BAB 2

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian mengenai mitigasi keamanan aplikasi mobile dilakukan oleh **Nugraha (2026)** yang mengevaluasi aplikasi panen sawit menggunakan MobSF, di mana hasil penelitian menunjukkan bahwa perbaikan konfigurasi *build-time* berdasarkan rekomendasi analisis statis dapat secara signifikan meningkatkan skor keamanan hingga kategori Low Risk [6]. Selanjutnya, **Himawan (2022)** menekankan pentingnya metode Analisis Statis dalam mendeteksi malware pada berkas APK guna melindungi informasi pengguna dari ancaman aplikasi yang terproduksi secara masif [13].

Pendekatan berbeda diambil oleh **Tansen & Nurdiarto (2020)** yang menerapkan *hybrid analysis* dengan metode *one-shot case study* untuk mencari kejanggalan pada malware yang menginfeksi perangkat Android [11]. Karakteristik izin akses yang berbahaya dianalisis oleh **Sakti & Arifin (2025)**, yang menemukan bahwa aplikasi dari sumber tidak resmi cenderung meminta izin berlebihan yang tidak sesuai dengan fungsi utamanya[3] Hal ini sejalan dengan studi dari **Subakja dkk. (2025)** pada aplikasi transportasi online yang mengungkap adanya kerentanan pada parameter izin aplikasi dan algoritma kriptografi yang sudah tidak relevan atau ketinggalan [7].

Penelitian pada aplikasi pesan instan yang dilakukan oleh **Nugraha & Yasa (2024)** menggunakan standar OWASP MASVS menunjukkan bahwa aplikasi populer seperti Viber masih memiliki puluhan indikasi kerentanan keamanan[4]. Untuk pengujian di sisi jaringan, **Karuniawan (2023)** berhasil mendeteksi transmisi data sensitif berupa username dan password yang dikirimkan tanpa enkripsi melalui pemantauan lalu lintas data menggunakan Wireshark[14].Penggunaan kecerdasan buatan menjadi tren signifikan dalam optimasi deteksi malware, dibuktikan oleh **Adebayo (2019)** yang berhasil mencapai akurasi 98,17% melalui algoritma *Particle Swarm Optimization* (PSO) [15]. Sementara itu, **Maajid dkk. (2021)** membangun sistem deteksi otomatis menggunakan platform *Malice* di lingkungan Linux untuk mempercepat identifikasi sampel malware secara terisolasi [16].

Laporan teknis mengenai pembuatan indikator serangan (*IOC*) berdasarkan standar STIX disusun oleh **Rahayu & Trianto (2021)** untuk memperkuat pertahanan siber nasional[5]. Karakteristik izin akses yang paling sering disalahgunakan seperti akses internet dan penyimpanan lokal juga dipetakan secara mendalam oleh **Iman dkk. (2019)** [17]**. (2021)** menggabungkan fitur statis dan dinamis dengan arsitektur *Deep Learning* LSTM dibuktikan dapat mencapai tingkat akurasi identifikasi malware yang sangat tinggi, yaitu 98,7% [18]. Penelitian yang dilakukan oleh **Syifa Tiara Ratri & Kristina Setyowati** menunjukkan bahwa aplikasi Jogja Istimewa cukup efektif dalam mendukung pelayanan publik berbasis digital di Daerah Istimewa Yogyakarta. Namun, penelitian tersebut berfokus pada efektivitas pelayanan publik dan belum membahas aspek keamanan aplikasi Android secara teknis[19]

Tabel 2.1 Penelitian Terdahulu

NO	Peneliti	Metode	Objek Penelitian	Hasil
1.	Tansen & Nurdiarto (2020)	*Hybrid Analysis (Static + Dynamic)* menggunakan MobSF	*Malware* Android	Kombinasi analisis statis dan dinamis mampu mendeteksi karakteristik dan perilaku *malware* secara lebih akurat.
2.	Himawan, Septianzah & Setiadi (2022)	Analisis Statis menggunakan MobSF	Aplikasi Android (APK)	MobSF mampu menganalisis keamanan aplikasi Android dan mendeteksi potensi *malware* dalam file APK.

