e- ISSN: 2828–299X

Rancang Bangun Library Deteksi Aplikasi Berbahaya Pada Perangkat Android Berbasis Perizinan Dan Nama Paket

Mohammad Arbi Yoganata [1], Mohammat Faried Rahmat[2]
[1]Teknik Informatika, Fakultas Teknologi Informasi, Universitas Islam Balitar
[2] Sistem Komputer, Fakultas Teknologi Informasi, Universitas Islam Balitar
E-mail: *[1] mohammadarbiyoganata@gmail.com, [2] mrhmt81@gmail.com

Abstrak – *Penelitian ini bertujuan merancang pustaka (library) untuk mendeteksi aplikasi berbahaya pada perangkat Android menggunakan bahasa pemrograman Kotlin, dengan fokus pada analisis izin aplikasi dan identifikasi nama paket berisiko. Metode yang digunakan adalah Research and Development (R&D) dengan pendekatan Mobile-D, yang terdiri dari lima fase: eksplorasi, inisialisasi, produksi, stabilisasi, dan pengujian sistem. Pustaka ini dirancang untuk memberikan peringatan dini kepada pengguna dan pengembang mengenai aplikasi yang meminta izin berisiko tinggi, seperti akses ke pesan teks, kontak, dan lokasi. Dengan arsitektur modular yang mencakup komponen seperti SecurityScanner, RiskAnalyzer, dan PermissionManager. Hasil penelitian menunjukkan pustaka ini efektif sebesar 90% dimana 9 dari 10 kasus bernilai valid dalam mendeteksi aplikasi berbahaya dan memberikan rekomendasi keamanan. Meskipun memiliki kelebihan, terdapat keterbatasan dalam mendeteksi semua jenis aplikasi berbahaya dan perlunya pembaruan berkala. Penelitian ini memberikan kontribusi signifikan dalam meningkatkan kesadaran akan keamanan aplikasi Android.*

Kata Kunci *— Android, Kotlin, Library, Malware Detection*

1. PENDAHULUAN

Perkembangan pesat sistem operasi Android telah membawa tantangan baru terkait keamanan bagi penggunanya. Berbagai aplikasi berbahaya atau *malware* dapat mengakses data pribadi, menimbulkan kerugian finansial, serta merusak integritas sistem. Di sisi lain, pentingnya deteksi *malware* yang berbasis pada izin dalam aplikasi Android untuk mengenali aplikasi-aplikasi berbahaya.

Metode untuk mendeteksi *malware* pada Android umumnya terbagi menjadi dua kategori, analisis statis dan analisis dinamis. Analisis statis dilakukan dengan memeriksa kode sumber tanpa menjalankan aplikasi, sementara analisis dinamis mengamati perilaku aplikasi saat dijalankan. Menerapkan metode *reverse engineering* untuk menganalisis kode dari aplikasi yang diduga mengandung malware, dengan tujuan meningkatkan kesadaran pengguna mengenai dampak negatif *malware* pada perangkat Android[1]. *Malware* pada Android dapat mengancam pengguna dari berbagai aspek, termasuk merusak komponen-komponen sistem operasi melalui perangkat lunak jahat[2].

Dalam upaya meningkatkan keamanan perangkat Android, penelitian ini bertujuan untuk merancang dan membangun sebuah *library* menggunakan bahasa pemrograman Kotlin yang dapat mendeteksi aplikasi berbahaya. Deteksi ini dilakukan dengan menganalisis izin aplikasi serta mengidentifikasi nama paket yang dianggap berisiko. Diharapkan, pendekatan ini dapat memberikan kontribusi yang signifikan dalam melindungi pengguna dari ancaman *malware* pada perangkat Android.

2. METODE PENELITIAN

Penelitian ini menggunakan metode Research and Development (R&D) untuk merancang dan mengembangkan *library* pendeteksi aplikasi berbahaya pada perangkat Android. Metode R&D bertujuan menghasilkan produk tertentu dan menguji keefektifannya[3]. Dalam konteks ini, produk yang dikembangkan adalah *library* yang mampu mendeteksi aplikasi dengan perizinan berbahaya dan mengidentifikasi aplikasi berbahaya berdasarkan nama paket.

