PENGEMBANGAN APLIKASI WEB *MACHINE LEARNING* BERBASIS WEB UNTUK SERVER PENDETEKSI MALWARE ANDROID

Rinanza Zulmy Alhamri[1], Riswanda Al Farisi[2], Toga Aldila Cinderatama[3], Kunti Eliyen[4]

[1,2,3,4] Manajemen Informatika, Teknologi Informasi, Politeknik Negeri Malang, Indonesia
[1]rinanza.z.alhamri@polinema.ac.id, [2]riswanda.al@polinema.ac.id, [3] toga.aldila@polinema.ac.id, [4]kunti.eliyen@polinema.ac.id

Abstrak

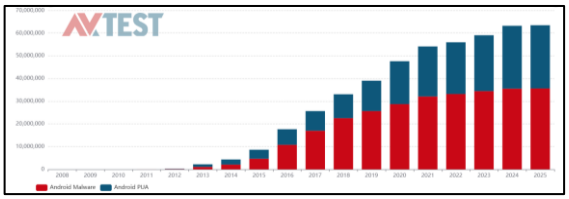
Meningkatnya jumlah perangkat Android berbanding lurus dengan meningkatknya ancaman *malware* pada sistem Android. Peningkatan jumlah *malware* pada sistem Android saat ini menuntut solusi berupa sistem deteksi *malware* yang andal. Beberapa penelitian sebelumnya telah melakukan analisis statis dalam mendeteksi *malware* Android menggunakan *Machine Learning* (ML) yang fokus pada perbandingan dan peningkatan akurasi. Namun sampai saat ini belum ada penelitian yang mengembangkan antarmuka *server* ML sebagai wadah operasional yang efektif untuk mendeteksi ini, khususnya yang memanfaatkan analisis *system calls.* Penelitian ini bertujuan untuk mengembangkan aplikasi *server Machine Learning* untuk mendeteksi *malware* Android berbasis web menggunakan *framework* Django. Penelitian ini menggunakan metode *Support Vector Machine* (SVM) untuk mengklasifikasikan jenis *malware*. Tahapan pengembangan perangkat lunak dilakukan menggunakan metode *Waterfall* yang meliputi studi literatur, analisis, perancangan, implementasi, dan pengujian. Untuk memvalidasi operasional sistem, dilakukan pengujian fungsional dengan pendekatan *Black-Box Testing*. Metode *black-box* digunakan untuk memastikan setiap fitur bekerja sesuai spesifikasi dari perspektif pengguna. Sistem ini menyediakan fungsi bagi administrator, meliputi: otentikasi pengguna (termasuk reset *password*), aktivasi deteksi menggunakan klasifikasi, pencarian data, klasifikasi otomatis, dan visualisasi statistik data melalui grafik batang. Hasil pengujian fungsional sistem dengan pendekatan *black-box testing* menunjukkan bahwa sistem web berhasil melakukan deteksi *malware* pada Android melalui klasifikasi otomatis yang diterapkan menggunakan ML. Semua fungsi utama sistem berjalan dengan valid dan efektif, serta berhasil memenuhi seluruh kebutuhan fungsional yang telah didefinisikan.

Kata kunci: black-box testing, deteksi malware android, machine learning, server

1. Pendahuluan

Malware masih menjadi salah satu ancaman paling serius dalam keamanan siber. Istilah *malware* merupakan singkatan dari *malicious software*, yaitu perangkat lunak berbahaya yang dirancang dan disebarkan oleh pihak yang tidak bertanggung jawab untuk merusak sistem, mencuri data, maupun mengganggu operasional perangkat yang diserang (Turaina et al., 2024). Melalui serangan *malware*, pelaku kejahatan siber dapat memperoleh akses awal ke dalam suatu sistem, yang selanjutnya dapat dimanfaatkan untuk berbagai tindakan lanjutan seperti pencurian data sensitif, penyalahgunaan informasi pribadi, hingga penghapusan atau manipulasi data secara sengaja. Seiring dengan meningkatnya aktivitas dan kompleksitas serangan siber di berbagai sektor, jumlah serta variasi *malware* yang beredar juga terus mengalami peningkatan.

Perkembangan teknologi perangkat *mobile* turut memberikan peluang baru bagi pelaku kejahatan siber untuk menargetkan sistem operasi yang banyak digunakan oleh masyarakat. Salah satu sistem operasi yang menjadi target utama adalah Android. Popularitas Android sebagai sistem operasi *mobile* dengan jumlah pengguna yang sangat besar menjadikannya sasaran yang menarik bagi pengembang *malware*. Data yang dirilis oleh AV-TEST, sebuah lembaga riset independen di bidang keamanan teknologi informasi yang berbasis di Jerman, menunjukkan bahwa jumlah varian *malware* yang menargetkan sistem operasi Android sejak tahun 2012 hingga saat ini telah mencapai lebih dari 35 juta varian (AV-TEST, 2025) dapat dilihat pada Gambar 1.



