PENGEMBANGAN APLIKASI PENDETEKSI *MALWARE* BERBASIS ANDROID MENGGUNAKAN PERINTAH *STRACE*

Rinanza Zulmy Alhamri[1], Kunti Eliyen[2], Toga Aldila Cinderatama[3], Agustono Heriadi[4]

[1,2,3,4]Program Studi Manajemen Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang
[1]rinanza.z.alhamri@polinema.ac.id ,[2]kunti.eliyen@polinema.ac.id, [3]toga.aldila@polinema.ac.id,
[4]agustono.heriadi@polinema.ac.id

.

Abstrak

Deteksi *malware* pada sistem operasi Android dapat dilakukan dengan mengidentifikasi aktivitas *system calls* aplikasi yang dicurigai sehingga berdasarkan data aktivitas *system calls* tersebut aplikasi dapat diklasifikasikan sebagai *malware* atau jinak. Banyak penelitian yang telah dilakukan untuk mengidentifikasi *malware* pada Android menggunakan *machine learning*, namun pembahasan mengenai pengembangan aplikasi belum mendetail sehingga proses deteksi *malware* sulit untuk diimplementasikan. Penelitian ini membahas mengenai pengembangan aplikasi pendeteksi *malware* berbasis Android yang dapat melakukan deteksi dengan menerapkan analisis dinamis berdasarkan *system calls*. Dalam aplikasi yang dikembangkan ini, eksekusi perintah *strace* diharapkan dapat melacak aktivitas *system calls* suatu aplikasi yang dicurigai sebagai *malware* berdasarkan *process identifier* (PID), sehingga dapat diidentifikasi apakah aplikasi tersebut termasuk *malware* atau bukan. Permasalahannya adalah bagaimana aplikasi deteksi *malware* yang dikembangkan dapat menerapkan perintah *strace*. Agar perintah *strace* bisa dijalankan pada aplikasi pendeteksi *malware* maka diterapkan *Wrap Shell Script* dimana *package* aplikasi mampu menjalankan kode *native shell* yang mengandung perintah *strace*. Metode pengembangan aplikasi yang digunakan berupa metode *Waterfall* meliputi studi literatur, analisis, perancangan, implementasi, dan pengujian. *Wrap Shell Script* dapat diterapkan dengan memanfaatkan *library* dan penyimpanan data menggunakan Firebase Real-time database. Perangkat *smartphone* yang digunakan harus dalam kondisi *rooting* agar aplikasi dapat mengeksekusi perintah *strace* karena perintah *strace* hanya dapat bekerja menggunakan akses *root*. Hasil dari penelitian ini adalah aplikasi dapat menjalankan lima fungsi meliputi dapat melakukan registrasi, otentikasi, melihat daftar PID aplikasi, pendeteksian malware, dan melihat hasil deteksi malware.

Kata kunci: ***malware, system calls, strace, wrap shell script, android***

1. Pendahuluan

Peningkatan penggunaan internet melalui perangkat *mobile* ini selaras dengan peningkatan kejahatan siber pada perangkat mobile terutama smartphone berbasis Android. Menjadi perhatian serius selama tahun 2023 untuk kejahatan siber pada perangkat *smartphone* Android yaitu serangan *malware*. Menurut Badan Siber dan Sandi Negara selama Agustus 2023, terjadi serangan siber ke Indonesia mencapai 219.414.104 serangan (Kristianti, 2023). Dari total serangan tersebut, dominasi serangan ada pada serangan *malware* di mana mencapai 115.208.766 serangan atau sebesar 52,51%. Hal ini diyakini karena adanya faktor perilaku pengguna *mobile* di mana banyak pengguna mengunduh aplikasi bajakan pada perangkat *smartphone*. *Malicious software* atau *malware* adalah perangkat lunak yang berbahaya di mana dikembangkan oleh penjahat untuk melemahkan perangkat keras *host*-nya. *Malware* memiliki beberapa jenis dan dampaknya masing-masing diantaranya *adware* yang membuat banyak *pop-up* iklan, *spyware* yang dapat memantau aktivitas pengguna perangkat *mobile*, *ransomeware* yang dapat mengunci data, dan trojan di mana menempel pada aplikasi resmi namun ketika diinstal bisa merusak sistem (Fadly, 2021)(Azahari et al., 2021). Oleh karena itu, kerangka kerja deteksi *malware* semakin berkembang sebagai solusi dalam melakukan deteksi *malware* pada *smartphone* Android.

Sebagai solusi atas maraknya serangan *malware* terhadap *smartphone* Android, telah banyak dilakukan penelitian seperti penelitian paling sederhana yaitu mengupload file apk mencurigakan ke pihak ketiga seperti VirusTotal, secara singkat akan dilakukan deteksi *signature* denga *reengineering* oleh VirusTotal dan akan diperoleh langsung hasilnya (Sinambela et al., 2020). Kemudian melakukan kombinasi antara analisis dinamis mengacu pada API *call signature* dengan analisis statis mengacu pada *intent filter, system command,* dan *fingerprint* dalam mendeteksi *malware* menggunakan metode-metode Machine Learning (ML) (Hadiprakoso et al., 2020) serta

metode-metode Deep Learning (DL) (Hadiprakoso et al., 2021). Menggunakan analisis statis saja, dibandingkan berbagai metode *supervised* Machine Learning (ML) dalam mendeteksi *malware* (Hadiprakoso et al., 2022), digunakan metode DL dengan algoritma Convolutional Neural Network (CNN) (Sharipuddin et al., 2024), serta optimasi CNN menggunakan *sequential* dan *feature selection* (Winarnie, 2024). (Chitayae & Muhammad, 2023) melakukan deteksi *malware* pada Android berdasarkan website sumber apk menggunakan klasifikasi K-Nearest Neighbor (KNN). Berdasarkan penelitian yang dilakukan, tema deteksi *malware* berbasis Android lebih berfokus pada kinerja akurasi deteksi.