2.1 Tahapan Pengembangan

Metode Mobile-D adalah salah satu metodologi pengembangan perangkat lunak yang dirancang khusus untuk aplikasi *mobile*. Metode ini merupakan adaptasi dari metodologi Agile, yang berfokus pada kelincahan dan fleksibilitas dalam pengembangan perangkat lunak. Mobile-D menggabungkan prinsip-prinsip Agile dengan

pendekatan yang lebih spesifik untuk kebutuhan pengembangan aplikasi *mobile* yang cocok untuk penelitian ini[4].

Gambar 1 Tahapan Mobile D

Berikut ini adalah tahapan dalam metode pengembangan *mobile-d* menurut [5]:

1. Explore
   Pada fase ini, dilakukan perencanaan dan penyusunan proyek yang akan dikerjakan. Tahap ini meletak isu-isu dasar pengembangan sistem, antara lain arsitektur produk, proses pengembangan , dan lingkungan pengembangan.
2. Initialize
   Fase ini bertujuan menyiapkan dan memverifikasi semua isu-isu kritis dalam pengembangan yang menentukan keberhasilan proyek. Di akhir tahap ini, diharapkan semua sumber daya telah siap untuk memulai membangun sistem.
3. Productionize
   Pada tahap ini, semua kebutuhan fungsional diimplementasikan pada produk dengan fokus pada pengembangan fitur utama.
4. Stabilize
   Fase ini melibatkan pengujian dan perbaikan untuk memastikan stabilitas dan kualitas produk, termasuk identifikasi dan perbaikan cacat.
5. System Test and Fix
   Tahap akhir ini melibatkan pengujian sistem secara keseluruhan dan perbaikan akhir sebelum rilis produk.

Penerapan metodologi Mobile-D efektif dalam pengembangan aplikasi *mobile*, seperti yang ditunjukkan dalam penelitian yang mengadopsi metodologi ini untuk mengembangkan aplikasi dengan arsitektur sistem berlapis, menghasilkan produk yang fungsional dan memenuhi kebutuhan pengguna. Selain itu, studi lain menunjukkan bahwa penggunaan metodologi Mobile-D dalam pengembangan aplikasi *mobile* dapat meningkatkan efisiensi dan kualitas produk akhir, dengan mengadopsi pendekatan iteratif dan kolaboratif yang sesuai dengan dinamika proyek perangkat lunak *mobile* [6].

3. HASIL DAN PEMBAHASAN

3.1 Analisa Kebutuhan Sistem

Perancangan *library* untuk mendeteksi aplikasi berbahaya ini bertujuan menjadi referensi sekaligus alat bantu bagi pengembang dalam proses pengembangan aplikasi. Dengan menggunakan *library*, waktu pengembangan aplikasi dapat dipersingkat, sehingga lebih efisien bagi pengembang dalam menyelesaikan aplikasi yang sedang atau akan mereka kembangkan.

Library ini dirancang menggunakan bahasa pemrograman Kotlin berdasarkan berbagai pertimbangan yang telah dilakukan oleh peneliti. Kotlin dipilih karena memiliki komunitas yang besar serta sintaksis yang relatif mudah dipahami oleh para pengembang aplikasi *mobile*. Selain itu, perancangan *library* ini mencakup dua aspek utama, yaitu kebutuhan fungsional dan non-fungsional, yang masing-masing memiliki peran spesifik dalam memastikan efektivitas dan efisiensi penggunaannya.

Tabel 1 Kebutuhan Fungsional

Kebutuhan	Deskripsi
Deteksi Perizinan Berbahaya	*Library* akan menganalisis dan mengidentifikasi aplikasi yang meminta izin berisiko tinggi.
Identifikasi Aplikasi Berdasarkan Nama Paket	*Library* akan memiliki daftar awal nama paket aplikasi yang diketahui berbahaya atau mencurigakan.

e- ISSN: 2828–299X

Tabel 2 Kebutuhan Non-Fungsional

Kebutuhan	Deskripsi
Kompatibilitas	*Library* akan dirancang untuk kompatibel dengan versi Android yang paling umum digunakan saat ini.
Kinerja	Proses deteksi akan dioptimalkan agar berjalan efisien.

