kuat sangat diperlukan untuk melindungi privasi pengguna. Salah satu metode yang banyak digunakan adalah OTP (One

Time Password), yang berfungsi untuk memverifikasi identitas pengguna dalam transaksi penting atau proses login.

Meskipun OTP dapat diimplementasikan melalui berbagai saluran, seperti SMS atau aplikasi pihak ketiga, OTP berbasis email yang menggunakan Firebase menawarkan kemudahan integrasi dan skalabilitas yang menarik untuk aplikasi Android (Asih & Hasibuan, 2023). Namun, implementasi OTP ini tidak bebas dari kerentanannya, baik pada sisi kode aplikasi, komunikasi jaringan, maupun konfigurasi sistem yang kurang optimal. Meskipun Firebase menyediakan platform yang kuat, efektivitas OTP sangat bergantung pada cara implementasi dan pengelolaan kode oleh pengembang.

Keamanan produk perangkat lunak secara umum juga menjadi perhatian penting, mencakup berbagai aspek seperti kerahasiaan, integritas, dan ketersediaan (Stanciu, 2023). Dalam penelitian ini, evaluasi terhadap OTP Firebase dilakukan dengan mempertimbangkan perbedaan mendasar dibandingkan metode OTP lainnya, seperti OTP berbasis SMS dan OTP melalui aplikasi pihak ketiga. Salah satu alasan utama adalah aspek keamanan, di mana OTP Firebase menawarkan integrasi yang lebih erat dengan ekosistem Google, tetapi masih memiliki potensi celah keamanan yang perlu dieksplorasi lebih dalam.

Studi ini menyoroti kelemahan dan keunggulan masing-masing metode melalui analisis keamanan yang lebih komprehensif, termasuk potensi serangan man-in-the-middle, phishing, dan eksploitasi API. Beberapa penelitian sebelumnya telah mengevaluasi keamanan OTP dalam berbagai konteks, seperti penggunaan algoritma Speck untuk enkripsi OTP guna mencegah serangan sniffing dan intercept nirkabel (Taqwim et al., 2021), serta optimasi autentikasi OTP menggunakan algoritma Random Forest untuk mendeteksi dan mencegah penipuan (Lubis & Riswanto, 2024). Selain itu, pendekatan OTP berbasis Time-Based One-Time Password (TOTP) juga telah diterapkan untuk meningkatkan perlindungan data digital (Wibawa et al., 2024).

Mengacu pada studi-studi tersebut, penelitian ini mengadopsi kombinasi metode *Static Application Security Testing* (SAST) dan *Interactive Application Security Testing* (IAST), karena keduanya saling melengkapi dalam mendeteksi celah keamanan secara lebih menyeluruh, dibandingkan hanya mengandalkan salah satu metode. Dengan pendekatan tersebut, penelitian ini memberikan kontribusi signifikan dalam menilai keamanan OTP Firebase secara lebih mendalam dibandingkan studi sebelumnya. Penelitian ini difokuskan pada implementasi OTP berbasis email dengan Firebase dalam aplikasi Android menggunakan Kotlin sebagai bahasa pemrograman. Kotlin menawarkan berbagai fitur yang meningkatkan keamanan aplikasi, seperti penanganan null yang eksplisit dan dukungan untuk coroutine. Meskipun demikian, implementasi OTP berbasis Firebase masih menghadapi beberapa masalah kerentanannya, yang dapat mempengaruhi keamanan aplikasi secara keseluruhan. Oleh karena itu, pengujian keamanan dengan pendekatan yang tepat menjadi sangat penting untuk memastikan bahwa aplikasi aman digunakan oleh pengguna.

Dalam konteks keamanan aplikasi seluler, berbagai standar dan teknik telah dikembangkan untuk memastikan keamanan dan privasi pengguna. OWASP (Open Worldwide Application Security Project) telah menetapkan standar industri seperti MASVS (Mobile Application Security Verification Standard) (Schleier, Holguera, Mueller, Willemsen, et al., 2024) dan MASTG (Mobile Application Security Testing Guide) (Schleier, Holguera, Mueller, & Willemsen, 2024), yang memberikan panduan komprehensif untuk pengujian dan pengembangan aplikasi seluler yang aman. Selain itu, penelitian tentang aplikasi seluler berbasis cloud (Chimuco et al., 2023) menyoroti pentingnya pemahaman taksonomi serangan, mekanisme keamanan, dan spesifikasi pengujian keamanan untuk ekosistem ini.

Metode autentikasi yang aman juga menjadi fokus utama dalam penelitian. Studi tentang sistem autentikasi perbankan online (Danthy et al., 2024) mengusulkan penggunaan Time-Bound Password untuk meningkatkan keamanan transaksi keuangan. Namun, penelitian lain (Ikram et al., 2025) mengungkapkan risiko keamanan dan privasi yang terkait dengan aplikasi autentikator

OTP seluler, menyoroti perlunya jaminan keamanan dan privasi yang lebih baik dalam aplikasi tersebut.

Deteksi dan pencegahan malware pada perangkat seluler Android juga merupakan area penelitian yang penting (Keteku et al., 2024). Berbagai teknik dan alat telah dikembangkan untuk mengidentifikasi dan mencegah malware, termasuk analisis statis dan dinamis. Penelitian tentang alat SAST (Static Application Security Testing) (Li et al., 2023; Smyčka, 2024) menunjukkan pentingnya evaluasi dan penerapan alat SAST yang efektif untuk mengidentifikasi kerentanan dalam kode sumber. Selain itu, platform analisis statis (Sonnekalb et al., 2023) dapat digunakan untuk memonitor tren keamanan dalam repositori dan mengidentifikasi potensi kerentanan.

Analisis statis ini penting karena kerentanan dapat muncul dari berbagai faktor, termasuk kelemahan pada kode sumber (Stanciu, 2023). Meskipun telah dilakukan penelitian bertahun-tahun, deteksi otomatis kerentanan dalam aplikasi seluler tetap menjadi tantangan. Alat SAST dapat mengidentifikasi kelemahan pada kode sumber atau kode yang dikompilasi tanpa menjalankan aplikasi, namun memiliki keterbatasan seperti deteksi kerentanan yang terbatas, kurangnya pembaruan, dan ketahanan terbatas terhadap teknik pengaburan (Pagano et al., 2023).

Pengujian keamanan aplikasi seluler juga melibatkan pendekatan dinamis (Sutter et al., 2024), yang memungkinkan analisis perilaku aplikasi saat dijalankan. Tinjauan literatur sistematis tentang analisis keamanan dinamis di Android (Sutter et al., 2024) menyoroti pentingnya alat instrumentasi dan pemantauan jaringan dalam mengidentifikasi kerentanan. Selain itu, penelitian tentang pengujian keamanan otomatis untuk aplikasi seluler (Nutalapati, 2023) menekankan peran toko aplikasi dalam memastikan keamanan dan kualitas aplikasi yang diinstal oleh pengguna.

Dalam beberapa tahun terakhir, penelitian tentang keamanan aplikasi seluler juga mencakup studi tentang operasi tersembunyi dan pencurian informasi pribadi (Lim et al., 2024). Studi ini menyoroti potensi penyalahgunaan fitur aksesibilitas Android dan perlunya langkah-langkah keamanan yang komprehensif untuk melawan eksploitasi tersebut. Selain itu, penelitian tentang kemajuan dalam pengujian keamanan (Pargaonkar, 2023) mencakup berbagai metodologi dan tren yang muncul, seperti integrasi DevSecOps dan pengujian keamanan berkelanjutan.

Terakhir, penelitian tentang evaluasi dan penerapan alat SAST (Smyčka, 2024) memberikan panduan sistematis untuk menerapkan alat SAST dalam praktik pengembangan perangkat lunak. Selain itu, penelitian tentang platform analisis statis (Sonnekalb et al., 2023) menawarkan solusi untuk memantau peringatan keamanan di seluruh riwayat versi repositori. Selain metode SAST, penelitian ini juga akan menggunakan metode IAST (Pargaonkar, 2023). Alat yang digunakan untuk SAST adalah Mobile-Security-Framework (MobSF), sementara untuk IAST menggunakan AppSweep. Dari berbagai alat yang tersedia untuk SAST, DAST, dan IAST, beberapa di antaranya meliputi eslint, mypy, trivy, trufflehog, pylint, super-linter, checkov, checkstyle, moose, semgrep, pmd, pypi-auto-scanner, error-prone, kube-bench, molecule, ansible-lint, git-secrets, clair, biome, kics, black, shellcheck, ruff, gitleaks, infer, SonarQube, CodeQL, Contrast, Horusec, Insider, SBwFSB, Toller, Ares, Q-Testing, Chimp, Monkey, ACVTool, TimeMachine, WCTester, Sapienz, Stoat, dan AndroTest (Li et al., 2023; Pargaonkar, 2023; Schleier, Holguera, Mueller, Willemsen, et al., 2024; Smyčka, 2024).

Pemilihan MobSF untuk SAST didasarkan pada keunggulannya sebagai alat open-source yang secara khusus dirancang untuk aplikasi mobile dan telah terbukti efektif serta terpercaya. Sementara itu, pemilihan AppSweep untuk IAST dipilih karena masih sangat sedikit sumber yang menyebutkan penggunaan alat ini untuk IAST, serta metode ini belum banyak diterapkan, terutama dalam konteks aplikasi mobile dan pemantauan kerentanannya secara real-time. Selain itu, AppSweep juga memiliki versi gratis yang mempermudah penerapannya dalam penelitian ini.