Gambar 1. Jumlah *Malware* Android

Peningkatan jumlah *malware* tersebut menggambarkan bahwa ancaman terhadap perangkat Android terus berkembang dan membutuhkan pendekatan deteksi yang lebih efektif dan adaptif. Salah satu pendekatan yang banyak digunakan dalam mendeteksi *malware* adalah pemanfaatan *Machine*

Learning (ML). *Machine Learning* merupakan suatu metode dalam bidang kecerdasan buatan yang memungkinkan sistem untuk mempelajari pola dari data dan menghasilkan keputusan atau prediksi secara otomatis tanpa harus diprogram secara eksplisit secara berulang (Wijoyo et al., 2024) (Priastiwi & Adlin Sinaga, 2023). Dengan memanfaatkan teknik pembelajaran dari data, sistem ML dapat digunakan untuk mengidentifikasi pola perilaku *malware* serta membedakannya dari aplikasi normal (Togu Novriansyah Turnip et al., 2023). ML berlandaskan beberapa rumpun ilmu seperti matematika, statistika, dan data mining (Ahadi Ningrum, 2023).

Berbagai penelitian terkait deteksi *malware* Android menggunakan pendekatan ML telah dilakukan. Sebagian besar penelitian tersebut berfokus pada pengembangan model klasifikasi untuk meningkatkan tingkat akurasi deteksi *malware dengan memanfaatkan berbagai metode dan fitur* analisis, baik melalui pendekatan analisis statis maupun kombinasi beberapa algoritma pembelajaran mesin. Selain itu, beberapa penelitian juga telah mengembangkan aplikasi berbasis web untuk mengimplementasikan model *machine learning* sehingga dapat digunakan oleh pengguna melalui antarmuka sistem berbasis web denga Python (Alan Chandra Darmawan & Lizda Iswari, 2022).

Meskipun demikian, sebagian besar penelitian yang ada masih menitikberatkan pada peningkatan performa algoritma klasifikasi, seperti peningkatan akurasi model maupun optimasi metode ML. Pengembangan sistem yang berfungsi sebagai *server* pendeteksi *malware* dengan antarmuka yang memungkinkan pengguna untuk menjalankan serta memonitor proses deteksi *malware* secara langsung masih relatif terbatas. Padahal, keberadaan sistem *server* yang terintegrasi dengan model ML sangat penting untuk mendukung implementasi sistem deteksi *malware* yang lebih praktis dan mudah digunakan.

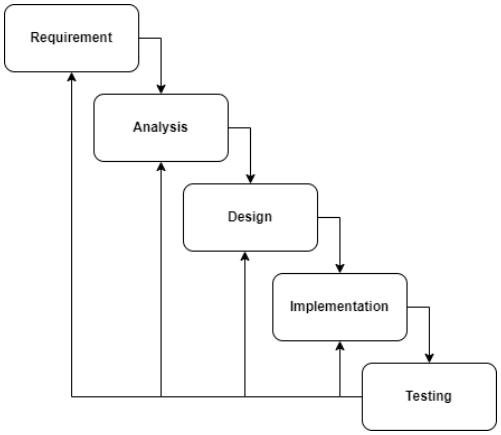
Berdasarkan kondisi tersebut, diperlukan suatu penelitian yang tidak hanya berfokus pada pengembangan model ML, tetapi juga pada pengembangan aplikasi *server* yang mampu mengintegrasikan model tersebut ke dalam sebuah sistem berbasis web. Sistem ini diharapkan dapat berfungsi sebagai *server* pendeteksi *malware* Android yang memungkinkan pengguna untuk melakukan proses deteksi serta memantau kinerja sistem secara lebih mudah. Penelitian ini mencoba memberikan bukti bagaimana konsep deteksi *malware* pada Android memanfaatkan ML berdasarkan informasi *system calls* aplikasi yang dicurigai, bisa dilakukan secara empiris. Diharapkan hasil penelitian ini dapat memberikan kontribusi bagaimana penerapan deteksi *malware* pada Android memanfaatkan ML secara *client-server* melalui internet bisa berjalan.

Penelitian tentang deteksi *malware* Android memanfaatkan ML sebelumnya lebih menitikberatkan pada optimasi kinerja metode (Hadiprakoso, Rendra Aditya, & Pramitha, 2022)(Zhang et al., 2022), sedangkan penelitian ini memberikan penjelasan bagaimana ML deteksi *malware* diterapkan sebagai *server*. Penelitian mengenai penerapan ML sebagai *server* telah dilakukan seperti metode Decission Tree (Alan Chandra Darmawan & Lizda Iswari, 2022) dan *Multilayer Perception* (Ahadi Ningrum, 2023) menggunakan Python Flask serta metode *Naïve Bayes* menggunakan Python Django. Berbeda dengan penelitian sebelumnya, penelitian ini menerapkan aplikasi server ML bermetode *Support Vector Machine* (SVM) berbasis web menggunakan bahasa pemrograman Python dengan *framework* Django untuk mendeteksi *malware* Android.

Dalam penelitian ini, pengembangan aplikasi dilakukan menggunakan bahasa pemrograman Python dengan memanfaatkan *framework* Django. *Framework* ini dipilih karena menyediakan berbagai fitur yang mendukung pengembangan aplikasi web yang terstruktur dan aman, salah satunya melalui penggunaan *Object Relational Mapper* (ORM) yang memudahkan pengelolaan basis data serta meningkatkan keamanan dalam pengembangan aplikasi *server* (Ramadhan et al., 2023). Melalui penelitian ini diharapkan dapat dihasilkan sebuah aplikasi *machine learning* berbasis web yang berfungsi sebagai *server* pendeteksi *malware* Android. Sistem yang dikembangkan diharapkan mampu mempermudah pengguna dalam menjalankan model deteksi *malware* sekaligus melakukan pemantauan terhadap proses deteksi yang dilakukan oleh sistem.