Sedangkan pada penelitian lain lebih mengangkat topik kerangka kerja dalam mendeteksi *malware* Android dengan beragam studi kasus tertentu. Beberapa kerangka kerja deteksi malware semakin berkembang seperti pada penelitian (Khariwal et al., 2020) diperkenalkan IPDroid di mana analisis deteksi berdasarkan API *call* dan perizinan aplikasi, sedangkan dalam (Tarwireyi et al., 2022) diperkenalkan BarkDroid di mana deteksi malware berdasarkan Bark Frequency Cepstral Coefficients sebagai parameter fitur klasifikasi. Penelitian selanjutnya diperkenalkan DynaMalDroid di mana deteksi *malware* menggunakan ML berdasarkan analisis dinamis *system call* (Manzil & S, 2022) dan dengan menggunakan ML dianalisis statis menggunakan data *permission-based* (Rawat et al., 2022). Untuk penelitian terbaru memperkenalkan pendekatan pemanfaatan ML terbaru dengan penerapan Co-Existence of Features pada proses ekstraksi fitur (Odat & Yaseen, 2023).

Dari sekian penelitian yang dilakukan, tidak banyak studi yang membahas pengembangan aplikasi Android dalam mendeteksi *malware* sehingga dapat digunakan oleh pengguna. Sebelumnya telah dikembangkan aplikasi Android pendeteksi *malware* sederhana berbasis Java dimana aplikasi digunakan untuk mengubah file hasil pelacakan *system call* dari .txt menjadi .csv (Herlambang et al., 2018). Penelitian terakhir tentang aplikasi pendeteksi *malware* berbasis Android paling terkini adalah penelitian (Zhang et al., 2022) di mana mengembangkan aplikasi Android yang mampu melakukan pelacakan *system calls* untuk kemudian diunggah ke server untuk dilakukan klasifikasi deteksi *malware* secara *localhost*. Dari penelitian tersebut, belum ada pembahasan pengembangan aplikasi pendeteksi malware berbasis Android secara komprehensif dan siap digunakan pengguna sehingga memudahkan pengguna dalam mendeteksi aplikasi Android.

Untuk itu perlu adanya penelitian yang membahas khusus tentang pengembangan aplikasi pendeteksi *malware* berbasis Android. Aplikasi dikembangkan sebagai antarmuka bagi pengguna yang ingin melakukan deteksi *malware* terhadap aplikasi yang diinstal pada perangkatnya dimana dikembangkan berdasarkan kerangka kerja DynaMalDroid menggunakan analisis dinamis berupa *system call* dalam mendeteksi *malware* (Manzil & S, 2022). Kembali mereferensi kerangka kerja DynaMalDroid, sebelumnya telah dilakukan penelitian berupa penerapan *machine learning* untuk deteksi *malware* berdasarkan *system calls* yang mengacu pada tabel *system calls* ARM (32-bit/EABI) (Alhamri et al., 2024), sehingga aplikasi yang dikembangkan pada penelitian ini selaras dengan penelitian tersebut di mana penelitian ini mengembangkan aplikasi pendeteksi *malware* pada Android sebagai aplikasi antarmuka pengguna.

Dalam aplikasi yang dikembangkan ini, eksekusi perintah *strace* diharapkan dapat melacak aktivitas *system calls* suatu aplikasi yang dicurigai sebagai *malware* berdasarkan *process indentifier* (PID), sehingga dapat diidentifikasi apakah aplikasi tersebut termasuk *malware* atau bukan. Perintah *strace* adalah perintah *terminal* yang digunakan untuk memonitor interaksi antara proses aplikasi dengan *kernel* Linux dimana eksekusinya menggunakan *command prompt*. Permasalahan pada penelitian ini adalah bagaimana aplikasi deteksi *malware* yang dikembangkan dapat menerapkan perintah *strace* tersebut. Agar perintah *strace* bisa dijalankan pada aplikasi pendeteksi *malware* maka diterapkan *Wrap Shell Script* dimana *package* aplikasi mampu menjalankan kode *native shell* yang mengandung perintah *strace*. Tujuan penelitian ini adalah agar aplikasi pendeteksi *malware* pada *smartphone* Android memungkinkan untuk mampu mengeksekusi perintah *strace* melalui bahasa *native* seperti *command prompt* menggunakan Native Development Kit (NDK) Android pada konsep *Wrap Shell Script*. Dengan aplikasi dapat menjalankan *shell script*, maka perintah *strace* untuk melacak *system calls* bisa diterapkan.