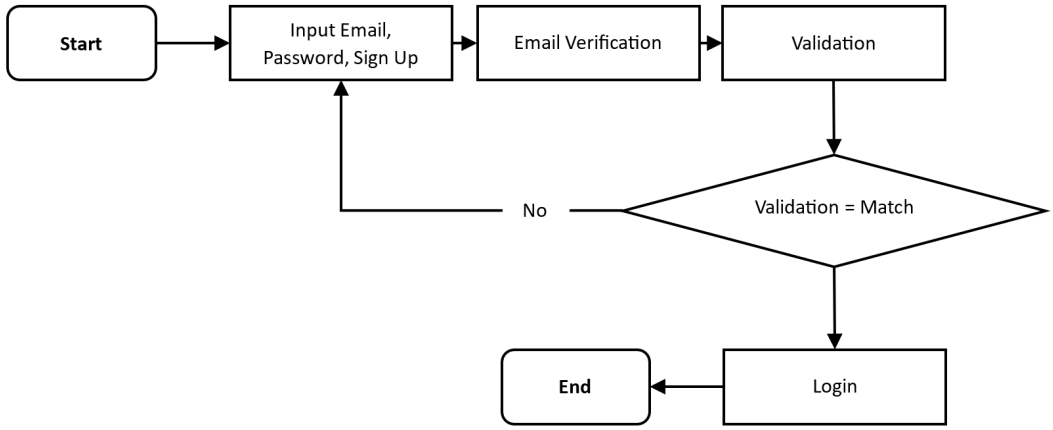
2. METODE PENELITIAN

Penelitian ini mengevaluasi keamanan implementasi OTP Firebase berbasis email pada aplikasi Android dengan menggunakan metode pengujian keamanan Static Application Security Testing (SAST) dan Interactive Application Security Testing (IAST). Pendekatan ini mencakup tiga aspek utama: implementasi OTP Firebase, pengujian SAST menggunakan Mobile Security Framework (MobSF), dan pengujian IAST menggunakan AppSweep.

2.1 OTP Firebase

Implementasi OTP berbasis email menggunakan Firebase Authentication memanfaatkan library Firebase untuk pengelolaan autentikasi. Firebase Authentication menyediakan antarmuka yang memungkinkan pengembang mengintegrasikan berbagai metode autentikasi, termasuk OTP, secara sederhana dan aman. Library dan modul yang digunakan adalah Firebase Authentication SDK yang digunakan untuk pengiriman dan validasi OTP dan Email Integration untuk mengirimkan kode OTP melalui email ke pengguna menggunakan layanan Firebase (Ikram et al., 2025; Lim et al., 2024).

Gambar 1 menunjukkan alur proses OTP, yang terdiri dari tiga tahap utama. Pertama, pada tahap permintaan OTP, aplikasi memanggil API Firebase untuk menghasilkan kode OTP dan mengirimkannya ke alamat email pengguna. Kedua, pada tahap validasi OTP, pengguna memasukkan kode OTP yang diterima, kemudian aplikasi melakukan verifikasi dengan Firebase untuk memastikan keabsahan kode tersebut. Terakhir, pada tahap manajemen token, setelah OTP berhasil diverifikasi, Firebase menghasilkan token autentikasi yang digunakan untuk menginisiasi sesi login pengguna.



Gambar 1 Alur OTP

2.2 SAST (Static Application Security Testing)

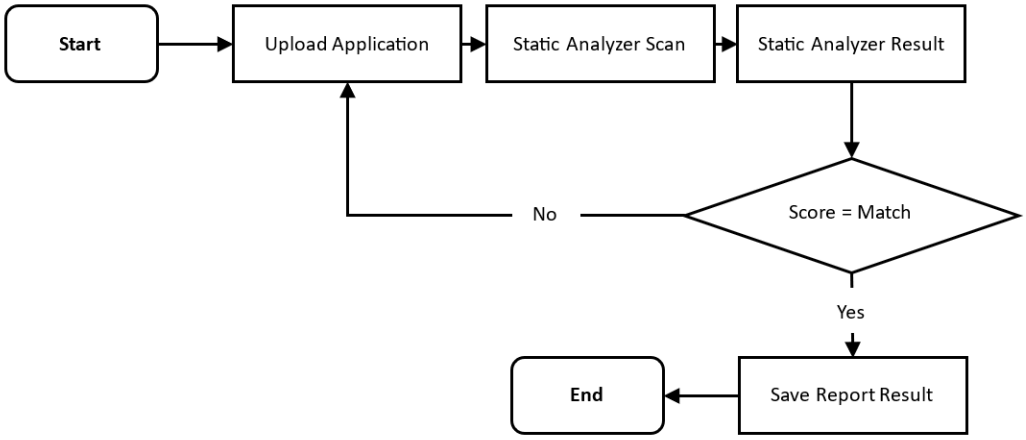
Static Application Security Testing (SAST) adalah metode pengujian keamanan yang menganalisis kode sumber aplikasi tanpa perlu menjalankan aplikasi tersebut. Pengujian ini biasanya dilakukan pada tahap awal pengembangan untuk mendeteksi kerentanan seperti penggunaan API yang tidak aman, *hardcoded credentials*, atau algoritma enkripsi yang lemah. Beberapa jenis SAST antara lain: Code Review Tools, yang mengidentifikasi masalah berdasarkan aturan tertentu seperti yang dilakukan oleh SonarQube; Binary Analysis Tools, yang menganalisis file biner seperti APK untuk menemukan kerentanan; serta Framework Analysis Tools, yang fokus pada analisis kerangka kerja yang digunakan, seperti Android SDK atau Firebase.

Dalam penelitian ini, Mobile Security Framework (MobSF) dipilih sebagai alat utama untuk pengujian SAST karena beberapa alasan. Pertama, MobSF mendukung analisis baik untuk file

APK (Android) maupun IPA (iOS), sehingga mampu menganalisis kode sumber dan juga file biner aplikasi. Kedua, MobSF memiliki fitur deteksi konfigurasi Firebase yang lemah, termasuk API Key yang terekspos atau akses database yang tidak aman. Ketiga, alat ini dapat diintegrasikan dengan proses CI/CD, memungkinkan otomatisasi pengujian keamanan dalam pipeline pengembangan. Keempat, sebagai solusi open-source yang andal, MobSF menjadi alternatif yang kuat dibandingkan alat berbayar seperti Checkmarx, Veracode, atau SonarQube (Nutalapati, 2023; Smyčka, 2024).

Gambar 2 memperlihatkan alur pengujian dengan MobSF, yang terdiri dari tiga tahap utama. Pertama, file APK diunggah ke antarmuka MobSF untuk dianalisis. Kedua, dilakukan analisis statis untuk menghasilkan laporan keamanan. Ketiga, hasil laporan dianalisis untuk mengidentifikasi kelemahan dan rekomendasi perbaikan yang diberikan oleh MobSF.

Sebagai alat pengujian keamanan aplikasi seluler, MobSF menawarkan berbagai fitur, termasuk Static Analysis yang menganalisis APK tanpa menjalankannya, Dynamic Analysis untuk menguji aplikasi pada emulator, dan Code Analysis untuk mendeteksi kelemahan dalam kode sumber aplikasi. Selain itu, MobSF juga melakukan pemindaian terhadap potensi kerentanan, seperti *hardcoded credentials* (misalnya API key atau password), komunikasi jaringan yang tidak aman (HTTP tanpa SSL), penggunaan *library* pihak ketiga yang rentan, serta kebijakan izin aplikasi yang terlalu luas. Kategori kerentanan yang dievaluasi oleh MobSF dapat dilihat pada Tabel 1, yang mengklasifikasikan tingkat keparahan kerentanan mulai dari *High*, *Medium*, hingga *Low*.



Gambar 2 Alur SAST dan MobSF

Tabel 1 Kategori Kerentanan MobSF

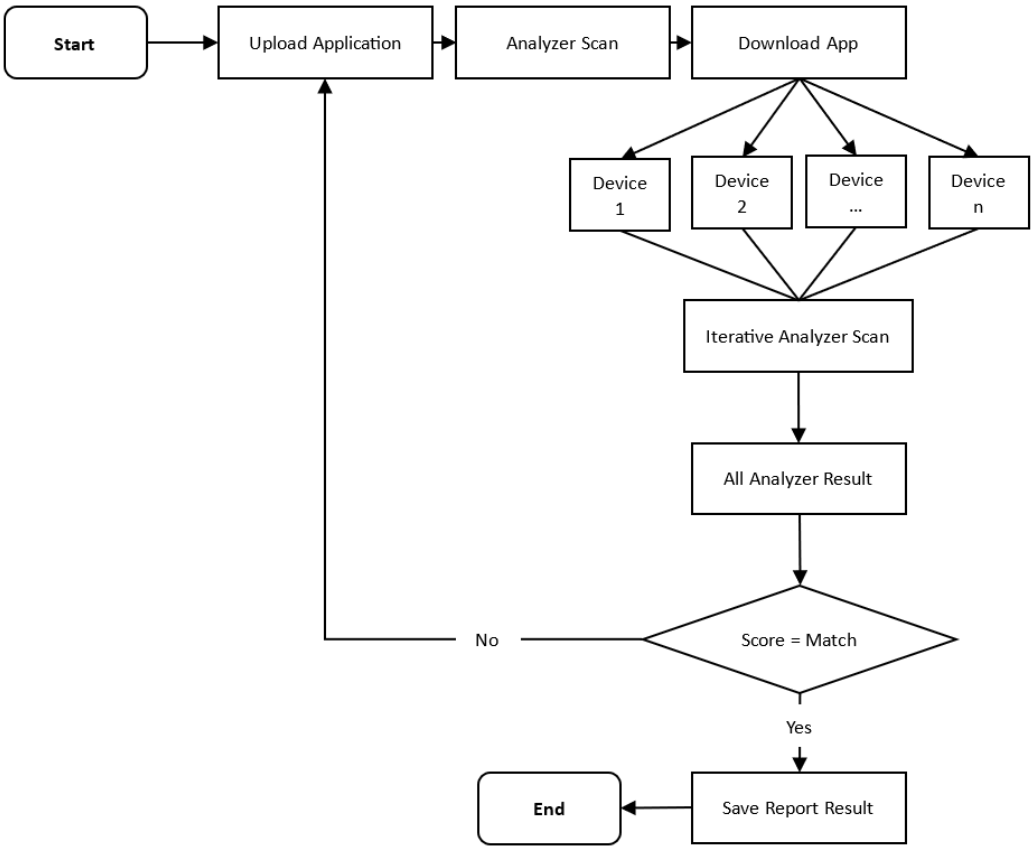
Tingkat Keparahan	Deskripsi
High	Kerentanan dengan tingkat risiko sangat tinggi yang dapat membahayakan aplikasi dan pengguna, sehingga perlu segera diperbaiki. Semakin tinggi tingkat risikonya, semakin besar potensi dampaknya.
Warning	Potensi risiko keamanan yang dapat menimbulkan masalah, namun tidak sekrusial tingkat High. Tetap perlu diperbaiki untuk mencegah dampak lebih lanjut. Semakin tinggi tingkat risikonya, semakin besar potensi dampaknya.
Info	Masalah minor yang tidak secara langsung mengancam keamanan, tetapi dapat memengaruhi kualitas aplikasi atau pengalaman pengguna. Semakin tinggi tingkat risikonya, semakin besar potensi dampaknya.
Secure	Hasil pengujian menunjukkan bahwa aplikasi memiliki perlindungan yang memadai dan tidak ditemukan kerentanan keamanan. Semakin tinggi nilainya, semakin baik tingkat keamanannya.