2. Metode

Penelitian ini menggunakan metode pendekatan *Waterfall*, metode ini dipilih karena memberikan kerangka kerja yang sistematis dan sekuensial untuk pengembangan sistem (Jibran et al., 2025). Tahapan yang dilaksanakan mengikuti urutan linear, meliputi *requirement, analysis, design, implementation, dan testing*. Tahapan pelaksanaan menggunakan metode *waterfall* dapat dilihat pada Gambar 2.



Gambar 2. Metode Waterfall

2.1 *Support Vector Machine*

Penelitian ini memanfaatkan metode Support Vector Machine (SVM) dalam dalam mendeteksi *malware* pada Android. Metode SVM adalah metode ML tersupervisi yang memanfaatkan kernel berdimensi tinggi untuk memisahkan dan mengklasifikasikan data. SVM digunakan untuk mendeteksi aplikasi Android yang mencurigakan berdasarkan aktivitas *system call* aplikasi tersebut. *System call* sendiri adalah layanan dasar sistem operasi yang diakses oleh aplikasi, di mana menjadi representasi dari perilaku aplikasi Android itu sendiri.

Metode SVM dijalankan dengan membuat model SVM deteksi *malware* Android. Sebagai *milestone* penelitian ini telah dikembangkan model deteksi *malware* Android sebelumnya menggunakan SVM berdasarkan *system call* merujuk pada ARM(32-bit/EABI (Alhamri et al., 2024). Model dikembangkan seperti pada Gambar 3, dimulai dengan mengumpulkan dataset *system call* berlabel dari 50 aplikasi jinak dan 50 aplikasi *malware* di mana setiap *system call* diambil 3000 data aktivitas. Kemudian 80% dataset diterapkan beberapa metode ML meliputi SVM, *Decision Tree* (DT), *K-Nearest Neighbor* (KNN), dan *Naïve Bayes* (NB) menggunakan pustaka scikit-learn pada pemrograman Python. Terakhir model diuji kinerja klasifikasinya menggunakan 20% dataset berdasarkan akurasi, *precision, recall,* dan f1*score*. Tahapan pengembangan model SVM dapat dilihat pada Gambar 3.

Gambar 3. Tahapan Pengembangan Model SVM

Hasil pengujian kinerja deteksi *malware* Android memperlihatkan bahwa SVM sama unggulnya dengan DT dengan akurasi sama-sama 79% dibandingkan KNN dan NB. Sebagai pembanding lainnya, penelitian (Mochammad Anshori et al., 2019) menyimpulkan bahwa SVM sama unggulnya dengan KNN dalam mendeteksi *malware* Android dengan akurasi sama-sama 71,67% dibandingkan DT dan NB. Sedangkan penelitian (Zhang et al., 2022) menyimpulkan bahwa SVM merupakan metode paling unggul dengan akurasi 99% dibandingkan DT, KNN, dan NB. Metode SVM yang unggul dalam mendeteksi *malware* Android berdasarkan penelitian-penelitian sebelumnya tersebut menjadi acuan penelitian kali ini untuk menerapkannya dalam bentuk aplikasi *server* berbasis web.

Model SVM yang telah dikembangkan tersebut diekspor untuk dapat digunakan sebagai *server* ML pada lingkungan Python khususnya *framework* Django. Sebagai *client* telah dilakukan penelitian untuk mengembangkan aplikasi antarmuka deteksi *malware* Android (Alhamri et al., 2025) pada telepon pintar Android kondisi *root*. Sebagai *client*, aplikasi antarmuka deteksi *malware* dapat mengambil aktivitas *system call* aplikasi yang dicurigai berdasarkan *process identifier* dalam bentuk *.txt* untuk kemudian mengirimnya ke *cloud* untuk dianalisa.

2.2 Studi Literatur

Mengumpulkan informasi penerapan *framework* Django untuk pengembangan aplikasi ML sebagai *server* pendeteksi *malware* berbasis Android meliputi:

a. Data primer – melakukan pengambilan data studi literatur pada artikel, jurnal, serta material online penerapan *framework* Django dalam mengembangkan aplikasi *server* pendeteksi *malware* berbasis Android.
b. Data skunder – melakukan pengambilan data studi literatur pada artikel, jurnal, serta material online mengenai kerangka kerja dan konsep deteksi *malware* berbasis Android menggunakan metode ML yaitu SVM.

2.3 Analisis

Analisis dilakukan untuk mengkaji bagaimana aplikasi ML berbasis web bisa dikembangkan dengan menggunakan *framework* Django sesuai dengan kebutuhan. Berikut langkah-langkah analisis sistem meliputi:

a. Analisis Masalah

Sesuai dengan penelitian sebelumnya, telah dikembangkan kerangka sistem dalam mendeteksi aplikasi *malware* berbasis Android serta aplikasi pendeteksi *malware* berbasis Android sebagai antarmuka klien. Terdapat *server* dimana sebagai *supervised machine learning* yang menggunakan metode SVM dalam klasifikasi *malware* berbasis Python. ML telah dilatih dan dapat mengklasifikasikan data *tracing system calls* melalui penyimpanan *cloud Firebase Real-time Database* dan *Storage*. Saat ini *server* pendeteksi *malware* Android dikembangkan sebagai *back-end* menggunakan bahasa pemrograman Python. Secara otomatis *server* pendeteksi *malware* Android bekerja ketika *server* diaktifkan. Permasalahannya adalah belum adanya fungsi antarmuka untuk melakukan monitoring dan pengelolaan histori deteksi *malware* Android pada aplikasi *server* ML saat ini.

b. Pemecahan Masalah

Agar kinerja ML dalam mendeteksi *malware* berbasis Android bisa dijalankan dan dimonitor secara mudah oleh pengguna maka dikembangkan aplikasi berbasis web pada *server* pendeteksi *malware* Android. Adapun proses bisnis pemecahan masalah dilakukan dengan pengguna bisa otentikasi, pengaktifan deteksi *malware*, serta melihat statistik.