Deteksi perizinan berbahaya akan difokuskan untuk menganalisis aplikasi yang meminta izin berisiko tinggi, seperti akses ke pesan teks, kontak, dan lokasi. Izin-izin ini diprioritaskan karena sering dimanfaatkan oleh aplikasi berbahaya untuk mengakses data sensitif pengguna. Dengan mendeteksi aplikasi yang meminta izin tersebut, *library* dapat memberikan peringatan dini kepada pengguna atau pengembang mengenai potensi risiko keamanan.

Identifikasi aplikasi berbahaya berdasarkan nama paket akan dilengkapi dengan daftar awal nama paket aplikasi yang diketahui berbahaya atau mencurigakan. Daftar ini memungkinkan *library* untuk melakukan deteksi dasar terhadap aplikasi yang terpasang pada perangkat. Meskipun daftar ini mungkin belum lengkap, namun akan menjadi dasar yang solid untuk pengembangan lebih lanjut dan pembaruan di masa depan.

Kompatibilitas akan dirancang untuk kompatibel dengan versi Android yang paling umum digunakan saat ini. Hal ini memastikan bahwa *library* dapat diintegrasikan dengan mudah ke dalam berbagai aplikasi tanpa memerlukan modifikasi signifikan, sehingga memudahkan pengembang dalam mengadopsi dan menerapkannya. Kinerja pada aplikasi akan dioptimalkan agar berjalan efisien tanpa mengurangi performa perangkat atau aplikasi yang mengintegrasikan *library* ini. Efisiensi ini penting untuk memastikan bahwa pengalaman pengguna tidak terganggu dan perangkat tetap beroperasi dengan lancar meskipun library berjalan di latar belakang.

Dalam proses perancangan *library* deteksi aplikasi berbahaya diperlukan spesifikasi perangkat keras dan perangkat lunak yang mendukung perancangan *library*. Berikut rincian dari kebutuhan perancangan *library* yang diperlukan:

Tabel 3 Kebutuhan perangkat

No.	Perangkat	Deskripsi
1	Laptop/PC	Perangkat untuk pengembangan *library*
2	Android	Perangkat untuk menjalankan aplikasi
3	Internet	Sumber jaringan untuk mengunduh dependensi yang diperlukan
4	Android Studio	IDE untuk pengembangan aplikasi
5	Gradle	Manajemen dependensi
6	Java Development Kit	Kompilasi kode

3.2 Perancangan Library

Perancangan *library* untuk mendeteksi aplikasi berbahaya pada perangkat Android dilakukan dengan pendekatan modular dan berorientasi objek menggunakan bahasa pemrograman Kotlin. Pendekatan ini memastikan library mudah diintegrasikan, dipelihara, dan diperluas sesuai kebutuhan. Berikut adalah detail perancangan yang meliputi arsitektur, komponen utama, alur kerja, dan implementasi kode.

3.2.1 Arsitektur Library

Library yang dikembangkan menggunakan pendekatan arsitektur modular untuk memastikan efisiensi dalam pengembangan, pemeliharaan, dan perluasan fitur. Pendekatan ini memungkinkan setiap komponen untuk berdiri sendiri namun tetap saling terintegrasi. Arsitektur ini terdiri dari empat komponen utama:

1. Core Module
   Komponen inti yang mengelola logika utama deteksi dan analisis aplikasi. *Core Module* berfungsi sebagai pusat pengolahan data, yang mencakup pemrosesan data dari sumber eksternal seperti *metadata* aplikasi dan penyajian hasil analisis secara langsung ke antarmuka pengguna.
2. Model
   Berisi definisi struktur data, seperti entitas aplikasi, perizinan, dan hasil analisis. Struktur data ini dirancang untuk memfasilitasi pengolahan data secara terorganisir dan konsisten.
3. Utils
   Menyediakan fungsi pembantu seperti validasi data, *logging*, pengelolaan waktu, dan manajemen konstanta. Modul ini bertindak sebagai pendukung untuk menjaga efisiensi tanpa mengganggu modularitas sistem inti.