2.3 IAST (Interactive Application Security Testing)

Interactive Application Security Testing (IAST) adalah metode pengujian keamanan yang menggabungkan elemen dari pengujian statis (SAST) dan dinamis (DAST). Pengujian ini dilakukan saat aplikasi berjalan untuk memberikan wawasan mengenai bagaimana aplikasi merespons serangan keamanan secara langsung. Dalam penelitian ini, AppSweep dipilih sebagai alat utama karena keunggulannya dalam mendeteksi kerentanan aplikasi Android secara interaktif dan kontekstual. Alur proses pengujian IAST menggunakan AppSweep digambarkan pada Gambar 3.

AppSweep memiliki beberapa keunggulan dibandingkan alat IAST lainnya. Pertama, alat ini memiliki integrasi yang kuat dengan standar keamanan OWASP Mobile Top 10, memastikan bahwa pengujian berfokus pada kerentanan paling kritis dalam aplikasi seluler. Kedua, AppSweep melakukan analisis berbasis IAST yang memungkinkan deteksi kerentanan secara runtime, memberikan hasil yang lebih akurat dibandingkan metode statis (SAST) yang hanya menganalisis kode sumber tanpa menjalankannya (Li et al., 2023; Pargaonkar, 2023). Selain itu, AppSweep menyediakan automated security testing yang dapat dengan mudah diintegrasikan ke dalam alur kerja pengembangan (CI/CD pipeline) dan menghasilkan laporan yang komprehensif serta mudah dipahami.

AppSweep juga menjadi pilihan yang ekonomis karena menyediakan versi gratis, memungkinkan pengembang melakukan pengujian tanpa biaya tambahan. Dibandingkan dengan alat lain seperti Veracode IAST dan Contrast Security, AppSweep lebih fokus pada aplikasi mobile (Android), sedangkan dua alat lainnya umumnya digunakan untuk aplikasi web dan backend. Dengan demikian, AppSweep menjadi solusi yang fleksibel dan hemat biaya untuk mengamankan aplikasi Android, terutama pada tahap awal dan menengah pengembangan.



Gambar 3 Alur IAST Menggunakan Appsweep

Alur pengujian dengan AppSweep dijelaskan sebagai berikut:
- Unggah file APK ke AppSweep melalui antarmuka web atau plugin Android Studio.
- Instal dan jalankan aplikasi pada emulator atau perangkat nyata.
- Analisis laporan keamanan berdasarkan data yang dikumpulkan selama pengujian.

Fitur utama AppSweep meliputi pengujian pada emulator atau perangkat nyata, laporan berbasis OWASP Mobile Top 10, serta analisis izin aplikasi, penggunaan API, dan respons terhadap potensi serangan. Jenis-jenis kerentanan yang dianalisis AppSweep dirangkum pada Tabel 2 Kategori Kerentanan AppSweep, yang mencakup isu seperti data leakage, penyimpanan data tidak aman, ketidakamanan komunikasi jaringan, dan akses tidak sah ke API atau database.

Tabel 2 Kategori Kerentanan AppSweep

Tingkat Keparahan	Deskripsi
High	Masalah kritikal yang dapat membahayakan aplikasi dan data pengguna. Harus segera diperbaiki untuk mencegah potensi eksploitasi. Semakin tinggi tingkat risikonya, semakin berbahaya.
Medium	Kerentanan yang meningkatkan risiko keamanan aplikasi, tetapi tidak sebesar tingkat High. Perlu diperbaiki untuk mengurangi kemungkinan ancaman. Semakin tinggi tingkat risikonya, semakin berbahaya.
Low	Masalah dengan dampak kecil yang tidak langsung membahayakan, tetapi dapat memengaruhi kualitas atau kinerja aplikasi. Sebaiknya tetap diperbaiki untuk meningkatkan keamanan secara keseluruhan. Semakin tinggi tingkat risikonya, semakin berbahaya.
Issues	Jumlah total kerentanan yang terdeteksi, mencakup kategori High, Medium, dan Low.
OWASP	Klasifikasi masalah keamanan berdasarkan standar OWASP MASVS dan OWASP MASTG (Schleier, Holguera, Mueller, & Willemsen, 2024; Schleier, Holguera, Mueller, Willemsen, et al., 2024).
Libraries	Daftar kerentanan yang ditemukan dalam pustaka (libraries) yang digunakan dalam aplikasi. Merupakan penjabaran lebih detail dari issues.

Eksperimen dilakukan menggunakan dua emulator pada dua perangkat berbeda dengan spesifikasi sebagai berikut:

Perangkat 1 (PC - Pengembangan Kode):
- Prosesor Intel Core i3 generasi ke-10, RAM 8GB
- Sistem Operasi: Windows 11
- Android Emulator: Pixel 3a API 30 (Android 11)

Perangkat 2 (Laptop - Pengujian Keamanan):
- Prosesor Intel Core i3 generasi ke-12, RAM 8GB
- Sistem Operasi: Windows 11
- Android Emulator: Pixel 3a API 30 (Android 11)
- Pengujian dilakukan dari lokasi geografis berbeda menggunakan AppSweep

Dalam analisis keamanan aplikasi, terdapat kemungkinan terjadinya **false positives** (kerentanan terdeteksi padahal tidak ada) dan **false negatives** (kerentanan tidak terdeteksi). Untuk meminimalkan hal ini, dilakukan beberapa langkah:
- **Cross-validation:** hasil dari MobSF (SAST) dan AppSweep (IAST) dibandingkan untuk mendeteksi ketidaksesuaian hasil.
- **Manual review:** verifikasi manual terhadap hasil deteksi untuk memastikan akurasi laporan.
- **Re-run testing:** pengujian diulang beberapa kali dengan versi APK yang berbeda untuk memeriksa konsistensi hasil.

Perbandingan antara MobSF dan AppSweep dalam mendeteksi kerentanan dirangkum pada Tabel 3, yang menunjukkan bahwa kedua alat memiliki karakteristik pelengkap — MobSF unggul dalam mendeteksi kelemahan pada konfigurasi dan kode sumber, sedangkan AppSweep lebih efektif dalam mendeteksi kerentanan yang muncul selama runtime.

Tabel 3 Perbandingan MobSF dan AppSweep dalam Deteksi Kerentanan

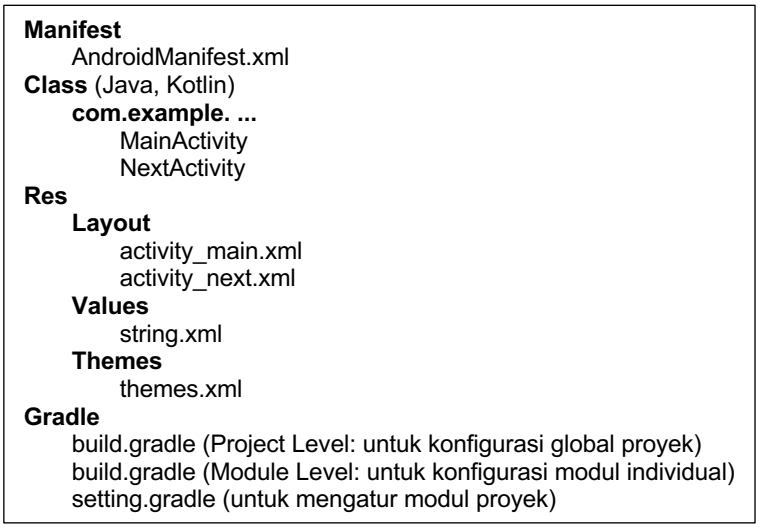
Jenis Kerentanan	MobSF (SAST)	AppSweep (IAST)
Hardcoded Credentials	Ya	Tidak
Insecure API Usage	Ya	Ya
Improper Cryptography	Ya	Tidak
Data Leakage	Tidak	Ya
Insecure Storage	Tidak	Ya
Insecure Communication	Ya	Ya
Runtime Exploitation	Tidak	Ya

3. HASIL DAN PEMBAHASAN

Pada bagian ini, akan dibahas secara rinci hasil implementasi dan pengujian keamanan dari aplikasi OTP Firebase berbasis email. Pembahasan mencakup tiga aspek utama, yaitu: (1) implementasi fitur OTP menggunakan Firebase Authentication, (2) pengujian keamanan aplikasi secara statis (SAST) menggunakan Mobile Security Framework (MobSF), dan (3) pengujian keamanan secara interaktif (IAST) menggunakan AppSweep. Setiap bagian dijelaskan secara sistematis dengan menampilkan potongan kode program, gambar tangkapan layar dari lingkungan pengembangan dan hasil pengujian, serta laporan analisis keamanan yang dihasilkan oleh masing-masing alat. Melalui pendekatan ini, diharapkan dapat memberikan pemahaman menyeluruh mengenai bagaimana setiap metode pengujian bekerja, jenis kerentanan yang terdeteksi, serta rekomendasi perbaikan yang dihasilkan dari proses analisis.

3.1 OTP Firebase

Pengujian implementasi OTP Firebase dilakukan dengan menggunakan Firebase Authentication untuk mengirimkan OTP melalui email dan memverifikasi kode yang diterima. Proses ini memerlukan pemahaman tentang bagaimana Firebase menangani autentikasi pengguna dan bagaimana kode implementasi dapat berfungsi dalam aplikasi Android. Untuk mengimplementasikan OTP menggunakan Firebase, struktur file aplikasi Android, yang ditunjukkan pada Gambar 4, terdiri dari beberapa komponen yang berfungsi untuk mengirimkan dan memverifikasi OTP.