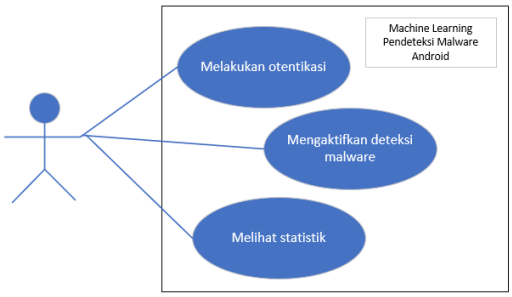
2.4 Perancangan

Berikut ini dijelaskan rancangan untuk pengembangan aplikasi ML berbasis web untuk *server* pendeteksi *malware* Android ditampilkan dengan diagram *use case*, dan arsitektur sistem.

a. Diagram Use Case

Unified Modeling Language (UML) merupakan sebuah bahasa pemodelan standar yang banyak digunakan di berbagai industri untuk membantu proses analisis, perancangan, dan pendefinisian kebutuhan perangkat lunak (Ramdany et al., 2024). UML berfungsi sebagai media representasi visual yang memudahkan pengembang memahami struktur, perilaku, dan interaksi antar komponen dalam sebuah sistem (Siska Narulita et al., 2024). *Use Case Diagram* adalah salah satu elemen yang sangat penting dalam UML yang berperan untuk memvisualisasikan hubungan dan interaksi antara suatu sistem dengan pihak-pihak yang terlibat di dalamnya, yang disebut sebagai aktor (Umar et al., 2025).

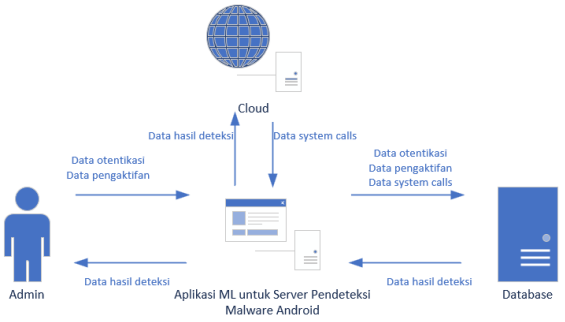
Melalui diagram ini, pengembang dapat menggambarkan secara jelas berbagai layanan, fitur, atau fungsi yang dimiliki oleh sistem, sekaligus menunjukkan bagaimana setiap aktor menggunakan dan berinteraksi dengan fungsi-fungsi tersebut (Eka et al., 2025). Dengan demikian, *Use Case Diagram* membantu memberikan pemahaman menyeluruh mengenai kebutuhan pengguna, ruang lingkup sistem, serta alur pemanfaatan sistem dari sudut pandang pengguna (Taufan et al., 2022). *Use Case Diagram* aplikasi dapat dilihat seperti pada Gambar 4.



Gambar 4. Use Case Diagram

b. Arsitektur Sistem

Pengguna hanya satu yaitu Admin yang secara langsung mengakses aplikasi ML berbasis web untuk mendeteksi *malware* Android. Aplikasi ML memperoleh data dari aplikasi klien melalui perantara penyimpanan *cloud* yaitu Firebase *Real-time Database*. Arsitektur sistem dapat dilihat seperti pada Gambar 5.



Gambar 5. Arsitektur Sistem

2.5 Implementasi

Implementasi aplikasi ML berbasis web untuk *server* pendeteksi *malware* Android dilakukan dengan tiga sub tahapan meliputi konfigurasi firebase, konfigurasi django, dan pengembangan aplikasi *server*. Pengembangan aplikasi dilakukan menggunakan bahasa pemrograman Python dengan *framework* Django serta penerapan *library* ML pada sistem. Bahasa pemrograman python adalah bahasa pemrograman tingkat tinggi yang bersifat interpretatif dan digunakan untuk berbagai keperluan umum. Bahasa ini menekankan penggunaan indentasi untuk meningkatkan keterbacaan kode serta mendukung pemrograman berorientasi objek, sehingga memudahkan pengembang dalam menulis program yang terstruktur dan mudah dipahami, baik untuk proyek kecil maupun besar (Manoj Kumar & Dr Rainu Nandal, 2024).

Konfigurasi Firebase digunakan untuk menyediakan *library* Firebase terutama yang berkaitan dengan *storage* dan *database real time*. Konfigurasi Django dilakukan untuk membuat sistem aplikasi antar muka dengan menggunakan arsitektur *Model-View-Template* (MVT) (Sabita et al., 2022). Tahap implementasi sistem dapat dilihat seperti pada Gambar 6.