Kebaruan penelitian ini adalah mengembangkan aplikasi pendeteksi *malware* menggunakan bahasa Kotlin dan *Wrap Shell Script* dimana mampu untuk melacak *system calls* aplikasi mencurigakan menggunakan perintah *strace*. Keluaran perintah *strace* berupa .txt di mana untuk kemudian dikirim menuju server *cloud* secara online via internet. File hasil pelacakan berformat .txt tersebut digunakan untuk analisis deteksi *malware* pada sisi server *machine learning*. Penelitian ini merupakan bagian dari serangkaian penelitian tentang sistem pendeteksi *malware* pada *smartphone* berbasis Android, dimana penelitian sebelumnya telah membandingkan beberapa metode ML untuk mendeteksi *malware* berdasarkan analisis dinamis berupa pelacakan *system call* mengacu pada tabel ARM (32-bit/EABI) (Alhamri et al., 2024).

2. Metode

Penelitian ini memiliki lima tahapan utama meliputi Studi Literatur, Analisis, Perancangan, Implementasi, dan Pengujian sesuai dengan kaidah

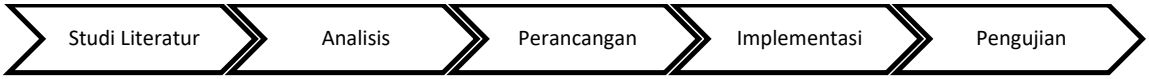
pengembangan perangkat lunak dengan metode *Waterfall* seperti yang dijelaskan pada Gambar 1.

2.1. Studi Literatur

Mengumpulkan informasi penerapan *Wrap Shell Script* untuk pengembangan aplikasi pendeteksi *malware* berbasis Android meliputi data primer yaitu melakukan pengambilan data studi literatur pada artikel, jurnal, serta material online tentang penggunaan *Wrap Shell Script* dalam menjalankan perintah *shell strace* pada aplikasi pendeteksi *malware* berbasis Android. Kemudian data skunder dengan melakukan pengambilan data studi literatur pada artikel, jurnal, serta material online mengenai kerangka kerja dan konsep deteksi *malware* berbasis Android berdasarkan *system calls*.

a. *Malware*

Malware (*malicious software*) adalah program komputer yang diciptakan dengan maksud dan tujuan tertentu (Azahari et al., 2021) dari penciptanya dan merupakan program yang mencari kelemahan dari sebuah *software*. Umumnya *malware* diciptakan untuk membobol atau merusak suatu *software* atau sistem operasi melalui *script* yang disisipkan secara tersembunyi oleh pembuatnya. *Malware* juga mampu merusak sistem seperti pada perangkat-perangkat *mobile* berbasis Android.



Gambar 1. Tahapan Penelitian

b. Deteksi *Malware* dengan Analisis Dinamis

Melakukan deteksi *malware* dengan pendekatan analisis dinamis erat kaitannya dengan melakukan pelacakan terhadap aktivitas aplikasi yang dicurigai seperti penggunaan *registry*, aktivitas jaringan, aliran data yang digunakan, atau pelacakan *system call* secara *real-time* (Azahari et al., 2021). Sedangkan analisis statis lebih ke *reverse engineering* pada isi program aplikasi yang dicurigai dan analisis hibrid menggabungkan analisis dinamis dengan analisis statis. Hasil dari pelacakan akan dianalisis oleh server melalui berbagai macam metode klasifikasi seperti ML atau DL. Seperti pada kerangka deteksi *malware* DynaMalDroid untuk mendeteksi *malware* Android berdasarkan *system calls* (Manzil & S, 2022) maka digunakan perintah *shell* berupa *strace* dimana output perintah secara umum disimpan ke dalam file .txt. File .txt menjadi sampel untuk dianalisis dalam server klasifikasi.

c. Perintah *Strace*

Strace adalah perangkat bagi pengguna untuk melakukan diagnostik, *debugging*, dan instruksional pada sistem operasi berbasis Linux. Secara umum deteksi *malware* berdasarkan *system call* memanfaatkan perintah *strace*, sehingga aktivitas penggunaan *system call* oleh aplikasi yang dicurigai bisa dilacak (Manzil & S, 2022)(Zhang et al., 2022). Penelitian sebelumnya pelacakan dilakukan dengan menjalankan sistem operasi Android secara simulasi untuk kemudian melalui komputer *host* dijalankan terminal untuk mengeksekusi perintah strace dalam durasi tertentu sehingga diperoleh file .txt berisi aktivitas *system calls*.

Aplikasi yang dilacak *system calls*-nya harus memiliki aktivitas apapun itu selama durasi pelacakan, apabila tidak ada aktivitas sama sekali maka output file .txt akan kosong isi aktivitasnya. Biasanya dalam penelitian sebelumnya digunakan aplikasi yang menghasilkan aktivitas acak seperti klik, geser, sentuh pada aplikasi mencurigakan yang dilacak secara otomatis (Manzil & S, 2022)(Zhang et al., 2022). Contoh hasil pelacakan *system call* pada aplikasi Android dengan output file format .txt ditampilkan pada Gambar 2.