Gambar 4 Struktur File Aplikasi Android

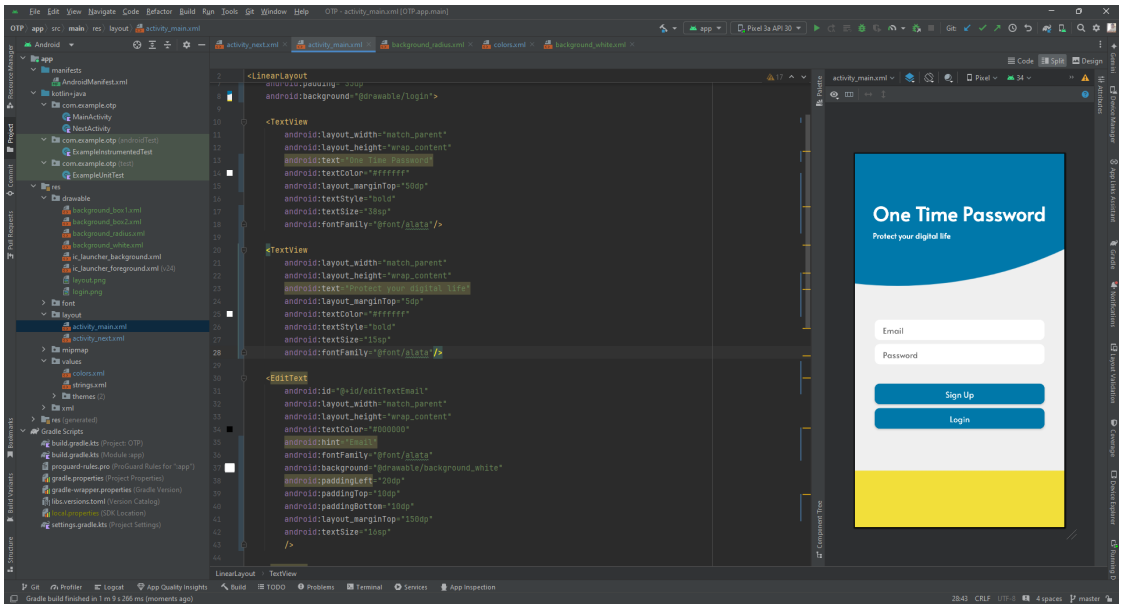
JISKA (Jurnal Informatika Sunan Kalijaga)
Vol. 11, No. 1, JANUARI, 2026: 13 – 31

Pada Gambar 5 menunjukkan potongan kode untuk mengirim dan memverifikasi OTP melalui email. Kode Tersebut merupakan bagian dari aplikasi Android yang dituliskan menggunakan Kotlin dan dapat dilihat pada repositori berikut: https://github.com/rahayuda/OTP-Evaluation.git. Pada potongan kode tersebut menggunakan dua fungsi utama, yaitu sendOTP dan verifyOTP. Fungsi sendOTP berfungsi untuk mengirimkan OTP ke alamat email pengguna melalui layanan FirebaseAuth, sedangakan fungsi verifyOTP digunakan untuk memverifikasi OTP yang dimasukkan oleh pengguna untuk proses autentikasi. Gambar 6 menunjukkan tampilan dashboard pada Android Studio yang digunakan sebagai lingkungan pengembangan aplikasi ini.

```
// Mengirim OTP melalui email
fun sendOTP(email: String) {
    FirebaseAuth.getInstance().sendSignInLinkToEmail(email, actionCodeSettings)
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                println("OTP dikirim ke $email")
            } else {
                println("Gagal mengirim OTP: ${task.exception?.message}")
            }
        }
}

// Memverifikasi OTP
fun verifyOTP(email: String, otp: String) {
    FirebaseAuth.getInstance().signInWithEmailLink(email, otp)
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                println("Login berhasil!")
            } else {
                println("Validasi OTP gagal: ${task.exception?.message}")
            }
        }
}
```

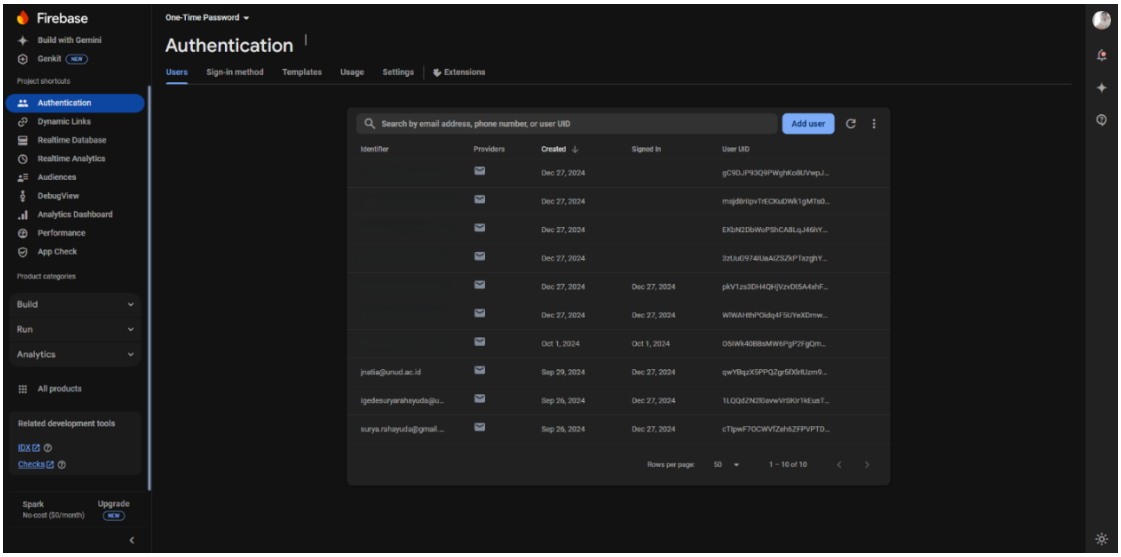
Gambar 5 Kode Pemrograman Pengiriman OTP



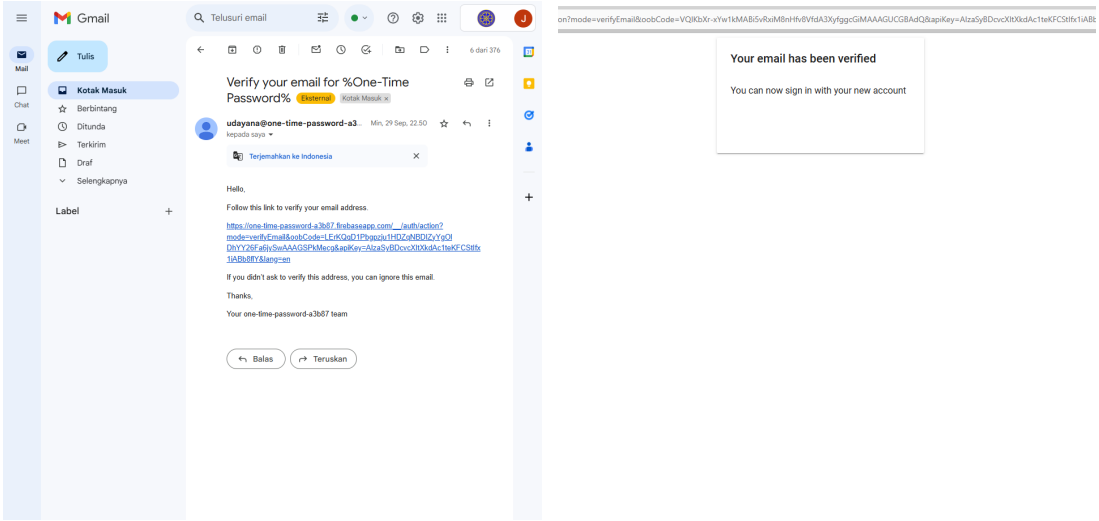
Gambar 6 Android Studio Dashboard

Untuk memverifikasi implementasi OTP, dilakukan konfigurasi di halaman Firebase Authentication. Firebase menyediakan antarmuka yang memungkinkan pengembang melihat status autentikasi pengguna, termasuk pengguna yang telah berhasil melakukan verifikasi email. Gambar 7 menunjukkan tangkapan layar dari halaman Firebase Authentication, yang menampilkan daftar pengguna yang terdaftar dan telah terverifikasi.

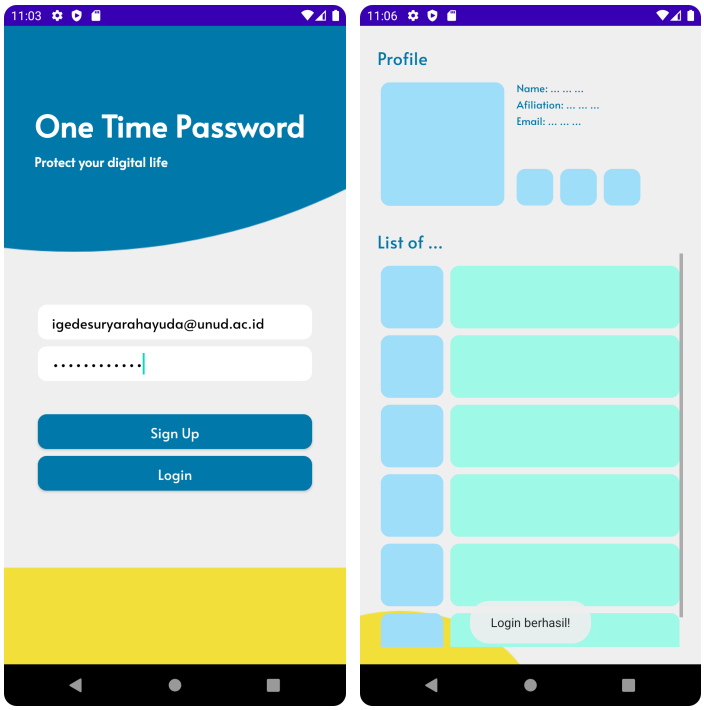
Jika akun pengguna belum terdaftar, pengguna tidak dapat melakukan login. Proses pendaftaran dilakukan dengan memasukkan email yang valid, setelah itu Firebase mengirimkan email verifikasi kepada pengguna. Pengguna kemudian harus memverifikasi email tersebut untuk menyelesaikan pendaftaran. Gambar 8 menampilkan tangkapan layar email verifikasi yang dikirim serta status verifikasi yang berhasil dilakukan. Setelah terdaftar dan berhasil memverifikasi email, pengguna dapat melakukan login dan diarahkan ke halaman profil. Sebaliknya, jika akun belum terdaftar atau email belum diverifikasi, proses login akan gagal. Gambar 9 menunjukkan tangkapan layer halaman login dan halaman profil pengguna setelah berhasil login.