3.	Nugraha & Mansur (2026)	*Static dan Dynamic Analysis* menggunakan MobSF	Aplikasi *Mobile* Sistem Informasi	Analisis meningkatkan skor keamanan aplikasi serta menemukan kerentanan backend seperti IDOR dan Broken Access Control.
4.	Abdul Sakti & Dafrid Cahyadi Arifin	Static dan Dynamic Analysis menggunakan tools MobSF	Lima sampel aplikasi Android (APK) dari Google Play Store dan situs pihak ketiga	Aplikasi dari sumber tidak resmi memiliki risiko lebih tinggi (izin akses berlebihan, struktur file dimodifikasi, dan komunikasi server mencurigakan). Aplikasi resmi lebih aman meski ada kerentanan kecil. MobSF efektif memberikan skor keamanan kuantitatif

5.	Triyawan Bagus Subakja, Mona Fronita, Syaifullah ,Tengku Khairil Ahsyar, Syafril Siregar.(2025)	Analisis Statis menggunakan MobSF dengan parameter: dangerous permissions, weak crypto, root detection, SSL bypass, dan domain *malware* check	Tiga aplikasi transportasi *online* populer di Indonesia: Gojek, Grab, dan Maxim	Ketiga aplikasi memiliki risiko keamanan tingkat sedang (skor 44-50). Ditemukan celah pada *dangerous permissions* dan *weak crypto* di semua aplikasi; Maxim terdeteksi rentan terhadap *SSL Bypass*. Tidak ditemukan adanya *root detection*
6.	Peneliti pada Jurnal Info Kripto (2024)	Analisis statis pada source code menggunakan MobSF serta evaluasi berdasarkan standar OWASP MASVS, NIST SP 800-163 Rev 1, dan NIAP	11 aplikasi pesan instan berbasis Android (Element, Line, Mesibo, Rocket.Chat, Signal, Skype, Snapchat, Speek!, Telegram, Viber, dan WhatsApp)	Speek! mendapatkan skor keamanan tertinggi (paling aman) dengan kerentanan paling sedikit, sedangkan Viber adalah yang paling tidak aman dengan temuan 32

				kerentanan
7.	Syifa Tiara Ratri & Kristina Setyowati	Analisis efektivitas aplikasi *e-government*	Aplikasi Jogja Istimewa (JOGIS)	Aplikasi JOGIS dinilai cukup efektif dalam mendukung pelayanan publik digital di DIY, namun penelitian belum membahas aspek keamanan aplikasi Android secara teknis.

2.2 Gap Penelitian

Tabel 2.2 Gap Penelitian

Aspek	Penelitian Sebelumnya	Penelitian Ini
Objek Penelitian	Malware Android, aplikasi transportasi online, aplikasi pesan instan, dan aplikasi mobile umum	Aplikasi Jogja Istimewa (JOGIS) sebagai aplikasi layanan publik Pemerintah Daerah DIY
Fokus Penelitian	Deteksi malware, identifikasi kerentanan umum, analisis permission, dan evaluasi keamanan aplikasi komersial	Analisis tingkat keamanan aplikasi layanan publik berbasis Android
Metode Analisis	Analisis Statis, dynamic analysis, hybrid analysis,	Analisis Statis menggunakan Mobile Security Framework (MobSF)

	machine learning, dan network analysis	
Tools yang Digunakan	MobSF, Wireshark, Malice, PSO, Deep Learning LSTM, dan tools lainnya	Mobile Security Framework (MobSF) berbasis Docker
Output Penelitian	Identifikasi malware, deteksi kerentanan, analisis lalu lintas jaringan, dan evaluasi keamanan aplikasi	Identifikasi kerentanan, analisis permission, analisis konfigurasi keamanan, dan penilaian tingkat keamanan aplikasi JOGIS
Kontribusi Penelitian	Berfokus pada aplikasi komersial, malware Android, atau metode deteksi keamanan secara umum	Memberikan gambaran tingkat keamanan aplikasi JOGIS serta rekomendasi perbaikan keamanan pada aplikasi layanan publik daerah

2.3 Dasar Teori

2.3.1 Keamanan Aplikasi *Mobile*

Keamanan aplikasi mobile merupakan salah satu aspek paling penting dalam pengembangan perangkat lunak yang bertujuan untuk melindungi data dan sistem dari berbagai ancaman keamanan. Aplikasi mobile sering kali membutuhkan akses dari berbagai jenis data sensitif seperti informasi pribadi pengguna, lokasi, maupun data autentikasi ,oleh karena itu diperlukan mekanisme keamanan yang baik untuk mencegah penyalahgunaan data tersebut. Ancaman keamanan pada aplikasi mobile dapat berasal dari berbagai sumber seperti malware, serangan jaringan, maupun kelemahan pada kode dari aplikasi. Beberapa jenis kerentanan yang sering ditemukan pada aplikasi *Mobile* antara lain *insecure data storage*, *insecure communication*, *improper authentication*, *serta insufficient cryptography* , permission Android sendiri merupakan mekanisme kontrol akses yang digunakan aplikasi untuk mengakses sumber daya perangkat pengguna seperti lokasi, kamera, penyimpanan, dan jaringan internet.