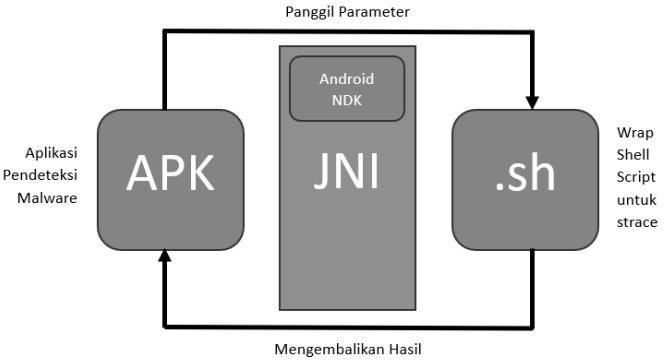
```
6632  restart_syscall(<... resuming interrupted futex ...> <unfinished ...>
6631  futex(0xee43783c, FUTEX_WAIT_PRIVATE, 2, NULL <unfinished ...>
6628  clock_gettime(CLOCK_MONOTONIC,  <unfinished ...>
6611  ioctl(8, BINDER_WRITE_READ <unfinished ...>
6628  <... clock_gettime resumed> {tv_sec=2405, tv_nsec=507145982}) = 0
6628  epoll_pwait(108,  <unfinished ...>
6607  restart_syscall(<... resuming interrupted futex ...> <unfinished ...>
6605  futex(0xee435c3c, FUTEX_WAIT_PRIVATE, 2, NULL <unfinished ...>
6603  restart_syscall(<... resuming interrupted futex ...> <unfinished ...>
6604  clock_gettime(CLOCK_MONOTONIC,  <unfinished ...>
```

Gambar 2. Output Perintah *Strace*

d. *Wrap Shell Script*

Wrap Shell Script digunakan untuk developer jika ingin melakukan *debugging* dan *profiling* terhadap suatu aplikasi menggunakan kode *native* (Android Developer, 2022). Berguna sebagai alat *debugging* di mana parameter yang ada pada Wrap Shell Script akan dieksekusi pertama kali ketika aplikasi dimulai. Contoh aplikasi *debug* memanfaatkan *Wrap Shell Script* adalah melacak *system calls* suatu aplikasi Android dengan *strace*, menemukan *bug* memori dengan Malloc Debug atau Address Sanitizer (ASan), serta *profiling* aplikasi dengan menggunakan Simpleperf.

Saat menggunakan apk yang dapat di-*debug* yang berisi wrap.sh, sistem akan mengeksekusi skrip dan meneruskan perintah untuk memulai aplikasi sebagai argumen. Skrip bertanggung jawab untuk memulai aplikasi, tetapi dapat membuat perubahan lingkungan atau argumen apapun. Secara singkat Wrap Shell Script mengandung kode *native* sintaks *shell* di mana dapat dieksekusi dengan bantuan Java Native Interface (JNI) yang telah tersedia *library*-nya dengan memanfaatkan Android NDK seperti Gambar 3. Pada penelitian ini digunakan *library* KtSh (Rummler, 2021) dimana menerapkan *konsep Wrap Shell Script* pada proyek aplikasi Android sehingga aplikasi dapat menerapkan perintah *shell.*



Gambar 3. Penerapan *Wrap Shell Script*

2.2. Analisis

Terdapat server di mana sebagai penganalisis memanfaatkan *supervised* ML dimana pada serangkaian penelitian yang dilakukan sebelumnya telah membandingkan berbagai *supervised learning* (Alhamri et al., 2024). ML telah dilatih dan dapat mengklasifikasikan data pelacakan *system call* menggunakan file format .txt sebagai inputan. Sehingga dalam sistem deteksi *malware* pada Android, di sisi klien terdapat aplikasi pendeteksi *malware* berbasis Android yang mampu melacak aktivitas *system call* aplikasi mencurigakan. Aplikasi memanfaatkan *Wrap Shell Script* menggunakan *library* KtSh agar aplikasi dapat mengeksekusi perintah *shell strace.*

Ketika pengguna mencurigai suatu aplikasi Android sebagai *malware* maka bisa diambil *Process Identifier* (PID) dari aplikasi tersebut. Untuk mengetahui PID maka digunakan perintah *ps*. Setelah diketahui PID dari aplikasi, kemudian nomor PID digunakan sebagai variabel untuk dilakukan *strace*. Sebelum hasil output *strace* diperoleh, pengguna harus melakukan aktivitas pada aplikasi yang dicurgai tersebut secara acak dalam durasi waktu tertentu. Setelah output *strace* berupa file .txt diperoleh, selanjutnya file .txt tersebut dikirim menuju server analisis *malware* untuk diklasifikasikan.

Pada studi kali ini, akan dimanfaatkan penyimpanan *cloud* sebagai penerapan deteksi *malware* secara online via internet. Aplikasi pendeteksi *malware* akan mengirim file strace .txt ke *cloud* Firebase. Untuk menyimpan file .txt maka digunakan Firebase Storage, sedangkan untuk menyimpan hasil deteksi digunakan Firebase Real-time Database. Penyimpanan *cloud* deteksi *malware* pada Firebase Real-time Database tersebut akan menunggu respon dari ML berupa hasil analisis bernilai String meliputi “malware” atau “jinak”. Penyimapanan *cloud* digunakan sebagai perantara antara aplikasi pendeteksi *malware* Android sebagai klien dengan ML untuk analisis sebagai server.