Gambar 7 Firebase Authentication



Gambar 8 Verifikasi Email



Gambar 9 Halaman Login dan Profil

3.2 SAST (Static Application Security Testing)

Analisis Keamanan Statis (SAST) dilakukan menggunakan Mobile Security Framework (MobSF) untuk menganalisis file APK aplikasi secara statis. MobSF memindai aplikasi sebelum dijalankan untuk mendeteksi kerentanan yang dapat dieksploitasi oleh pihak yang tidak bertanggung jawab. Pengujian ini sangat penting untuk mengidentifikasi potensi masalah keamanan pada tahap awal pengembangan, sehingga dapat diatasi sebelum aplikasi dirilis ke publik. Tabel 4 menunjukkan hasil pemindaian dari lima percobaan yang dilakukan pada aplikasi OTP, dengan berbagai tingkat kerentanannya yang terdeteksi oleh MobSF.

Tabel 4 Evaluasi OTP App dengan MobSF

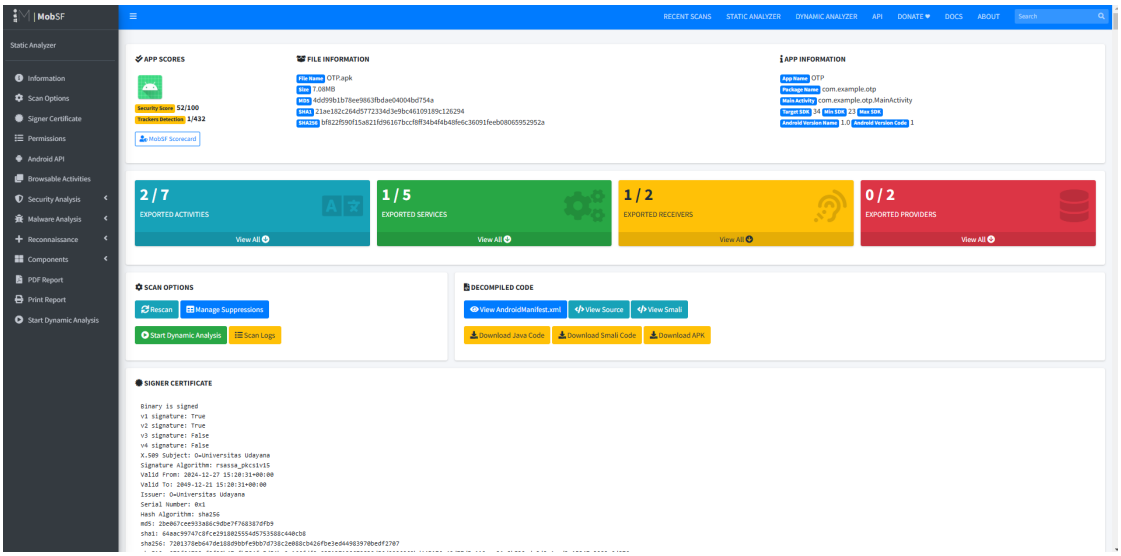
Evaluation	High	Warning	Info	Secure	Score	Grade	Report
OTP-1	3	8	1	1	43	B	OTP-1.pdf
OTP-2	1	8	1	1	52	B	OTP-2.pdf
OTP-3	0	6	1	1	60	A	OTP-3.pdf
OTP-4	0	3	1	1	67	A	OTP-4.pdf
OTP-5	0	3	1	2	78	A	OTP-5.pdf

Hasil pengujian menunjukkan adanya variasi kerentanan pada aplikasi OTP yang bervariasi, mulai dari tingkat tinggi (High) hingga kategori yang lebih ringan. Pada percobaan awal (OTP-1), ditemukan tiga temuan kategori "High", yang menunjukkan bahwa aplikasi masih sangat rentan pada tahap awal pengujian. Namun, setelah dilakukan rangkaian perbaikan, jumlah temuan kerentanan menurun secara signifikan pada percobaan berikutnya. Percobaan OTP-5 bahkan mencapai skor tertinggi yaitu 78 dengan peringkat "A", seperti ditunjukkan pada Gambar 10 (MobSF Score pada OTP-2) dan Gambar 11 (MobSF Score pada OTP-5).

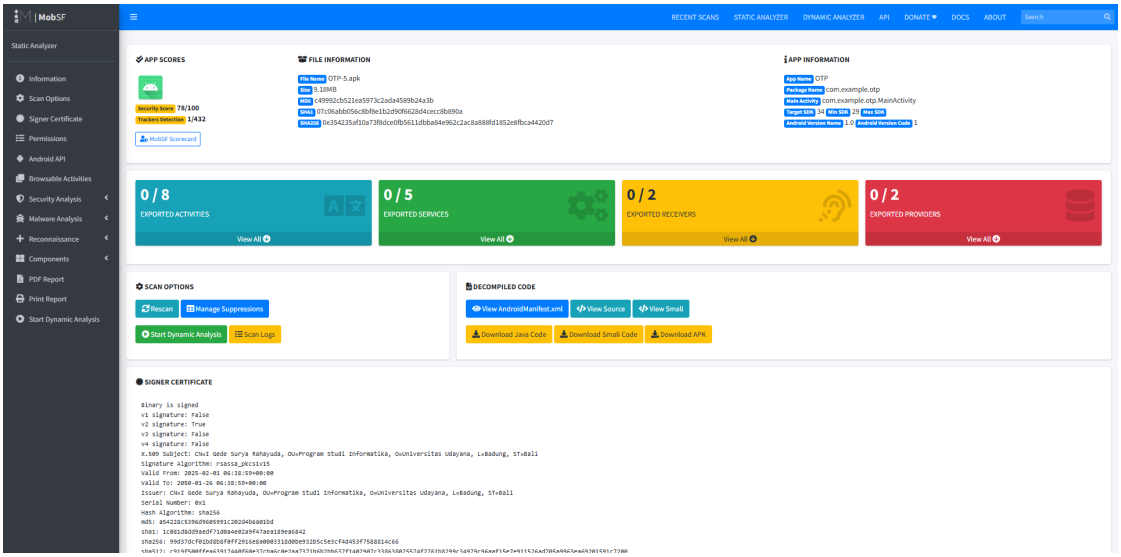
Kerentanan tingkat tinggi yang terdeteksi pada pengujian pertama dan kedua (OTP-1 dan OTP-2) berpotensi mengekspos data pengguna atau menyebabkan kerusakan fungsional pada aplikasi. Oleh karena itu, temuan ini harus segera diperbaiki agar menghindari potensi eksploitasi dan dampak serius terhadap keamanan sistem. Selain itu, MobSF juga mendeteksi sejumlah

peringatan dan informasi yang terdeteksi. Namun, temuan ini tidak terlalu kritikal dan dapat diabaikan karena tidak memberikan dampak signifikan terhadap keamanan aplikasi. Fokus utama dalam perbaikan seharusnya adalah menghilangkan masalah dengan kerentanan kategori "High" yang berpotensi lebih membahayakan.

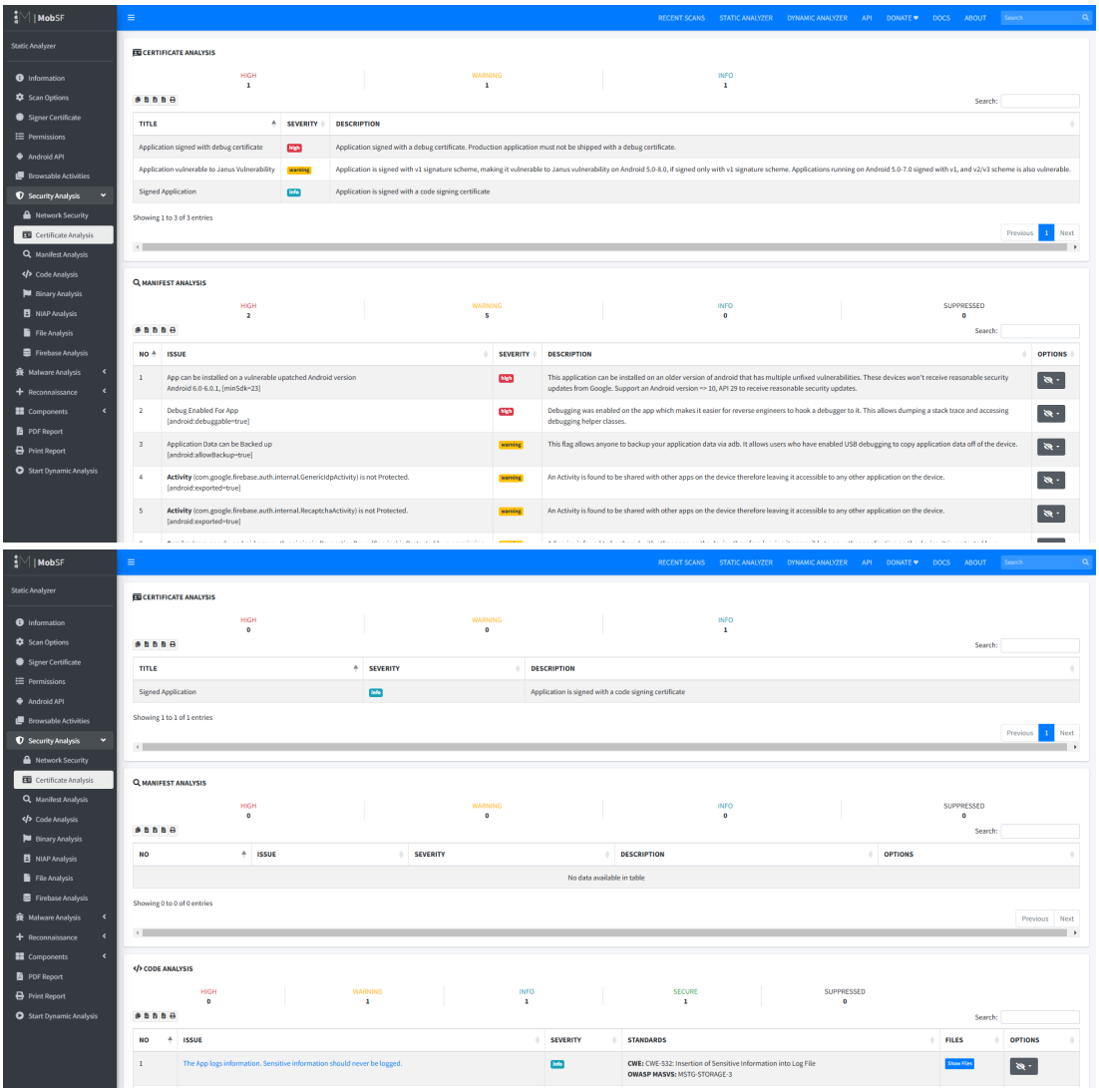
Secara keseluruhan, hasil pengujian ini menunjukkan bahwa meskipun aplikasi OTP telah mengalami peningkatan dari waktu ke waktu, potensi kerentanannya tetap perlu menjadi perhatian utama, terutama karena aplikasi ini menangani data sensitif seperti kode OTP. Kerentanan tingkat tinggi, seperti penggunaan pengkodean yang kurang aman atau manajemen kunci yang tidak terlindungi dengan baik, dapat membuka celah bagi serangan yang mengancam integritas dan keamanan aplikasi. Implikasi praktis dari temuan ini adalah pentingnya penerapan keamanan secara menyeluruh dalam tahap pengembangan serta pelaksanaan pemindaian keamanan secara berkala setelah aplikasi dirilis, dengan fokus utama pada penghapusan kerentanan kategori "High".