Penggunaan fitur atau akses yang mencapai tingkat dangerous permission tanpa pengelolaan keamanan yang tepat dapat meningkatkan risiko kebocoran data dan *unauthorized access*.[20]Pengujian keamanan aplikasi mobile dilakukan untuk mengidentifikasi potensi kerentanan yang terdapat pada aplikasi sehingga dapat dilakukan perbaikan sebelum aplikasi digunakan oleh pengguna secara luas.

2.3.2 Android Application Package (APK)

Android Application Package (APK) merupakan format file yang digunakan oleh sistem operasi Android untuk mendistribusikan dan menginstal aplikasi. File APK berisi berbagai komponen yang diperlukan untuk menjalankan aplikasi Android seperti kode program, resource, dan file konfigurasi. Keamanan pada komunikasi jaringan pada aplikasi mobile menjadi aspek penting karena kerentanan pada API maupun proses pertukaran data dapat dimanfaatkan untuk melakukan *interception*, manipulasi data, maupun akses ilegal terhadap sistem backend aplikasi[21]

Struktur dasar dari file APK terdiri dari beberapa komponen utama yaitu:

1. AndroidManifest.xml

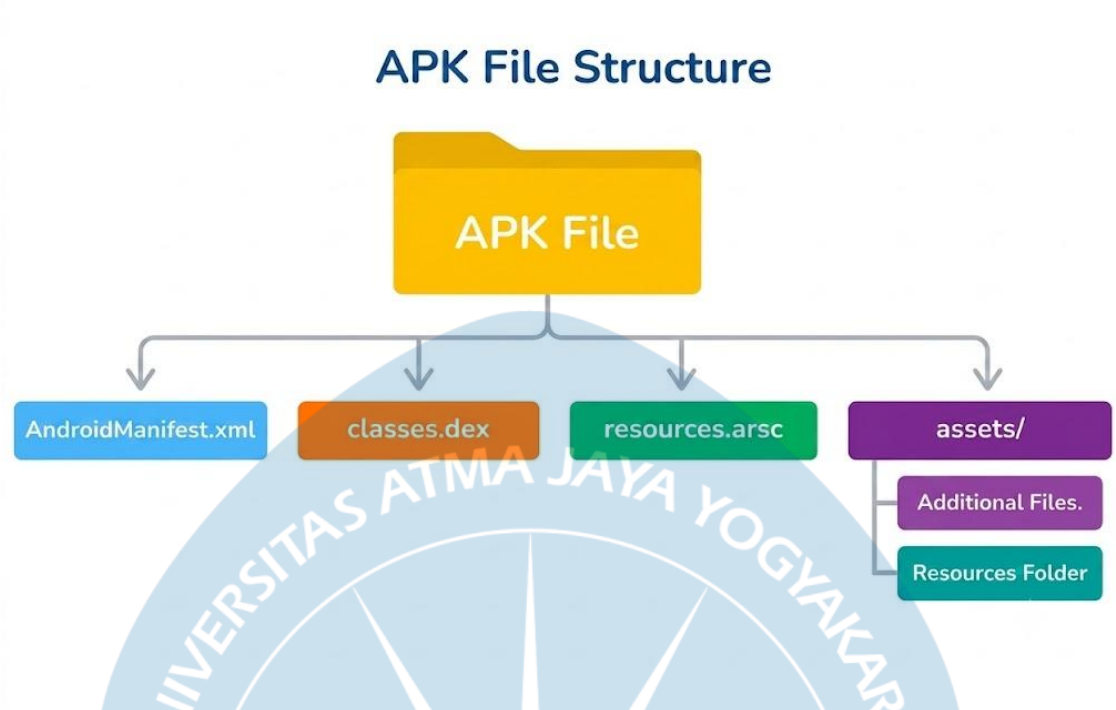
   Berisi informasi mengenai konfigurasi aplikasi seperti permission, activity, service, dan komponen lainnya.
2. classes.dex

   Merupakan file yang berisi kode program aplikasi Android dalam bentuk Dalvik bytecode.
3. resources.arsc

   Berisi resource aplikasi seperti string, layout, dan gambar.
4. assets

   Berisi file tambahan yang digunakan oleh aplikasi.