e- ISSN: 2828–299X

3.2.2 Komponen Utama Library

SecurityScanner adalah kelas utama yang bertanggung jawab atas proses *scanning* dan analisis aplikasi. Fungsinya mencakup berbagai aktivitas penting, seperti deteksi perizinan berbahaya, di mana *SecurityScanner* mampu mengidentifikasi aplikasi yang menggunakan perizinan berisiko tinggi dengan memeriksa izin yang diminta aplikasi terhadap daftar izin berbahaya yang telah ditentukan. Selain itu, *SecurityScanner* juga mengelola daftar aplikasi yang terdeteksi sebagai berbahaya atau mencurigakan, sehingga pengguna dapat mengetahui risiko potensial. Komponen ini juga memanfaatkan *metadata*, seperti nama paket, versi, dan pengembang, untuk menilai risiko aplikasi dengan lebih akurat.

RiskAnalyzer adalah komponen yang bertugas menghitung tingkat risiko aplikasi berdasarkan perizinan yang diminta. Fungsi utama *RiskAnalyzer* meliputi perhitungan risiko, yang menggunakan algoritma berbasis kriteria untuk menghasilkan skor risiko, mempertimbangkan jumlah dan jenis perizinan yang diminta serta reputasi aplikasi. RiskAnalyzer juga mengklasifikasikan aplikasi ke dalam tingkat bahaya rendah, sedang, atau tinggi berdasarkan skor risiko yang dihasilkan. Selain itu, *RiskAnalyzer* memberikan rekomendasi tindakan kepada pengguna, seperti mencopot pemasangan aplikasi atau membatasi penggunaan perizinan tertentu.

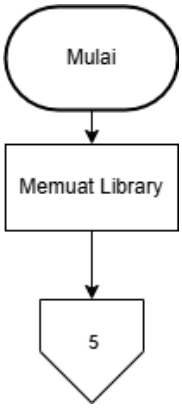
PermissionManager dirancang untuk memantau dan mengelola perizinan aplikasi. Komponen ini memiliki kemampuan untuk memverifikasi perizinan, memastikan konsistensi dan keabsahan izin yang diminta oleh aplikasi sesuai dengan kebutuhan fungsionalitas aplikasi. Selain itu, *PermissionManager* dapat mengidentifikasi perizinan berbahaya berdasarkan regulasi keamanan terkini, serta melakukan *monitoring* terhadap pola penggunaan perizinan selama aplikasi berjalan untuk mendeteksi aktivitas mencurigakan.