Gambar 10 Mobile Security Framework (MobSF) Score pada OTP-2



Gambar 11 Mobile Security Framework (MobSF) Score pada OTP-5



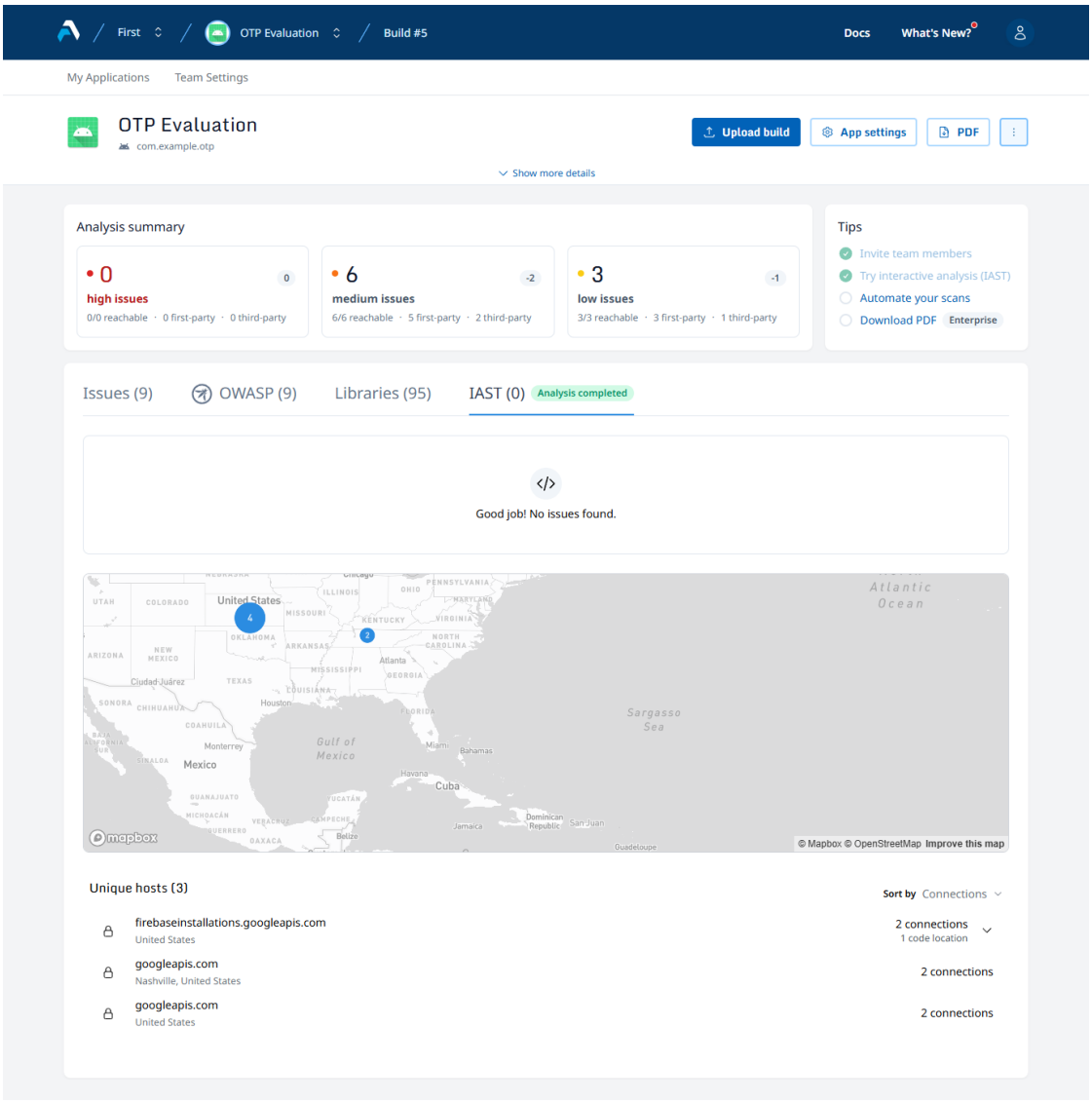
Gambar 12 Perbandingan High Issues pada OTP-1 dan OTP-5 MobSF

3.3 IAST (Interactive Application Security Testing)

Pengujian Interactive Application Security Testing (IAST) dilakukan menggunakan AppSweep untuk mengevaluasi keamanan aplikasi saat dijalankan di perangkat atau emulator. AppSweep membantu memeriksa berbagai masalah keamanan seperti kebocoran data, kesalahan konfigurasi izin, dan penggunaan API yang tidak aman, dengan mengacu pada standar OWASP Mobile Top 10. Hasil pengujian ditampilkan dalam berbagai halaman seperti Issues, OWASP, Libraries, dan laporan hasil pemindaian IAST. Gambar 13 memperlihatkan tampilan halaman IAST pada percobaan OTP-5, sedangkan Tabel 5 menyajikan hasil evaluasi keseluruhan dari lima percobaan aplikasi OTP yang telah dilakukan.

Tabel 5 Evaluasi OTP App dengan AppSweep

Evaluation	High	Medium	Low	Issues	OWASP	Libraries	Issues	Grade
OTP-1	2	8	4	14	14	125	0	Good
OTP-2	1	8	4	13	13	126	0	Good
OTP-3	0	8	4	12	12	125	0	Good
OTP-4	0	8	4	12	12	125	0	Good
OTP-5	0	6	3	9	9	95	0	Good