Analisis terhadap file APK dapat dilakukan untuk mengetahui struktur aplikasi serta potensi kerentanan keamanan yang terdapat didalamnya.



Gambar 2.1 Komponen APK File

2.3.3 Analisis Statis

Analisis Statis merupakan metode analisis perangkat lunak yang dilakukan tanpa menjalankan aplikasi. Analisis ini dilakukan dengan memeriksa kode sumber atau struktur program ,alur eksekusi , konfigurasi ,serta kemungkinan munculnya kerentanan keamanan[22]. Berbeda dengan analisis dinamis yang hanya mengamati perilaku aplikasi ketika dijalankan, analisis statis memungkinkan seluruh bagian kode diperiksa sehingga potensi kerentanan dapat diidentifikasi sejak tahap awal pengembangan maupun sebelum aplikasi dipublikasikan.Metode ini dapat digunakan untuk mendeteksi berbagai jenis kerentanan seperti penggunaan permission yang berlebihan, insecure API usage, serta konfigurasi keamanan yang tidak tepat.

Dalam proses analitis statis , metode ini menampilkan *Control Flow Graph* (CFG), *Call Graph*, maupun *Data Flow Graph* untuk menggambarkan hubungan antar metode, jalur eksekusi program, serta aliran data. *Control Flow Graph* (CFG) digunakan untuk memetakan urutan eksekusi setiap instruksi dalam aplikasi, sehingga memungkinkan

identifikasi jalur program yang berpotensi menimbulkan kerentanan. *Call Graph* menggambarkan hubungan pemanggilan antar metode atau fungsi sehingga memudahkan analisis interaksi antar komponen aplikasi. Sementara itu *Data Flow Graph* digunakan untuk mencari dan menelusuri bagaimana data berpindah dan diproses di dalam aplikasi, termasuk mendeteksi kemungkinan aliran data sensitif menuju lokasi yang tidak seharusnya. Hal tersebut yang akan menjadi dasar untuk mendeteksi berbagai kelemahan keamanan, seperti penggunaan permission yang berlebihan atau tidak sesuai kebutuhan, penggunaan *Application Programming Interface* (API) yang tidak aman, penyimpanan informasi sensitif secara langsung di dalam kode (*hardcoded credentials*), maupun konfigurasi aplikasi yang berpotensi menimbulkan celah keamanan.[22]

2.3.4 *Mobile* Security Framework (MobSF)

Mobile Security Framework (MobSF) merupakan salah satu alat yang digunakan untuk melakukan analisis keamanan aplikasi Mobile secara otomatis. MobSF mendukung analisis statis dan dinamis terhadap aplikasi Android dan iOS.

MobSF dapat melakukan berbagai jenis analisis seperti:

a. Analisis struktur file APK
b. Analisis permission aplikasi
c. Analisis API calls
d. Analisis komunikasi jaringan
e. Deteksi kerentanan keamanan berdasarkan standar keamanan *Mobile*

Selain itu MobSF juga menyediakan laporan hasil analisis yang berisi berbagai informasi mengenai potensi kerentanan keamanan yang ditemukan pada aplikasi.

2.3.5 Docker

Docker merupakan platform yang digunakan untuk menjalankan aplikasi dalam bentuk container. Container memungkinkan aplikasi dijalankan dalam lingkungan yang terisolasi sehingga memudahkan proses deployment dan pengujian perangkat lunak.

Dalam penelitian ini Docker digunakan untuk menjalankan Mobile Security Framework (MobSF) sehingga proses analisis aplikasi Android dapat dilakukan secara terisolasi tanpa mempengaruhi sistem utama.[23]

Penggunaan Docker juga memudahkan proses instalasi dan konfigurasi MobSF karena seluruh dependensi yang dibutuhkan telah tersedia di dalam container.

2.3.6 OWASP *Mobile* Top 10

OWASP *Mobile* Top 10 merupakan daftar sepuluh kerentanan keamanan yang paling umum ditemukan pada aplikasi *Mobile* . Daftar ini dikembangkan oleh Open Web Application Security Project (OWASP) sebagai panduan bagi pengembang dan peneliti dalam meningkatkan keamanan aplikasi *Mobile* [24].