3.3 Alur Kerja Library

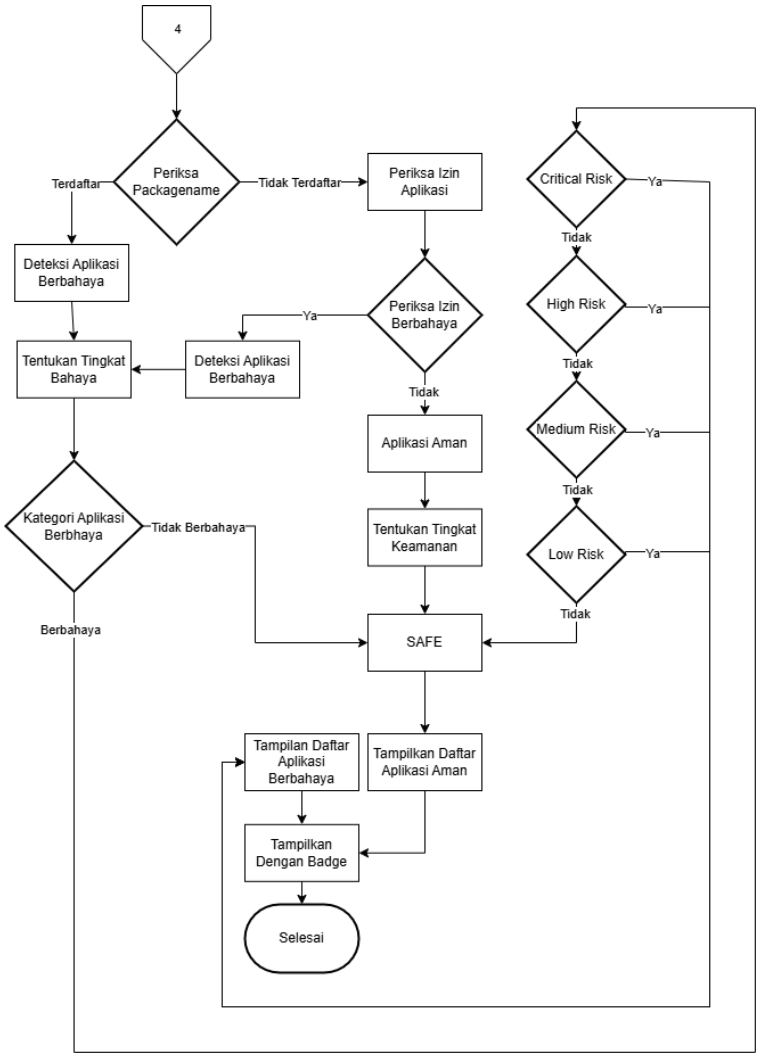
Pada tahap inisialisasi, library di inisialisasi dengan konteks Android yang diperlukan untuk mengakses data sistem. Beberapa aktivitas utama dalam tahap ini meliputi memuat daftar perizinan berbahaya, di mana *library* memuat data perizinan yang dianggap berbahaya berdasarkan regulasi dan pedoman keamanan terkini. Selain itu, komponen-komponen seperti *SecurityScanner*, *RiskAnalyzer*, dan *PermissionManager* di inisialisasi untuk mempersiapkan proses scanning dan analisis data. Tahap ini juga memastikan bahwa perangkat memiliki izin yang diperlukan untuk menjalankan fungsi library secara optimal melalui validasi lingkungan.

Pada tahap *scanning*, proses dilakukan untuk mengidentifikasi aplikasi dan perizinan yang digunakan. Langkah-langkah utama dalam tahap ini mencakup pengambilan daftar aplikasi terinstal, di mana *library* menggunakan *PackageManager* untuk mendapatkan daftar aplikasi yang terinstal pada perangkat. Selanjutnya, setiap perizinan yang diminta oleh aplikasi dianalisis terhadap daftar perizinan berbahaya yang telah dimuat sebelumnya. Nama paket aplikasi juga diperiksa terhadap daftar aplikasi yang telah diidentifikasi sebagai berbahaya berdasarkan data dari sumber terpercaya.

Tahap analisis adalah tahap terakhir di mana *library* melakukan analisis mendalam berdasarkan data yang telah dikumpulkan. Aktivitas utama dalam tahap ini meliputi perhitungan skor risiko, yang mempertimbangkan jumlah dan jenis perizinan yang diminta, pola penggunaan perizinan, dan reputasi aplikasi. Aplikasi kemudian dikategorikan berdasarkan tingkat risikonya menjadi rendah, sedang, atau tinggi. Terakhir, *library* menghasilkan rekomendasi tindakan untuk meningkatkan keamanan, seperti mencopot pemasangan aplikasi atau membatasi perizinan tertentu. Urutan rincian prosesnya secara jelas bisa dilihat pada *flowchart* berikut.