Sebagai pemecahan masalah maka digunakan *Wrap Shell Script* dan didukung dengan Android NDK sehingga aplikasi yang dikembangkan bisa mengeksekusi kode *native shell* berupa perintah strace dan pendukungnya yaitu ps secara langsung di *smartphone* Android. Adapun proses bisnis pemecahan masalah ditampilkan pada Gambar 4 di mana pengguna bisa registrasi, kemudian bisa otentikasi ke aplikasi deteksi *malware*.

Gambar 4. Proses Bisnis Aplikasi Pendeteksi *Malware* Berbasis Android

Pengguna bisa melihat PID aplikasi yang dicurigai di *smartphone*. Jika pengguna mencurigai suatu aplikasi adalah *malware*, pengguna bisa mendeteksi aplikasi tersebut dengan memasukkan PID, dimana secara fungsional aplikasi akan melacak *system calls* berdasarkan PID tersebut dan mengirimkannya ke *cloud* Firebase. Pengguna bisa melihat hasil deteksi berdasarkan respon dari server ML melalui penyimpanan *cloud*. Pada penelitian ini fokus terhadap peningkatan *value* aplikasi pendeteksi *malware* berbasis Android dengan mampu mengeksekusi perintah *shell strace* menggunakan *Wrap Shell Script*.

2.3. Perancangan

Perancangan aplikasi akan dijelaskan menggunakan *use case,* arsitektur sistem, dan skema database.

a. *Use Case*

Use case terdiri dari lima *case* meliputi registrasi, otentikasi, melihat daftar PID, mendeteksi malware, dan melihat hasil deteksi seperti pada Gambar 5.

b. Arsitektur Sistem

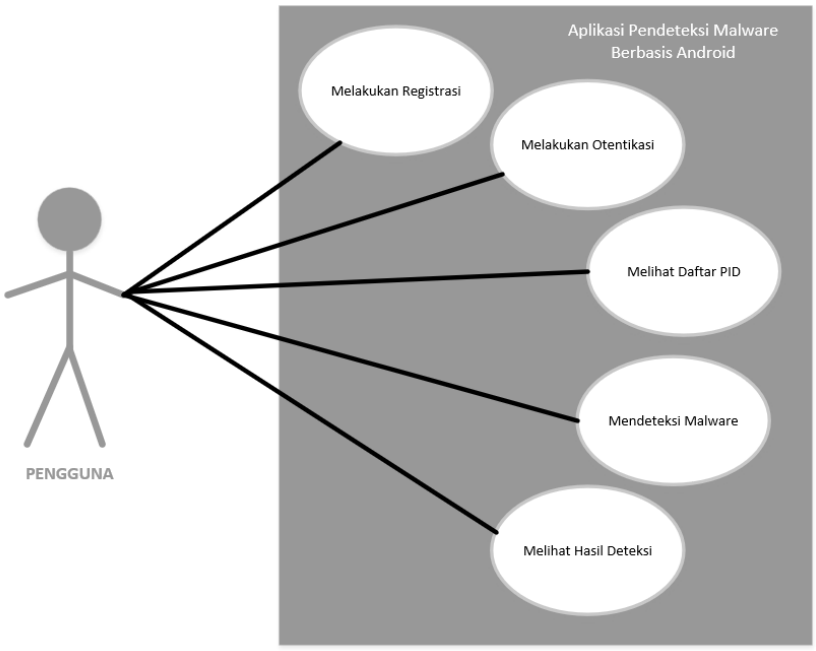
Pengguna aplikasi hanya satu yaitu pengguna yang secara langsung mengakses aplikasi pendeteksi *malware* berbasis Android *menggunakan Wrap Shell Script* dengan rsitektur sistem ditampilkan pada Gambar 6. Data dari aplikasi pendeteksi *malware* dikirim menuju penyimpanan *cloud* Firebase via intenrnet. Sedangkan data hasil klasifikasi sebagai respon server, akan diambil hasilnya oleh aplikasi pendeteksi *malware*.

c. Skema Database

Data disimpan pada Firebase terutama pada layanan Real-time Database dan Storage. Real-time database digunakan untuk menyimpan data hasil deteksi sedangkan Storage digunakan untuk menyimpan data hasil strace sebagai bahan analisis klasifikasi deteksi. Penyimpanan data untuk hasil deteksi disimpan di dalam *reference* DeteksiMalware untuk kemudian terdapat *child* PID yang terdiri *sub-child* file, nama, dan status untuk menyimpan data deteksi aplikasi yang dicurigai. *Sub-child* file digunakan untuk link file .txt *strace* ke Storage, nama digunakan untuk nama aplikasi yang dicurigai, sedangkan status digunakan untuk menampilkan hasil deteksi apakah jinak atau *malware* seperti pada Tabel 1.