Beberapa kerentanan yang termasuk dalam OWASP *Mobile* Top 10 antara lain:

a. Improper Platform Usage
b. Insecure Data Storage
c. Insecure Communication
d. Insecure Authentication
e. Insufficient Cryptography
f. Insecure Authorization
g. Client Code Quality Issues
h. Code Tampering
i. Reverse Engineering
j. Extraneous Functionality[25]

OWASP *Mobile* Top 10 sering digunakan sebagai standar dalam melakukan pengujian keamanan aplikasi *Mobile* untuk mengidentifikasi berbagai potensi kerentanan keamanan.

2.3.7 OWASP MASVS Privacy

Hasil analisis menggunakan MobSF, aplikasi JOGIS terbukti menggunakan beberapa permission yang berkaitan langsung dengan privasi pengguna, seperti ACCESS_FINE_LOCATION, ACCESS_BACKGROUND_LOCATION, CAMERA, RECORD_AUDIO, serta izin akses terhadap media dan penyimpanan perangkat. Permission tersebut termasuk kategori sensitif karena memungkinkan aplikasi mengakses data pribadi maupun sumber daya penting pada perangkat pengguna ,hal ini bisa dikaitkan dengan domain MASVS-PRIVACY yang mana penggunaan permission tersebut perlu memperhatikan prinsip *least privilege*, yaitu hanya meminta akses yang benar-benar diperlukan untuk mendukung fungsi aplikasi.

*OWASP Mobile Application Security Verification Standard (*MASVS) memiliki beberapa domain keamanan yang digunakan sebagai landasan dalam mengevaluasi keamanan aplikasi mobile. Salah satu domain yang relevan dengan penelitian ini adalah MASVS-PRIVACY yang berfokus pada perlindungan privasi pengguna dan pengelolaan data sensitif pada aplikasi mobile. MASVS-PRIVACY bertujuan untuk memastikan bahwa aplikasi mengelola data pengguna secara bertanggung jawab, membatasi akses terhadap informasi sensitif, serta memberikan transparansi terkait pengumpulan dan penggunaan data. Domain ini terdiri dari beberapa kontrol keamanan, yaitu MASVS-PRIVACY-1 hingga MASVS-PRIVACY-4.

MASVS-PRIVACY-1 menekankan bahwa aplikasi harus meminimalkan akses terhadap data sensitif dan sumber daya perangkat. MASVS-PRIVACY-2 berkaitan dengan upaya mencegah identifikasi pengguna secara tidak perlu. MASVS-PRIVACY-3 menekankan transparansi aplikasi dalam proses pengumpulan dan penggunaan data. Sementara itu, MASVS-PRIVACY-4 mengatur agar pengguna memiliki kendali terhadap data yang dimiliki dan digunakan oleh aplikasi.Dalam penelitian ini, pembahasan difokuskan pada MASVS-PRIVACY-1 karena hasil analisis statis menggunakan MobSF menghasilkan informasi mengenai penggunaan permission dan akses terhadap sumber daya sensitif perangkat Android yang dapat dievaluasi berdasarkan prinsip minimisasi akses data.

2.3.8 Android Permission dan Klasifikasinya

Izin akses atau permission adalah metode perlindungan dalam sistem operasi Android yang berfungsi untuk mengatur apa saja yang dapat diakses oleh aplikasi terhadap sumber daya perangkat. Dengan adanya permission, pengguna bisa mengetahui informasi atau fitur perangkat yang akan diakses oleh aplikasi, seperti kamera, lokasi, mikrofon, penyimpanan, dan daftar kontak. Android membagi permission menjadi beberapa kelompok, yaitu normal permission, dangerous permission, signature permission, dan special permission[26]. Normal permission adalah izin dengan tingkat risiko rendah yang secara otomatis diberikan oleh sistem.

Dangerous permission adalah izin yang mengakses informasi sensitif pengguna sehingga memerlukan izin langsung dari mereka. Signature permission hanya dapat digunakan oleh aplikasi yang menggunakan sertifikat yang sama, sedangkan special permission diberikan untuk fungsi tertentu yang memerlukan pengawasan ekstra dari sistem operasi[27]. Penggunaan permission yang tidak sesuai dengan konteks aplikasi dapat menyebabkan masalah keamanan dan privasi bagi pengguna. Oleh karena itu, melakukan analisis terhadap permission menjadi salah satu hal yang sangat penting dalam menguji keamanan aplikasi Android.