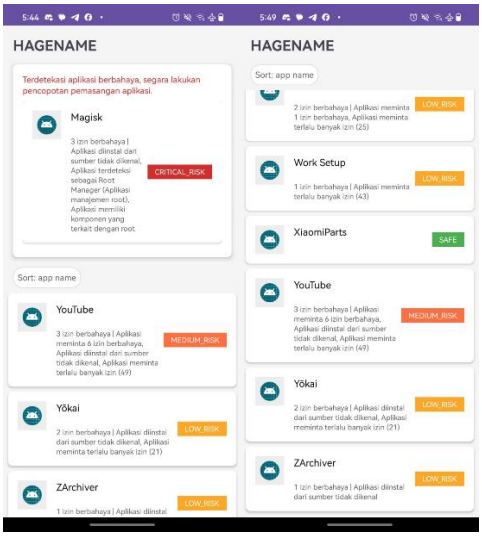
Gambar 2 Flowchart Proses Analisa Risiko A1



Gambar 3 Flowchart Proses Analisa Risiko A1

3.4 Hasil Perancangan Library Detekasi Aplikasi Berbahaya

Hasil dari rancangan *library* Deteksi Aplikasi Berbahaya yang sudah di integrasikan ke dalam aplikasi adalah sebagai berikut.



Gambar 4 Tampilan Halam Utama Aplikasi

Gambar di atas menunjukkan tampilan halaman utama dari aplikasi uji coba untuk *library* yang telah dikembangkan. Pada halaman tersebut, terdapat nama aplikasi dengan ukuran yang cukup besar, serta sebuah kartu

yang menampilkan pesan peringatan jika terdeteksi nama paket (*package name*) yang terdaftar di *library*. Kartu tersebut berisi informasi tentang nama aplikasi dan perizinan yang telah terdeteksi, dilengkapi dengan label level risiko berwarna di sisi kanan. Di bawah kartu yang memuat aplikasi dengan peringatan risiko paling tinggi, terdapat daftar aplikasi lain yang memiliki berbagai level risiko yang tersusun rapi.

3.5 Pengujian Aplikasi

No.	Test Case	Teknik Testing	Input	Hasil Yang Diharapkan	Hasil Yang Diterima	Status Diterima
TC-1	Deteksi aplikasi dengan izin berbahaya	Equivalence Partitioning	Aplikasi memiliki izin android.permission .READ_SMS	Aplikasi terdeteksi sebagai berbahaya, risiko minimal Medium Risk, laporan menyebutkan izin berbahaya.	Aplikasi terdeteksi dan menampilkan level risiko	Valid
TC-2	Deteksi aplikasi tanpa izin berbahaya	Equivalence Partitioning	Aplikasi tanpa izin android.permission .READ_SMS atau izin berbahaya lainnya	Aplikasi terdeteksi sebagai aman dengan risiko SAFE.	Aplikasi menampilkan level risiko SAFE	Valid
TC-3	Aplikasi dengan jumlah izin tepat 15	Boundary Value Analysis	Aplikasi meminta tepat 15 izin	Aplikasi dianggap aman/tidak berlebihan, laporan tidak menyebutkan masalah pada jumlah izin.	Aplikasi terdeteksi aman	Valid
TC-4	Aplikasi dengan jumlah izin lebih dari 15	Boundary Value Analysis	Aplikasi meminta 16 izin	Aplikasi terdeteksi dengan masalah Excessive Permissions, risiko minimal Medium Risk, laporan menyebutkan jumlah izin berlebihan.	Aplikasi menampilkan dengan kartu dengan isi yang sesuai	Valid
TC-5	Deteksi aplikasi dari sumber tidak dikenal	Equivalence Partitioning	Aplikasi diinstal dari sumber tidak dikenal	Aplikasi terdeteksi sebagai berbahaya, risiko minimal Medium Risk, laporan menyebutkan masalah sumber instalasi.	Aplikasi terdeteksi berbahaya dan sudah ditampilkan	Valid
TC-6	Deteksi aplikasi dengan nama paket berbahaya	Equivalence Partitioning	Nama paket: com.topjohnwu.magisk	Aplikasi terdeteksi sebagai Root Manager, risiko Critical Risk, laporan menyebutkan aplikasi terdaftar dalam kategori berbahaya.	Aplikasi ditampilkan sebagai bahaya tingkat tertinggi	Valid
TC-7	Deteksi aplikasi tanpa nama paket berbahaya	Equivalence Partitioning	Nama paket tidak termasuk dalam daftar berbahaya	Aplikasi terdeteksi sebagai aman, risiko SAFE.	Aplikasi ditampilkan dengan label SAFE karena tidak termasuk dalam daftar	Valid
TC-8	Pemrosesan banyak aplikasi (beban tinggi)	Boundary Value Analysis	Total aplikasi yang di-scan: 100+	Library tetap dapat memproses semua aplikasi tanpa error, waktu pemrosesan	Aplikasi berjalan tanpa error dan waktu yang	Valid