Gambar 6. Tahap Implementasi

2.6 Pengujian

Pada tahapan ini, sistem yang telah dikembangkan akan melalui proses pengujian secara menyeluruh dengan tujuan utama untuk memastikan bahwa seluruh fitur dan fungsi yang tersedia dapat beroperasi dengan baik dan sesuai dengan yang diharapkan. Proses pengujian ini dilakukan dengan menerapkan metode *Black Box Testing*, khususnya dengan pendekatan *Functional Testing* yang berfokus pada pengujian fungsi-fungsi sistem tanpa melihat struktur atau kode program di dalamnya (Muhammad Helmi Satria Fedianto et al., 2023). Melalui pendekatan ini, setiap fungsi diuji berdasarkan masukan dan keluaran yang dihasilkan untuk

memastikan kesesuaiannya dengan kebutuhan serta spesifikasi yang telah dirancang pada tahap sebelumnya. rancangan pengujian sistem yang disusun menggunakan metode *Black Box Testing* sebagai acuan dalam pelaksanaan pengujian disajikan pada Tabel 1.

Tabel 1. Rancangan Pengujian Sistem

No	Fitur	Skenario	Harapan
1	Otentikasi	Admin melakukan otentikasi	Otentikasi berhasil dan terbuka *landing page*
2	Deteksi *Malware*	Admin mengaktifkan deteksi *malware*	Deteksi *malware* Android berhasil berjalan dan hasil dapat ditampilkan
3	Menampilkan Statistik	Admin melihat statistik	Menampilkan hasil deteksi dalam sebuah tabel dan grafik

3. Hasil dan Pembahasan

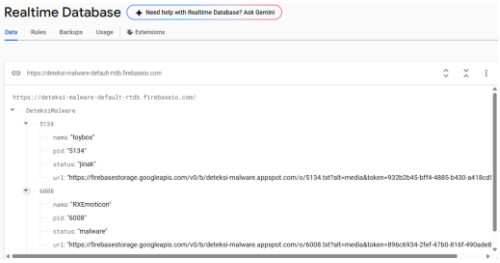
Penelitian ini menghasilkan sebuah sistem berbasis website untuk mendeteksi *malware* android.

3.1 Hasil Konfigurasi Database

Data pelacakan *system calls* dari aplikasi Android yang mencurigakan (digunakan sebagai acuan deteksi *malware*) dikirim dari aplikasi pendeteksi *malware* Android (dikembangkan pada penelitian sebelumnya) menuju *server* ML melewati *cloud* Firebase. Konfigurasi Firebase dilakukan dengan menyiapkan layanan Firebase meliputi *Real-time Database, Storage*, dan *Authentication*.

a. Real-time Database

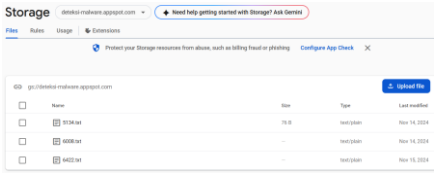
Real-time Database digunakan untuk menyimpan informasi data *system calls* aplikasi mencurigakan yang dikirim dari aplikasi pendeteksi *malware* Android. Informasi yang disimpan meliputi nama aplikasi, *process identifier*, status, kemudian alamat URL file txt pada Firebase *Storage* seperti pada Gambar 7.



Gambar 7. Real-time Database

b. Storage

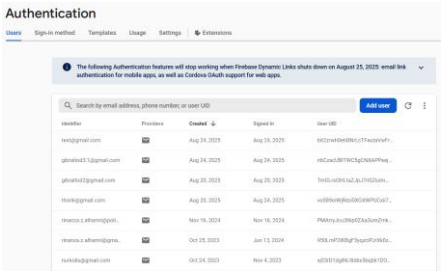
Storage digunakan untuk menyimpan data hasil pelacakan system calls yang sebenarnya berformat .txt. Penyimpanan file .txt pada storage menjadi acuan field URL pada *Real-time Database*. File .txt ini nantinya akan diambil oleh *server* ML dimana akan digunakan sebagai acuan dalam mengklasifikasikan apakah aplikasi Android yang dicurigai termasuk jinak atau *malware*. Tampilan hasil pelacakan *system calls* dapat dilihat pada menu storage seperti pada Gambar 8.



Gambar 8. Firebase Storage

c. Authentication

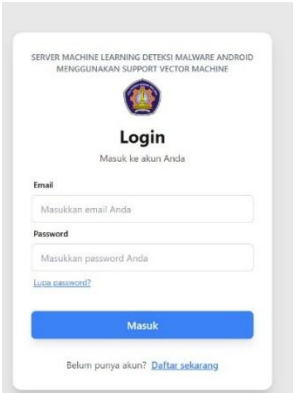
Layanan *Authentication* pada Firebase digunakan untuk mengaktifkan fungsi otentikasi pada *server* ML yang telah dikembangkan. Agar dapat menggunakan aplikasi pendeteksi *malware* Android, perlu melakukan otentikasi dengan layanan Google dimana Firebase *Authentication* ini menjadi acuan untuk proses tersebut. Pada aplikasi *server* ML pada penelitian ini, untuk dapat mengaktifkan server juga harus melakukan otentikasi dengan mengacu pada Firebase *Authentication* seperti pada Gambar 9.



Gambar 9. Authentication

3.2 Hasil Tampilan Otentikasi

Fungsi melakukan otentikasi digunakan agar user tertentu yang dapat mengaktifkan *server* ML. Diharapkan pada penelitian berikutnya, aplikasi *server* dapat ditaruh di *Virtual Private Server* secara *cloud* sehingga *server* bisa diakses secara *online*. Apabila *server* diakses secara *online* maka perlu ada manajemen *user*, dimana *user* tertentu yang dapat mengaktifkan *server* ML untuk deteksi *malware* Android. Fungsi otentikasi mengacu pada data *user* yang telah terdaftar pada Firebase *Authentication*. Tampilan *login* user dapat dilihat seperti pada Gambar 10.