2.3.9 AndroidManifest.xml

AndroidManifest. xml adalah salah satu file krusial dalam aplikasi Android yang berfungsi untuk menyimpan data penting seputar pengaturan aplikasi. File ini mencakup identitas aplikasi, deklarasi aktivitas, layanan, penerima siaran, penyedia konten, serta daftar izin yang diperlukan oleh aplikasi[28]. Ketika melakukan analisis keamanan, AndroidManifest. xml menjadi salah satu elemen yang esensial untuk diteliti karena mampu memberikan wawasan mengenai pengaturan aplikasi dan izin yang diminta. Dengan memeriksa file ini, analis dapat menemukan izin yang tidak wajar, komponen yang terbuka untuk aplikasi lain, serta konfigurasi yang dapat menyebabkan kerentanan dalam keamanan. Selain itu, AndroidManifest xml juga berfungsi sebagai sumber informasi awal untuk memahami struktur aplikasi sebelum menganalisis lebih mendalam terhadap kode sumber dan komponen lain yang terkait.

2.3.10 Kerentanan (Vulnerability) pada Aplikasi Android

Kerentanan diartikan sebagai kelemahan yang ada dalam sistem, aplikasi, atau pengaturan yang dapat dimanfaatkan oleh individu yang tidak berhak untuk mendapatkan akses, mengganggu layanan, atau mengambil informasi yang disimpan dalam sistem. Dalam konteks aplikasi Android, kerentanan dapat muncul sebagai akibat dari kesalahan dalam konfigurasi, pemilihan algoritma kriptografi yang lemah, penyimpanan data yang tidak aman, atau penerapan kode program yang tidak sesuai dengan praktik keamanan yang disarankan. Beberapa jenis kerentanan yang sering dijumpai pada aplikasi Android mencakup penggunaan izin akses yang berlebihan, penyimpanan data sensitif tanpa adanya enkripsi, jaringan komunikasi yang tidak aman, kredensial yang terintegrasi langsung, serta kesalahan dalam pengaturan pada komponen aplikasi.

Kerentanan ini dapat meningkatkan kemungkinan terjadinya kebocoran data, manipulasi informasi, atau akses ilegal terhadap sumber daya aplikasi. Oleh karena itu, langkah untuk mengidentifikasi kerentanan menjadi aspek krusial dalam analisis keamanan aplikasi Android. Dengan melakukan identifikasi kerentanan, pengembang dapat mengenali potensi risiko keamanan yang ada dalam aplikasi dan melakukan perbaikan sebelum aplikasi tersebut diluncurkan secara luas kepada pengguna.

2.4 Kerangka Pemikiran

Kerangka pemikiran merupakan gambaran alur penelitian yang menjelaskan proses analisis keamanan aplikasi Android menggunakan metode Analisis Statis dengan memanfaatkan Mobile Security Framework (MobSF). Penelitian ini berfokus pada analisis keamanan aplikasi Jogja Istimewa (JOGIS) sebagai aplikasi layanan publik berbasis Android untuk mengidentifikasi potensi kerentanan keamanan yang terdapat pada aplikasi. Proses penelitian dimulai dengan pengumpulan file APK aplikasi JOGIS yang kemudian dianalisis menggunakan MobSF berbasis Docker sebagai lingkungan analisis yang terisolasi dan terkontrol. Selanjutnya, MobSF melakukan Analisis Statis terhadap struktur aplikasi, AndroidManifest.xml, permission aplikasi, source code, serta komponen lain yang berkaitan dengan

keamanan aplikasi Android.

Hasil analisis dari MobSF berupa laporan keamanan yang berisi informasi mengenai potensi vulnerability, konfigurasi tidak aman, penggunaan permission, serta tingkat risiko keamanan aplikasi. Hasil tersebut kemudian dianalisis untuk mengetahui kondisi keamanan aplikasi JOGIS berdasarkan temuan yang diperoleh selama proses pengujian. Melalui proses penelitian tersebut diharapkan dapat diperoleh gambaran mengenai tingkat keamanan aplikasi JOGIS serta rekomendasi yang dapat digunakan sebagai upaya peningkatan keamanan aplikasi layanan publik berbasis Android.