No.	Test Case	Teknik Testing	Input	Hasil Yang Diharapkan	Hasil Yang Diterima	Status Diterima
				dalam batas wajar (< 5 detik per aplikasi).	dibutuhkan sesuai	
TC-9	Pemrosesan satu aplikasi (beban rendah)	Boundary Value Analysis	Total aplikasi yang di-scan: 1	Library tetap memproses aplikasi dengan benar, laporan dibuat untuk satu aplikasi, risiko sesuai dengan analisis keamanan aplikasi tersebut.	Aplikasi melakukan scan terhadap semua aplikasi yang ada tidak hanya 1	Tidak Valid
TC-10	Deteksi aplikasi dengan jenis izin ambigu	Equivalence Partitioning	Aplikasi dengan izin yang tidak termasuk dalam daftar izin berbahaya	Aplikasi dianggap aman, risiko tetap SAFE, laporan tidak menyebutkan izin sebagai masalah karena tidak termasuk dalam daftar izin berbahaya	Aplikasi ditampilkan sebagai SAFE	Valid

Pengujian pada aplikasi yang telah di buat dengan *library* memiliki berbagai hasil yang ada. Pada kasus yang menerima hasil "Tidak Valid" karena aplikasi dikodekan untuk menampilkan semua daftar aplikasi tanpa terkecuali. Pengujian dilakukan pada beberapa skenario utama yang dirancang untuk menguji fungsi dari library deteksi izin dan packagename pada perangkat Android. Terdapat 10 kasus uji yang dijalankan menggunakan metode blackbox testing.

Hasil pengujian menunjukkan bahwa 10 dari 10 kasus uji menghasilkan keluaran sesuai dengan ekspektasi yang telah ditentukan (valid), sehingga dapat disimpulkan bahwa library ini telah bekerja sesuai dengan kebutuhan dan spesifikasi fungsional yang dirancang.

Adapun pengujian ini memanfaatkan teknik Equivalence Partitioning dan Boundary Value Analysis, di mana setiap kasus uji difokuskan pada pengolahan input tertentu, deteksi aplikasi berisiko, serta penilaian performa library dalam kondisi beban rendah maupun tinggi. Berikut adalah rumus perhitungan tingkat keberhasilan pengujian blackbox testing yang digunakan:

$$pengujian\ sukses = \frac{9}{10} \times 100\%$$

pengujian sukses 90%

Dari hasil pengujian dan perhitungan dapat disimpulkan bahwa success rate dari pengujian yang dilakukan adalah sebesar 90%. Dengan begitu, dapat disimpulkan bahwa pengujian yang telah dilakukan menunjukkan tingkat validitas yang cukup baik. Selanjutnya, validitas ini dapat menjadi indikator untuk melakukan analisis lebih mendalam terhadap hasil pengujian. Jika success rate berada di atas 75%, maka sistem dianggap memenuhi standar minimum untuk diterima. Namun, masih ada ruang untuk perbaikan, terutama dalam beberapa kasus yang menunjukkan hasil invalid.

4. SIMPULAN

Penelitian ini berhasil mengembangkan *library* untuk mendeteksi aplikasi berbahaya pada perangkat Android, yang berfokus pada analisis izin aplikasi dan identifikasi nama paket berisiko. *Library* ini memiliki kelebihan dalam arsitektur modular yang memungkinkan pengembang untuk dengan mudah mengintegrasikan dan menggunakan komponen seperti *SecurityScanner, RiskAnalyzer,* dan *PermissionManager*. Hal ini berkontribusi pada peningkatan kesadaran pengguna dan pengembang tentang potensi risiko keamanan aplikasi.