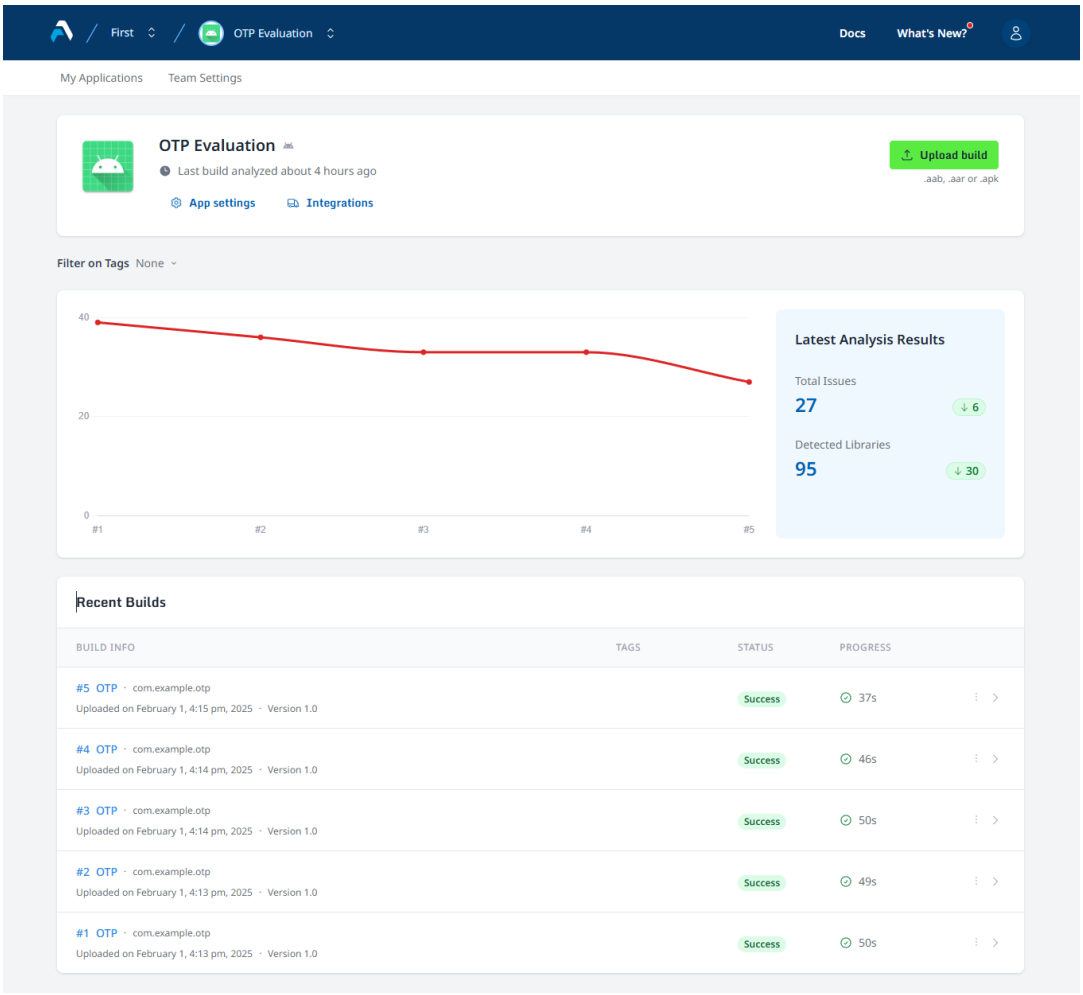


Gambar 13 Halaman IAST pada OTP-5

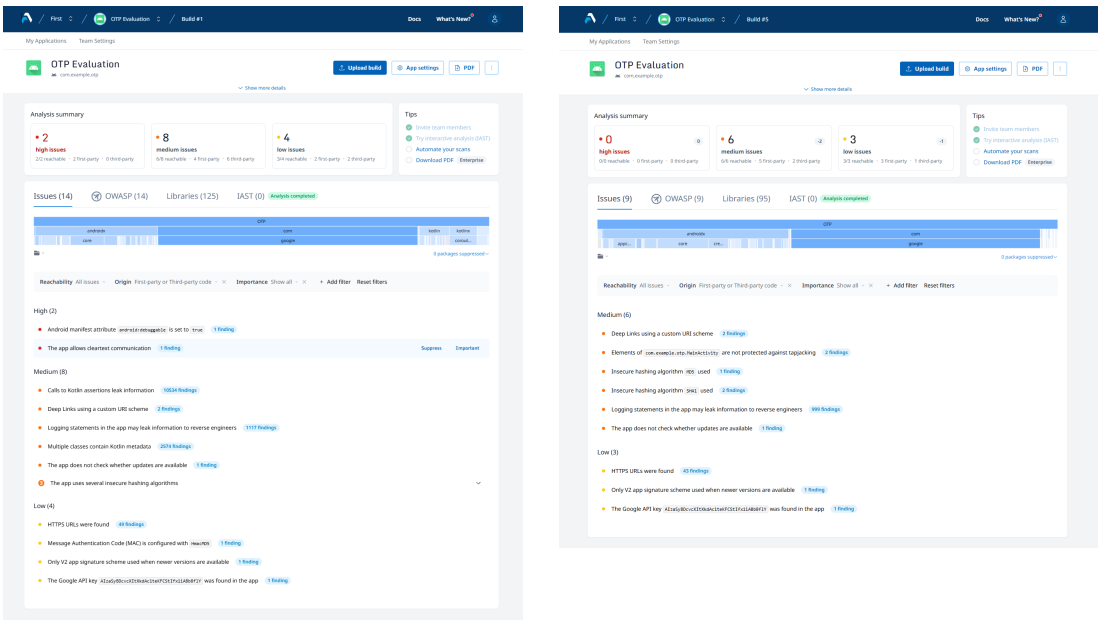
Berdasarkan hasil pengujian, aplikasi OTP secara keseluruhan memperoleh peringkat "Good" dalam hal kerentanannya. Tidak ada temuan yang masuk dalam kategori "Critical", yang menunjukkan bahwa aplikasi memiliki tingkat keamanan yang relatif baik. Namun, pada percobaan pertama dan kedua (OTP-1 dan OTP-2), terdapat temuan dalam kategori High yang mencakup potensi kebocoran data dan masalah izin yang signifikan. Kerentanan ini dapat memberikan akses yang tidak sah ke data sensitif atau memberikan hak akses yang salah kepada pengguna atau aplikasi pihak ketiga.

Seiring berjalannya pengujian, hasil menunjukkan peningkatan yang signifikan pada percobaan OTP-3 hingga OTP-5, di mana tidak ada lagi temuan dalam kategori High. Hal ini menunjukkan bahwa langkah-langkah perbaikan yang dilakukan berhasil mengatasi kerentanannya yang lebih serius. Secara kuantitatif, hasil pengujian menunjukkan bahwa jumlah kerentanannya berkurang secara signifikan dari 14 masalah pada OTP-1 menjadi 9 masalah pada OTP-5, seperti yang ditunjukkan pada Gambar 14. Hal ini menandakan kemajuan yang jelas dalam mengurangi jumlah dan tingkat kerentanannya. Sementara itu, Gambar 15 menggambarkan perbandingan jumlah High Issues antara OTP-1 dan OTP-5, yang menunjukkan penurunan signifikan setelah penerapan perbaikan.

JISKA (Jurnal Informatika Sunan Kalijaga)
Vol. 11, No. 1, JANUARI, 2026: 13 – 31



Gambar 14 Grafik Evaluasi OTP pada AppSweep dalam 5 Kali Percobaan



Gambar 15 Perbandingan High Issues pada OTP-1 dan OTP-5 AppSweep

Meskipun sebagian besar kerentanannya terdeteksi pada kategori Medium dan Low, masalah ini tetap perlu diperhatikan. Dalam konteks aplikasi OTP, bahkan yang dianggap rendah bisa dimanfaatkan oleh peretas untuk mengeksploitasi celah keamanan, terutama yang berkaitan dengan kebocoran data atau pengaturan izin aplikasi. Misalnya, kebocoran data dapat menyebabkan informasi pribadi pengguna terekspos, sementara masalah izin dapat memberi akses yang tidak sah kepada pihak ketiga. Oleh karena itu, pengembang harus memastikan semua jenis kerentanannya, baik yang tinggi maupun rendah, ditangani dengan serius guna menjaga integritas dan keamanan aplikasi. Menghilangkan kerentanannya di kategori High dan memastikan bahwa IAST Issues telah diatasi sepenuhnya akan sangat meningkatkan keamanan aplikasi secara keseluruhan.

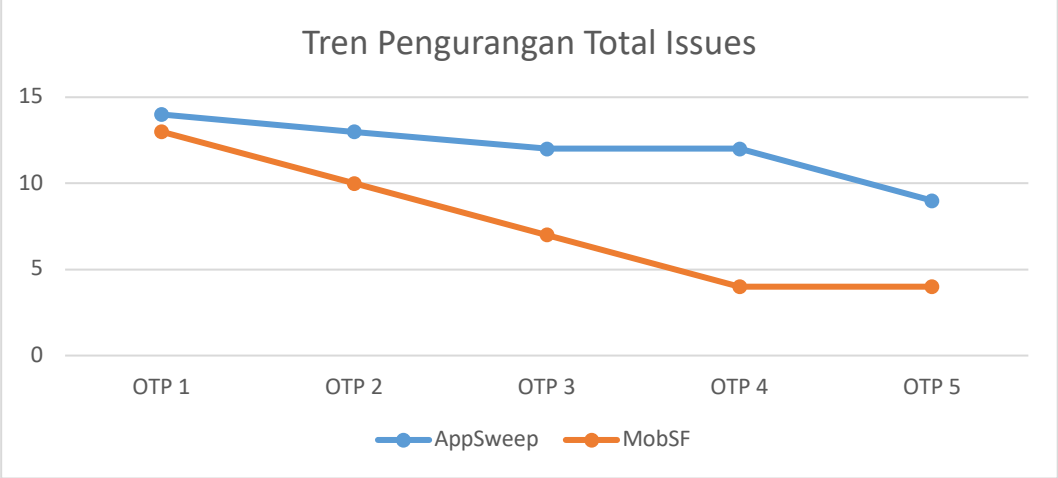
3.4 Perbandingan SAST dan IAST

Analisis Perbandingan SAST dan IAST menunjukkan perbedaan fokus dan keunggulan masing-masing metode. SAST cocok digunakan untuk deteksi dini selama proses pengembangan dan review kode, sementara IAST lebih efektif untuk mendeteksi eksploitasi di dunia nyata, meskipun lebih kompleks dan memerlukan runtime testing. Dalam konteks aplikasi mobile, kombinasi SAST dan IAST dapat meningkatkan keamanan secara menyeluruh dengan mengidentifikasi kelemahan sejak tahap awal pengembangan hingga pengujian di lingkungan runtime. Tabel 6 menyajikan perbandingan ringkas antara kedua metode ini.

Tabel 6 Perbandingan SAST dan IAST

Aspek	SAST (Static Application Security Testing)	IAST (Interactive Application Security Testing)
Metode Analisis	Analisis kode sumber, bytecode, atau biner tanpa menjalankan aplikasi.	Menguji aplikasi saat berjalan, memantau input, output, dan eksekusi runtime.
Cakupan Keamanan	Menemukan kerentanan dalam kode sebelum aplikasi dijalankan.	Mendeteksi kelemahan keamanan yang hanya muncul dalam kondisi runtime.
Waktu Deteksi	Dapat dilakukan lebih awal dalam pengembangan.	Mengidentifikasi kerentanan saat aplikasi berjalan, cocok untuk tahap pengujian akhir.
Akurasi	Cenderung menghasilkan false positives karena tidak mempertimbangkan kondisi runtime.	Lebih akurat dalam menemukan eksploitasi nyata karena berbasis eksekusi.
Kompleksitas	Relatif mudah diterapkan, tidak memerlukan eksekusi aplikasi.	Memerlukan lingkungan runtime yang mendukung pengujian interaktif.
Kecepatan	Lebih cepat karena tidak memerlukan eksekusi aplikasi.	Lebih lambat karena harus berjalan bersamaan dengan aplikasi.