Tabel 1. Skema Database

Reference	Child	Sub-Child	Fungsi
DeteksiMalware			Sebagai referensi ketika data disimpan dari aplikasi Android
	PID		Sebagai referensi aplikasi yang dicurigai
		File	Field untuk menyimpan link teks strace pada Storage
		Nama	Field untuk menyimpan nama aplikasi yang dicurigai
		Status	Field untuk menyimpan status deteksi aplikasi dicurigai



Gambar 5. Diagram Use *Case* Aplikasi Pendeteksi *Malware* Berbasis Android

Gambar 6. Arsitektur Sistem pada Aplikasi Pendeteksi *Malware* Berbasis Android

2.4. Implementasi

Implementasi aplikasi pendeteksi malware berbasis Android dilakukan dengan tiga sub tahapan meliputi Pengembangan Aplikasi Android, Penerapan *Wrap Shell Script*, dan Konfigurasi Firebase seperti pada Gambar 7.

a. Pengembangan Aplikasi Android

Menggunakan bahasa pemrograman Kotlin dengan IDE Android Studio untuk menyiapkan halaman registrasi, otentikasi, melihat PID, mendeteksi *malware*, dan melihat hasil deteksi.

b. Penerapan Wrap Shell Script

Menyiapkan perintah *shell* berupa strace untuk melacak *system call* dan ps untuk melihat PID pada aplikasi. Penerapan Wrap Shell Script dilakukan dengan menggunakan *library* KtSh yang memungkinkan eksekusi perintah *shell*.

c. Konfigurasi Firebase

Menyediakan proyek untuk kebutuhan aplikasi deteksi *malware* Android menggunakan layanan *Authentication*, Real-time Database, serta Storage. Kemudian menggunakan *library* Firebase pada aplikasi untuk registrasi, otentikasi, serta *upload-download* data.

2.5. Pengujian

Pengujian dilakukan dengan pengujian fungsional dimana melakukan pengujian aplikasi Android berdasarkan fungsi-fungsi yang telah direncanakan sebelumnya dengan pengujian Black-Box seperti pada penelitian (Masrudini et al., 2024).

Gambar 7. Tahap Implementasi

3. Hasil dan Pembahasan

3.1. Implementasi Data

Tampilan implementasi skema database penyimpanan hasil deteksi pada Firebase Real-time Database ditampilkan pada Gambar 8. Sedangkan penyimpanan data untuk file teks *system call* hasil strace pada aplikasi yang dicurigai disimpan di dalam Storage secara langsung dengan penamaan PID sesuai data yang dikirim dari aplikasi Android seperti yang ditampilkan pada Gambar 9.

3.2. Implementasi Antarmuka

Hasil implementasi antarmuka sesuai perancangan aplikasi. Terdapat tujuh fungsi meliputi fungsi melakukan registrasi, melakukan otentikasi, melihat PID, mendeteksi *malware*, dan melihat hasil deteksi *malware*.

Gambar 8. Hasil Implementasi Data Firebase Real-time Database

Gambar 9. Hasil Implementasi Data Firebase Storage

a. Melakukan Registrasi

Fungsi melakukan registrasi untuk mendata akun pengguna aplikasi pendeteksi *malware* berbasis Android di mana digunakan email Google memanfaatkan Firebase Authentication. Dengan pengguna mendaftar maka pengguna bisa menggunakan aplikasi pendeteksi *malware*. Registrasi disimpan pada Firebase Authentication dengan mendaftarkan email Goolge dan kemudian pengguna tinggal melakukan verifikasi melalui email tersebut. Gambar 10 merupakan tampilan registrasi pengguna.

b. Melakukan Otentikasi

Fungsi otentikasi adalah untuk mengelola pengguna sehingga aplikasi menyiapkan data hasil deteksi *malware* sesuai dengan data masing-masing pengguna. Pengguna melakukan otentikasi pada aplikasi pendeteksi *malware* berbasis Android dengan memasukkan *username* berupa alamat email Google dengan *password* sesuai dengan apa yang didaftarkan sebelumnya. Gambar 11 merupakan tampilan halaman otentikasi pengguna berupa *login*.

Gambar 10. Hasil Proses Registrasi

Gambar 11. Hasil Proses Otentikasi

c. Melihat PID

Fungsi melihat daftar PID adalah untuk mengetahui nomor PID proses aplikasi yang dicurigai pada menu PID. Dengan pengguna dapat melihat nomor PID, maka pengguna dapat melakukan perintah *strace* untuk mencatat aktivitas *system calls*. Gambar 12 merupakan tampilan melihat daftar PID dengan cara memasukkan nama aplikasi dan nomor PID nama aplikasi tersebut akan keluar.

d. Mendeteksi *Malware*

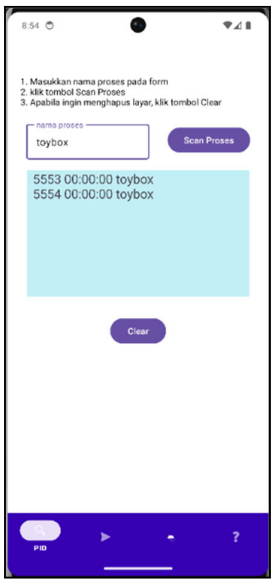
Fungsi mendeteksi *malware* diawali dengan melakukan pelacakan *system calsl* pada menu Strace. Menu Strace adalah untuk melacak aktivitas *system call* dari aplikasi yang dicurigai berdasarkan nomor PID. Pelacakan aktivitas *system calls* akan memperoleh keluaran berupa file .txt dengan nama sesuai nomor PID, di mana file .txt hasil strace tersebut digunakan untuk menentukan apakah aplikasi yang dicurigai adalah *malware* atau jinak. Gambar 13 merupakan tampilan melakukan *strace* dengan cara memasukkan nomor PID kemudian melakukan aktivitas acak pada aplikasi yang dicurigai.