Namun, terdapat beberapa kekurangan, seperti keterbatasan dalam mendeteksi semua jenis aplikasi berbahaya dan kebutuhan untuk pembaruan berkala agar tetap relevan dengan ancaman keamanan yang terus berkembang. Untuk pengembangan selanjutnya, disarankan untuk meningkatkan algoritma deteksi, menambahkan fitur analisis yang lebih mendalam, serta melakukan integrasi dengan platform lain untuk memperluas jangkauan deteksi. Hasil dari penelitian ini sangat penting dalam konteks keamanan perangkat Android, mengingat meningkatnya jumlah aplikasi berbahaya yang dapat mengancam data pribadi pengguna, sehingga *library* ini

diharapkan dapat menjadi alat yang berguna bagi pengembang dan pengguna dalam menjaga keamanan perangkat mereka..

5. SARAN

Untuk menutup kekurangan yang ada dalam penelitian ini, beberapa saran untuk penelitian lebih lanjut dapat dipertimbangkan. Pertama, peningkatan algoritma deteksi sangat penting untuk meningkatkan akurasi dalam mengidentifikasi aplikasi berbahaya, termasuk penerapan teknik *machine learning* atau analisis perilaku aplikasi. Kedua, penambahan fitur analisis yang lebih mendalam, seperti analisis kode sumber atau pemantauan aktivitas aplikasi secara real-time, dapat membantu mendeteksi perilaku mencurigakan dengan lebih efektif. Selanjutnya, integrasi *library* dengan platform lain, seperti sistem keamanan berbasis *cloud*, akan memperluas jangkauan deteksi dan memberikan perlindungan yang lebih komprehensif bagi pengguna. Selain itu, implementasi sistem pembaruan berkala untuk *library* sangat diperlukan agar tetap relevan dengan ancaman keamanan yang terus berkembang. Terakhir, melakukan pengujian *library* pada berbagai versi Android akan memastikan kompatibilitas dan efektivitas deteksi di seluruh perangkat yang menggunakan sistem operasi Android.

DAFTAR PUSTAKA

[1] R. B. O. Bagus Aji Saputro, Lisan Iqbal Alfitra, "Analisis Malware Android Menggunakan Metode Reverse Engineering," *J. Ilm. Dan Karya Mhs.*, vol. 1, no. 2, hal. 41–53, 2024, doi: 10.54066/jikma-itb.v1i2.169.

[2] M. T. Anandika Nur Iman, Avon Budiyono, S.T., M.T., Ahmad Almaarif, S.Kom., "ANALISIS MALWARE PADA SISTEM OPERASI ANDROID MENGGUNAKAN PERMISSION-BASED MALWARE ANALYSIS IN ANDROID OPERATION SYSTEM USING PERMISSION-BASED," *e-Proceeding Eng.*, vol. 6, no. 2, hal. 7845–7851, 2019.

[3] R. A. Sari, M. Sutrisno, A. Rahman, dan M. N. Al Kodri, "Penerapan Model Research and Development Untuk Media Belajar Desain Grafis Berbasis Android," *J. Sist. Informasi, Teknol. Inf. dan Komput.*, vol. 13, no. 2, hal. 100–111, 2023.

[4] Q. Widayati dan M. Nasir, "Metode Mobile-D Dalam Rancang Bangun Perangkat Lunak Kamus Istilah Ekonomi," *J. Ilm. Matrik (Ilmu Komputer)*, vol. 20, no. 1, hal. 51–60, 2018.

[5] P. Abrahamsson *et al.*, "Mobile-D: An agile approach for mobile application development," *Proc. Conf. Object-Oriented Program. Syst. Lang. Appl. OOPSLA*, no. September, hal. 174–175, 2004, doi: 10.1145/1028664.1028736.

[6] G. V. Villanueva, R. J. B. Beltran, dan R. R. V. Garcia, "Proposal of a mobile application using the Mobile-D methodology to support the type 2 diabetes control process in a hospital in Trujillo," *Proc. 2022 IEEE Eng. Int. Res. Conf. EIRCON 2022*, no. October, 2022, doi: 10.1109/EIRCON56026.2022.9934098.