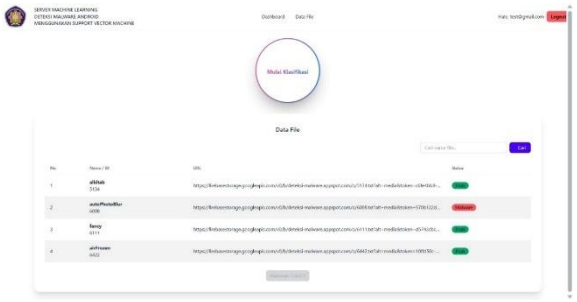
Gambar 10. Tampilan Otentikasi

3.3 Hasil Fungsi Mengaktifkan Deteksi *Malware*

Penelitian sebelumnya membahas tentang analisis statis pendeteksi *malware* pada Android menggunakan *supervised machine learning* (Hadiprakoso, Rendra Aditya, Pramitha, et al., 2022). Penelitian tersebut hanya melakukan komparasi metode supervised machine learning meliputi *Support Vactor Machine* (SVM*), Naïve Bayes* (NB), *Decision Tree* (DT), dan *K-Nearest Neighbors* (KNN) untuk melalukan pendeteksian *malware* Android. Penelitian tersebut dilakukan untuk mengetahui metode mana yang memiliki tingkat akurasi klasifikasi terbaik. Hasilnya metode SVM menjadi metode terbaik dalam mendeteksi *malware* Android secara analisis statis.

Penelitian ini fokus pada implementasi penerapan ML menggunakan metdoe SVM pada sebuah aplikasi website pendeteksi *malware* android. Fungsi mengaktifkan deteksi *malware* digunakan untuk mengaktifkan *server* agar melakukan proses klasifikasi menggunakan metode (SVM). Gambar 11 merupakan tampilan dalam mengaktifkan *server* machine learning dalam mendeteksi *malware*. Secara *default*, fungsi ini mengambil data-data yang telah tersimpan pada Firebase *Real-time Database*. Sebelum diklasifikasikan, data baru yang masuk *server* (data aplikasi mencurigakan baru) memiliki atribut Status yang masih kosong. Apabila tombol Mulai Klasifikasi diaktifkan, maka *server* akan mengambil *file* .txt pada data baru yang masuk tersebut untuk kemudian diklasifikasikan apakah termasuk aplikasi *malware* atau jinak. Hasil klasifikasi akan dikirim kembali menuju *Real-time Database* kembali dan akan bisa dilihat pada aplikasi deteksi *malware* Android (sebagai klien).

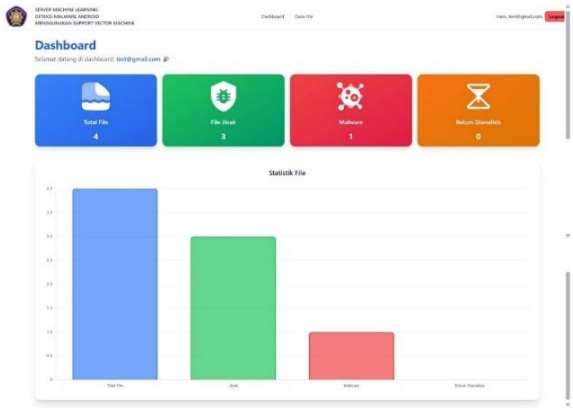
Dalam tabel data file seperti pada Gambar 11, terdapat 4 kolom meliputi nomor, nama atau id aplikasi, URL data *text* hasil *strace* aplikasi, dan status. Apabila tombol Mulai Klasifikasi diaktifkan, maka aplikasi *server* akan terus melakukan klasifikasi terus menerus jika ada data baru. Sedangkan data lama yang telah memiliki status akan diabaikan. Pengguna juga dapat untuk melakukan pencarian nama aplikasi dapat dilakukan pada *form Search*.



Gambar 11. Tampilan Aktifasi Deteksi *Malware*

3.4 Hasil Fungsi Melihat Statistik

Fungsi melihat statistik dilakukan dengan menghitung jumlah data masuk, kemudian menghitung berapa data aplikasi yang jinak, dan berapa data aplikasi yang termasuk *malware*. Warna biru menandakan jumlah total data yang masuk, warna hijau menandakan jumlah data *file* yang jinak, warna merah menandakan *file malware* dan warna orange menandakan *file* yang belum dianalisis atau diklasifikasikan. Untuk melihat statistik, tampilan statistik berada pada menu *Dashboard* dimana selain terdapat statistik data masuk juga terdapat *email* dari admin pengelola *server* serta grafik statistik berbentuk grafik bar. Tampilan data statistik perhitungan data yang masuk dapat dilihat pada Gambar 12.