Aktivitas acak pada aplikasi yang dicurigai dilakukan dalam waktu 3 menit. Saat pengguna berpindah ke aplikasi mencurigakan untuk melakukan aktivitas acak selama 3 menit, aplikasi deteksi *malware* dimana menjalankan perintah strace akan tetap berjalan karena perintah strace berjalan di

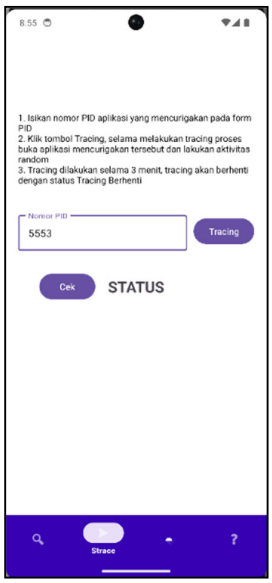
sisi *brackground*. Tampilan Status pada halaman menu Strace akan bernilai "Running" apabila masih melakukan pelacakan. Setelah hasil strace file .txt diperoleh, kemudian pengguna meng-*upload* data file strace tersebut untuk dideteksi pada menu Detect. Pengguna memilih file .txt sesuai nomor PID aplikasi yang dicurigai pada direktori Download, kemudian pengguna mengisikan data nama aplikasi yang dicurigai tersebut. Data strace dikirim menuju ke penyimpanan *cloud* Firebase Real-time Database dan Storage untuk memperoleh respon dari server. Tampilan mendeteksi *malware* ditampilkan pada Gambar 14.

e. Melihat Hasil Deteksi

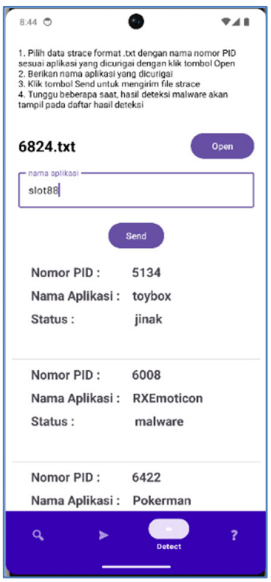
Fungsi melihat hasil deteksi digunakan untuk melihat daftar data strace yang sudah keluar hasil deteksinya. Fungsi ini satu tampilan dengan fungsi melakukan deteksi *malware* pada menu Detect. Ditampilkan pada tabel secara mendaftar meliputi nomor PID, nama aplikasi, serta hasil deteksi. Hasil deteksi *malware* diperlihatkan pada Gambar 15.



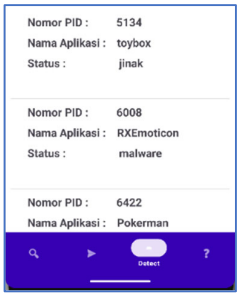
Gambar 12. Hasil Melihat Daftar PID



Gambar 13. Tampilan Melakukan Strace



Gambar 14. Tampilan Mendeteksi Malware



Gambar 15. Hasil Deteksi Malware

3.3. Pengujian

Pengujian dilakukan dengan menguji aplikasi sesuai kebutuhan fungsional yang direncanakan menggunakan Black-Box. Pengujian Fungsionalitas bermaksud untuk menguji aplikasi sesuai kebutuhan fungsional dari aplikasi pendeteksi *malware* berbasis Android. Tabel 2 merupakan hasil pengujian fungsional pada aplikasi pendeteksi *malware* berbasis Android. Secara keseluruhan, fungsi yang direncanakan berhasil dikembangkan dari kelima *case* yaitu melakukan registrasi, melakukan otentikasi, melihat nomor PID, melakukan deteksi *malware,* dan melihat hasil deteksi.

3.4. Pembahasan

Saat melakukan pengujian ditemukan permasalahan pada sisi aplikasi pendeteksi *malware* berbasis Android, di mana *library* KtSh tidak bisa mengeksekusi perintah-perintah *shell strace*. Hal ini dikarenakan perintah strace memerlukan akses *root* atau *superuser,* sehingga aplikasi meskipun bisa di-*instal* pada *smartphone* Android normal namun fungsi deteksi *malware* khususnya perintah *strace* tidak bisa dijalankan. Untuk itu agar aplikasi bisa berjalan normal, maka aplikasi dipasang pada *smartphone* Android yang sudah dilakukan *rooting*. Melakukan *rooting* pada *smartphone* Android merupakan langkah resiko karena selain cara *rooting* yang kompleks juga kondisi smartphone Android *root* menjadikan adanya celah keamanan. Untuk itu aplikasi ini masih sesuai dengan perangkat *smartphone* Android kondisi *rooting*, dimana celah keamanan memang benar-benar beresiko.

Agar hasil *strace* memiliki aktivitas *system calls*, pengguna harus benar-benar melakukan aktivitas secara sengaja maupun acak selama 3 menit pada aplikasi yang dicurigai. Dengan durasi aktivitas 3 menit maka akan didapatkan minimal 1000 daftar *system calls* dari aplikasi yang dicurigai. Namun demikian dibandingkan dengan aplikasi yang dikembangkan pada penelitian (Herlambang et al., 2018), aplikasi ini sudah mampu melakukan pelacakan *system calls* secara langsung pada perangkat *smartphone* Android. Lebih lanjut aplikasi yang dikembangkan ini, dapat berjalan melalui internet memanfaatkan *cloud* Firebase dibangingkan dengan aplikasi yang berjalan secara *localhost* seperti pada penelitian (Zhang et al., 2022).