Hasil evaluasi juga divisualisasikan dalam *line chart* yang menunjukkan tren penurunan jumlah isu keamanan dari OTP-1 hingga OTP-5, seperti terlihat pada Gambar 16. AppSweep secara konsisten mendeteksi lebih banyak isu dibandingkan MobSF, tetapi keduanya menunjukkan pola penurunan yang serupa, mencerminkan perbaikan keamanan pada setiap versi OTP. MobSF mengalami penurunan lebih tajam, dari 13 isu pada OTP-1 menjadi hanya 4 pada OTP-5, sementara AppSweep berkurang lebih bertahap, dari 14 menjadi 9. Hal ini menunjukkan bahwa MobSF cenderung lebih sensitif dalam mendeteksi perbaikan keamanan, sementara AppSweep mempertahankan standar deteksi yang lebih ketat. Kombinasi kedua metode ini dapat memberikan analisis yang lebih komprehensif, dengan MobSF mengidentifikasi celah yang dapat segera diperbaiki, sementara AppSweep memastikan evaluasi keamanan tetap menyeluruh hingga iterasi terakhir.



Gambar 16 Tren Pengurangan Total Issues

Hasil penelitian ini dapat diadaptasi dalam berbagai konteks yang memerlukan tingkat keamanan tinggi, seperti sistem perbankan dan e-commerce. Dalam perbankan digital, kombinasi SAST dan IAST dapat digunakan untuk mendeteksi dan mencegah eksploitasi keamanan pada aplikasi mobile banking, terutama dalam melindungi transaksi pengguna dan mencegah ancaman seperti man-in-the-middle attacks atau injeksi kode berbahaya. SAST dapat memastikan keamanan kode sejak tahap pengembangan, sementara IAST membantu mengidentifikasi kelemahan yang hanya muncul saat aplikasi berjalan, seperti vulnerable API calls atau autentikasi yang tidak aman. Dalam e-commerce, metode ini dapat diterapkan untuk menjaga keamanan transaksi pelanggan, terutama dalam sistem pembayaran online yang sering menjadi target serangan seperti phishing atau carding. Dengan menerapkan pendekatan serupa yang digunakan dalam penelitian ini, sistem perbankan dan e-commerce dapat meningkatkan ketahanan terhadap serangan siber serta memastikan pengalaman pengguna yang aman dan terpercaya.

4. KESIMPULAN

Penelitian ini berhasil mengimplementasikan OTP Firebase berbasis email pada aplikasi Android dengan pengujian keamanan menggunakan SAST (MobSF) dan IAST (AppSweep), yang efektif dalam mendeteksi kerentanan. Hasil pengujian menunjukkan perbaikan signifikan, di mana temuan kerentanannya berkurang dari 3 temuan tingkat tinggi pada percobaan pertama (OTP-1) menjadi 0 pada percobaan kelima (OTP-5), dengan skor aplikasi meningkat dari 43 (B) menjadi 78 (A) di MobSF. Begitu juga pada AppSweep, meskipun kerentanannya pada OTP-1 dan OTP-2 ditemukan pada kategori High, tidak ada temuan serupa pada OTP-3 hingga OTP-5, dengan jumlah total masalah menurun dari 14 menjadi 9. Keterbatasan penelitian ini mencakup cakupan dataset yang terbatas dan potensi bias alat pengujian, yang dapat diperbaiki dengan menggunakan dataset lebih besar dan metode tambahan seperti machine learning pada penelitian selanjutnya. Penelitian lanjutan juga akan mengadopsi model DevSecOps dengan IAST di tahap pengembangan (Dev) dan RASP di tahap operasi (Ops) untuk meningkatkan keamanan aplikasi lebih lanjut. Saat ini penelitian sudah mengintegrasikan RASP, namun masih pada level dasar dengan ProGuard. Metode machine learning mungkin akan diterapkan pada bagian Operasi (Ops), seperti pada IPS (Intrusion Prevention Systems), IDS (Intrusion Detection Systems), dan RASP (Runtime Application Self-Protection). Implikasi lebih luas dari penelitian ini adalah memberikan rekomendasi bagi pengembang aplikasi mobile untuk mengadopsi pendekatan pengujian keamanan yang lebih proaktif, dengan memanfaatkan kombinasi SAST dan IAST dalam siklus pengembangan perangkat lunak. Selain itu, penelitian masa depan dapat mengeksplorasi integrasi kecerdasan buatan (Artificial Intelligence) untuk otomatisasi deteksi kerentanan OTP serta menganalisis keamanan pada OTP berbasis biometrik. Dengan pendekatan ini, diharapkan sistem autentikasi berbasis OTP dapat semakin aman dan lebih adaptif terhadap berbagai ancaman siber di masa depan.

DAFTAR PUSTAKA

Asih, M. S., & Hasibuan, A. Z. (2023). Pengamanan Kunci Pintu Brankas Menggunakan Kriptografi One Time Pad (OTP) Berbasis Android. *Explorer*, *3*(2), 58–68. https://doi.org/10.47065/explorer.v3i2.738

Chimuco, F. T., Sequeiros, J. B. F., Lopes, C. G., Simões, T. M. C., Freire, M. M., & Inácio, P. R. M. (2023). Secure Cloud-Based Mobile Apps: Attack Taxonomy, Requirements, Mechanisms, Tests and Automation. *International Journal of Information Security*, *22*(4), 833–867. https://doi.org/10.1007/s10207-023-00669-z

Danthy, R., Pratham Pai, K., & Rao, V. (2024). Secure Online Banking Authentication System Using Time Bound Password. *2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT)*, 130–135. https://doi.org/10.1109/IC2PCT60090.2024.10486295

Ikram, M., Sentana, I. W. B., Asghar, H., Kaafar, M. A., & Kepkowski, M. (2025). More Than Just a Random Number Generator! Unveiling the Security and Privacy Risks of Mobile OTP Authenticator Apps. In M. Barhamgi, H. Wang, & X. Wang (Eds.), *Web Information Systems Engineering – WISE 2024* (pp. 177–192). Springer Nature Singapore. https://doi.org/10.1007/978-981-96-0576-7_14

Keteku, J., Dameh, G. O., Mante, S. A., Mensah, T. K., Amartey, S. L., & Diekuu, J.-B. (2024). Detection and Prevention of Malware in Android Mobile Devices: A Literature Review. *International Journal of Intelligence Science*, *14*(04), 71–93. https://doi.org/10.4236/ijis.2024.144005

Li, K., Chen, S., Fan, L., Feng, R., Liu, H., Liu, C., Liu, Y., & Chen, Y. (2023). Comparison and Evaluation on Static Application Security Testing (SAST) Tools for Java. *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 921–933. https://doi.org/10.1145/3611643.3616262

Lim, J. G. Q., Kwok, Z. Y., Soon, I., Yong, J. X., Song Yuhao, S., Binte Rosley, S. H., & Balachandran, V. (2024). A-COPILOT: Android Covert Operation for Private Information Lifting and OTP Theft: A Study on How Malware Masquerading as Legitimate Applications Compromise Security and Privacy. *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy*, 155–157. https://doi.org/10.1145/3626232.3658638

Lubis, D. J., & Riswanto, A. N. (2024). Implementasi Algoritma Random Forest untuk Optimasi Keamanan Autentikasi One-Time Password (OTP). *Informatech: Jurnal Ilmiah Informatika dan Komputer*, *1*(1), 23–29. https://doi.org/10.69533/eyp7ag46

Moon, I. T., Mimi, A., & Rahman Mridha, Md. M. (2023). Cryptographic Analysis: Popular Social Media Applications and Mitigations of Vulnerabilities. *2023 26th International Conference on Computer and Information Technology (ICCIT)*, 1–6. https://doi.org/10.1109/ICCIT60459.2023.10441090

Nutalapati, V. (2023). Automated Security Testing for Mobile Apps: Tools, Techniques, and Best Practices. *International Engineering & Applied Sciences (IRJEAS)*, *11*(1), 26. https://doi.org/10.55083/irjeas.2023.v11i01004

Pagano, F., Romdhana, A., Caputo, D., Verderame, L., & Merlo, A. (2023). SEBASTiAn: A Static and Extensible Black-Box Application Security Testing Tool for iOS and Android Applications. *SoftwareX*, *23*, Article ID: 101448. https://doi.org/10.1016/j.softx.2023.101448

Pargaonkar, S. (2023). Advancements in Security Testing: A Comprehensive Review of Methodologies and Emerging Trends in Software Quality Engineering. *International Journal of Science and Research (IJSR)*, *12*(9), 61–66. https://doi.org/10.21275/SR23829090815

Schleier, S., Holguera, C., Mueller, B., & Willemsen, J. (2024). *OWASP Mobile Application Security Testing Guide (MASTG)*. The OWASP Foundation. https://mas.owasp.org/MASTG/

Schleier, S., Holguera, C., Mueller, B., Willemsen, J., & Beckers, J. (2024). *OWASP Mobile Application Security Verification Standard v2.1.0*. The OWASP Foundation. https://mas.owasp.org/MASVS/

Smyčka, M. (2024). *Evaluation and Application of SAST Tools* [Masarykova Univerzita]. https://is.muni.cz/th/j4bxg/smycka_thesis.pdf

Sonnekalb, T., Knaust, C.-T., Gruner, B., Brust, C.-A., Heinze, T. S., Kurnatowski, L. von, Schreiber, A., & Mäder, P. (2023). A Static Analysis Platform for Investigating Security Trends in Repositories. *2023 IEEE/ACM 1st International Workshop on Software Vulnerability (SVM)*, 1–5. https://doi.org/10.1109/SVM59160.2023.00005

Stanciu, A.-M. (2023). Theoretical Study of Security for a Software Product. In *Lecture Notes in Networks and Systems* (Vol. 578, pp. 233–242). https://doi.org/10.1007/978-981-19-7660-5_20

Sutter, T., Kehrer, T., Rennhard, M., Tellenbach, B., & Klein, J. (2024). Dynamic Security Analysis on Android: A Systematic Literature Review. *IEEE Access*, *12*, 57261–57287. https://doi.org/10.1109/ACCESS.2024.3390612

Taqwim, M. A., Kusyanti, A., & Siregar, R. A. (2021). Implementasi Algoritme Speck untuk Enkripsi One-Time Password pada Two-Factor Authentication. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *5*(7), 3103–3111. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/9488

Wibawa, S., Suryanto, S., & Ningsih, R. (2024). Perlindungan Data Digital Dengan Time-Based One-Time Password (TOTP). *INSANtek*, *5*(1), 30–36. https://doi.org/10.31294/insantek.v5i1.3495