Gambar 12. Tampilan Data Statistik

3.5 Hasil Pengujian

Pengujian sistem dilakukan menggunakan metode *Black-Box Testing*. Metode tersebut akan menguji sistem secara fungsional untuk mengetahui hasil semua kinerja sistem. Hasil pengujian pada sistem web *server* pendeteksi *malware* Android menunjukkan bahwa semua skenario pengujian fitur sistem telah berhasil sesuai harapan yang telah ditentukan. Hasil pengujian dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pengujian

No	Fitur	Skenario	Harapan	Hasil
1	Otentikasi	Admin melakukan otentikasi	Otentikasi berhasil dan terbuka *landing page*	Sesuai
2	Deteksi *Malware*	Admin mengaktifkan deteksi *malware*	Deteksi *malware* Android berhasil berjalan dan hasil dapat ditampilkan	Sesuai
3	Menampilkan Statistik	Admin melihat statistik	Menampilkan hasil deteksi dalam sebuah tabel dan grafik	Sesuai

Berdasarkan hasil pengujian yang telah dilakukan, seluruh fitur pada aplikasi dapat berjalan sesuai dengan skenario pengujian yang telah ditentukan. Proses unggah file dapat dilakukan dengan baik, sistem mampu memproses data yang

diberikan oleh pengguna, serta hasil deteksi *malware* dapat ditampilkan secara benar melalui antarmuka aplikasi berbasis web. Pengujian menggunakan pendekatan *blackbox testing* juga memastikan bahwa interaksi antara pengguna dan sistem berjalan dengan baik. Setiap *input* yang diberikan oleh pengguna dapat diproses oleh sistem dan menghasilkan *output* yang sesuai dengan yang diharapkan. Dengan demikian, sistem yang dikembangkan dapat digunakan sebagai sarana untuk menjalankan model deteksi *malware* secara lebih praktis melalui antarmuka berbasis web.

Jika dibandingkan dengan penelitian sebelumnya, sebagian besar penelitian terkait deteksi *malware* Android lebih berfokus pada pengembangan dan evaluasi model ML, khususnya pada peningkatan nilai akurasi, *precision, recall*, maupun F1-*score* dari model yang digunakan. Sebagai contoh, penelitian yang sebelumnya melakukan analisis statis untuk mendeteksi *malware* Android dengan membandingkan beberapa metode *supervised machine learning* seperti *Support Vector Machine* (SVM*), Naïve Bayes, Decision Tree,* dan *K-Nearest Neighbor.* Hasil penelitian tersebut menunjukkan bahwa metode SVM memiliki performa terbaik dalam mendeteksi *malware* Android melalui pendekatan analisis statis (Hadiprakoso, Rendra Aditya, Pramitha, et al., 2022).

Dengan demikian, hasil pengujian pada penelitian ini menunjukkan bahwa sistem yang dikembangkan telah berhasil mengimplementasikan *machine learning* dengan motode SVM ke dalam sebuah aplikasi *server* berbasis web yang dapat digunakan untuk menjalankan proses deteksi *malware* Android. Keberhasilan pengujian menggunakan metode *blackbox testing* menunjukkan bahwa sistem mampu menjalankan seluruh fungsi utama sesuai dengan kebutuhan pengguna, sehingga dapat menjadi dasar dalam pengembangan sistem deteksi *malware* yang lebih aplikatif dan mudah digunakan.

4. Kesimpulan

Penelitian ini telah berhasil mengembangkan aplikasi *Machine Learning* Berbasis Web untuk *Server* Pendeteksi *Malware* Android dengan satu pengguna yaitu admin. Fungsi-fungsi meliputi otentikasi dengan fitur lupa atau reset password, bisa mengaktifkan deteksi *malware* dengan fitur klasifikasi ML model SVM, pencarian data, serta klasifikasi otamatis, dan bisa melihat statistik data masuk yang ditampilkan dengan grafik batang. Hasil pengujian fungsional sistem menggunakan metode *blackbox testing* menunjukkan bahwa sistem mampu memenuhi semua hasil skenario pengujian, yaitu meliputi proses otentikasi, proses deteksi *malware* secara otomatis dan menampilkan data statistik hasil deteksi *malware* Android.

Adapun saran yang bisa diberikan untuk penelitian berikutnya adalah melakukan *deployment* aplikasi *server* pendeteksi *malware* Android menuju jaringan *cloud* pada *Virtual Private Server* sehingga *server* benar-benar bekerja optimal sebagai *Infrastructure as a Service*. Penelitian berikutnya juga dapat menerapkan metode *deep learning* dalam meningkatkan hasil klasifikasi pendeteksi *malware* android.

Daftar Pustaka:

AV-TEST, "AV Atlas," AV-TEST , 29 Januari 2025. [Online]. Available: https://portal.av-atlas.org/malware/statistics. [Accessed 29 Januari 2025].

Ahadi Ningrum, A. (2023). Penerapan Framework Flask Pada Machine Learning Dalam Memprediksi Umur Transformer. *KONVERGENSI*, *19*(2), 51–59.

Alan Chandra Darmawan, & Lizda Iswari. (2022). Pengembangan Aplikasi Berbasis Web dengan Python Flask untuk Klasifikasi Data Menggunakan Metode Decision Tree C4.5. *Jurnal Pendidikan Dan Konseling*, *4*.

Alhamri, R. Z., Cinderatama, T. A., Eliyen, K., & Izzah, A. (2024). Supervised Learning Methods Comparison for Android Malware Detection Based on System Calls Referring to ARM (32-bit/EABI) Table. *Journal of Information Technology and Cyber Security*, *2*(1), 15–24. https://doi.org/10.30996/jitcs.10511

Alhamri, R. Z., Eliyen, K., Cinderatama, T. A., & Heriadi, A. (2025). Pengembangan Aplikasi Pendeteksi Malware Berbasis Android Menggunakan Perintah Strace. *JIP (Jurnal Informatika Polinema)*, *12*.

Eka, R., Dani, K., & Rismayana, A. H. (2025). Sistem Pelaporan Kerusakan Sarana Prasarana Berbasis Mobile Secara Realtime Dengan Notifikasi. *JIP (Jurnal Informatika Polinema)*, *12*.

Hadiprakoso, R. B., Rendra Aditya, W., & Pramitha, F. N. (2022). Analisis Statis Deteksi Malware Android Menggunakan Algoritma Supervised Machine Learning. *Jurnal Cyber Security Dan Forensic Digital*, *5*(1), 1.

Hadiprakoso, R. B., Rendra Aditya, W., Pramitha, F. N., Siber, P., & Negara, S. (2022). Analisis Statis Deteksi Malware Android Menggunakan Algoritma Supervised Machine Learning. *CyberSecurity Dan Forensik Digital*, *5*(1), 1.

Jibran, S. M., Jannah, N., Irang, D., & Rahmani, P. (2025). Pengembangan Sistem Informasi Manajemen Penjualan Berbasis Website untuk Meningkatkan Efisiensi Operasional pada Toko Win Glowing dengan Metode Waterfall. *Journal of Human And Education*, *5*(1), 576.

Manoj Kumar, & Dr Rainu Nandal. (2024). Python's Role in Accelerating Web Application Development with Django. *International Research Journal on Advanced Engineering and Management (IRJAEM)*, *2*(06), 2092–

2105. https://doi.org/10.47392/irjaem.2024.0307

Mochammad Anshori, Farhanna Mar'i, & Fitra A. Bachtiar. (2019). Comparison of Machine Learning Methods for Android Malicious Software Classification based on System Call. *International Conference on Sustainable Information Engineering and Technology (SIET)*.

Muhammad Helmi Satria Fedianto, Firza Prima Aditiawan, & Muhammad Muharrom Al Haromainy. (2023). Pengujian Sistem Jaringan Dokumentasi Dan Informasi Menggunakan Black Box Testing Dan White Box Testing. *Jurnal Publikasi Sistem Informasi Dan Manajemen Bisnis*, *3*(1), 213–221. https://doi.org/10.55606/jupsim.v3i1.2447

Priastiwi, L., & Adlin Sinaga, I. (2023). Aplikasi Klasifikasi Ulasan Pelanggan pada Cafe Ti Amo dengan metode Naïve Bayes Menggunakan Framework Django. *Triase, Imam Adlin Sinaga INNOVATIVE: Journal Of Social Science Research*, *3*, 13814–13826.

Ramadhan, J. A., Susilo, A., Irawan, Y., & Solehudin, A. (2023). Perancangan Aplikasi Pengelolaan Perangkat Jaringan Dengan Pemrograman Python Berbasis Web (Studi Kasus: SMKN 3 Kota Bekasi). *Jurnal Mahasiswa Teknik Informatika*, *7*(4), 2756.

Ramdany, S. W., Aulia Kaidar, S., Aguchino, B., Amelia, C., Putri, A., & Anggie, R. (2024). Penerapan UML Class Diagram dalam Perancangan Sistem Informasi Perpustakaan Berbasis Web. *Journal of Industrial and Engineering System*, *5*(1), 30–41.

Sabita, H., Herwanto, R., Syafitri3, Y., Dwi Prasetyo, B., & Komputer, F. I. (2022). Pengembangan Aplikasi Akreditasi Program Studi Berbasis Framework Django. *Jurnal Informatika*, *22*(01).

Siska Narulita, Ahmad Nugroho, & M. Zakki Abdillah. (2024). Diagram Unified Modelling Language (UML) untuk Perancangan Sistem Informasi Manajemen Penelitian dan Pengabdian Masyarakat (SIMLITABMAS). *Bridge : Jurnal Publikasi Sistem Informasi Dan Telekomunikasi*, *2*(3), 244–256. https://doi.org/10.62951/bridge.v2i3.174

Taufan, M. A., Rusdianto, D. S., & Ananta, M. T. (2022). Pengembangan Sistem Otomatisasi Use Case Diagram berdasarkan Skenario Sistem menggunakan Metode POS Tagger Stanford NLP. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *6*(8), 3733–3740. http://j-ptiik.ub.ac.id

Togu Novriansyah Turnip, Chatrine Febryanti Manurung, Yogi Septian Lubis, & Rachel Gultom. (2023). Klasifikasi Malware Android Aplikasi Menggunakan Random Forest Berdasarkan Fitur Statik. *Jurnal Teknik Informatika Dan Sistem Informasi*, *10*.

Turaina, R., Saputra, R., Metamedia, U., & Khatib Sulaiman Dalam No, J. (2024). Optimalisasi Deteksi Malware pada Platform Android dengan Pendekatan Ensemble Machine Learning. *Jurnal Nasional Komputasi Dan Teknologi Informasi (JNKTI)*, *7*(3).

Umar, R., Robiin, B., Erviana, V. Y., Taruna, M. I., & Kurniawan, A. (2025). Analysis of Software Requirement Specification and Use Case Diagram of Metaverse Museum Muhammadiyah. *Preservation, Digital Technology and Culture*, *54*(2), 125–133. https://doi.org/10.1515/pdtc-2024-0062

Wijoyo, A., Saputra, A. Y., Ristanti, S., Sya'ban, R., Amalia, M., & Febriansyah, R. (2024). Pembelajaran Machine Learning. *OKTAL : Jurnal Ilmu Komputer Dan Science*, *3*.

Zhang, X., Mathur, A., Zhao, L., Rahmat, S., Niyaz, Q., Javaid, A., & Yang, X. (2022, August 23). An Early Detection of Android Malware Using System Calls based Machine Learning Model. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3538969.3544413