Dengan dikembangkannya aplikasi deteksi *malware* Android yang mampu mengeksekusi perintah strace ini, maka aplikasi ini dapat diterapkan pada kerangka kerja DynaMalDroid (Manzil & S,

2022) atau penelitian (Zhang et al., 2022) di mana selanjutnya perlu dikembangkan server ML untuk analisis *malware* berdasarkan *system call* secara nyata.

Tabel 2. Hasil Pengujian Fungsional Menggunakan Black-Box

No	Kasus Uji	Input	Output Harapan	Hasil
1	Melakukan registrasi	- Email Google - Password	Tampil halaman registrasi, memperoleh email verifikasi dari Google	Berhasil
2	Melakukan otentikasi	- Email Google - Password	Tampil menu PID, dapat login dan terbuka semua menu navigasi, dapat logout	Berhasil
3	Melihat PID aplikasi Android yang running	- Nama aplikasi mencurigakan	Tampil menu PID dan tampil nomor PID sesuai nama aplikasi yang dicurigai	Berhasil
4	Mendeteksi malware berdasarkan PID	- Nomor PID - File hasil strace .txt - Nama aplikasi yang dicurigai	- Tampil menu Strace, dapat melakukan tracing *system call* berdasarkan inputan nomor PID secara *background*, tampil status *running* saat proses pelacakan masih berjalan dan memperoleh file strace .txt di direktori Download - Tampil menu Detect, bisa memilih file .txt hasil strace dengan nama nomor PID, bisa upload file .txt ke *cloud* dengan nama sesuai inputan pengguna	Berhasil
5	Melihat hasil deteksi malware	-	Tampil menu Detect berupa hasil deteksi secara mendaftar meliputi nomor PID, nama aplikasi, dan status	Berhasil

4. Kesimpulan

Telah berhasil dikembangkan aplikasi pendeteksi *malware* berbasis Android memanfaatkan perintah strace dengan satu pengguna dan fungsi meliputi bisa melakukan registrasi, bisa melakukan otentikasi, bisa melihat daftar PID aplikasi, bisa melakukan deteksi *malware,* dan bisa melihat hasil deteksi malware. Perintah *shell* strace bisa dieksekusi menggunakan *Wrap Shell Script* dengan *library* KtSh serta perintah-perintah *shell* lain seperti ps. Terdapat temuan bahwa aplikasi berjalan normal pada *smartphone* Android yang sudah dilakukan *rooting* karena perintah strace membutuhkan akses *root*.

Dapat dilakukan penelitian berikutnya berupa pengembangan aplikasi agen sebagai server ML secara online melalui internet dimana menjalankan *supervised* ML untuk menganalisis file .txt hasil strace. Server ML tersebut dapat mengambil data dari penyimpanan *cloud* sebagai perantara komunikasi antara server ML dengan klien aplikasi pendeteksi *malware* pada *smartphone* Android. Sehingga sistem deteksi *malware* berbasis Andorid dengan analisis dinamis yaitu aktivitas *system calls* dapat diterapkan secara nyata.

Ucapan Terimakasih

Ucapan terimakasih sebesar-besarnya diberikan kepada UPA P3M Politeknik Negeri Malang yang telah mendukung baik secara moril maupun meteriil terlaksananya penelitian yang dilakukan ini.

Daftar Pustaka:

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.

Android Developer. (2022, January 31). *Wrap shell script*. Https:// Developer.Android.Com/Ndk/Guides/Wrap-Script.

Azahari, A. M., Ahmad, A., Rahayu, S. B., Kamarudin, N. D., & Halip, M. H. M. (2021). Android Designed Malware Detection Challenges: A Future Research Direction. *ZULFAQAR*, *4*(1), 38–44.

Chitayae, N., & Muhammad, A. H. (2023). Identifikasi Malware pada Android Menggunakan Algoritma K-Nearest Neighbor . *JIFOTECH*, *3*(2), 63–68.

Fadly, M. (2021). *Tanda HP Android Kena Malware atau Virus dari Aplikasi Berbahaya*. Https://Tirto.Id/Tanda-Hp-Androidkena-Malware-Atau-Virus-Dari-Aplikasi-Berbahaya-EjV2.

Hadiprakoso, R. B., Adistya, W. R., & Pramitha, F. N. (2022). Analisis Statis Deteksi Malware Android Menggunakan Algoritma Supervised Machine Learning. *CyberSecurity Dan Forensik Digital* , *5*(1), 1–5.

Hadiprakoso, R. B., Kabetta, H., & Buana, K. S. (2020). Hybrid-Based Malware Analysis for Effective and Efficiency Android Malware Detection. *International Conference on Informatics, Multimedia, Cyber and Information System*.

Hadiprakoso, R. B., Qomariasih, N., & Yasa, R. N. (2021). Identifikasi Malware Android Menggunakan Pendekatan Analisis Hibrid dengan Deep Learning. *JTIULM*, *6*(2), 77–84.

Herlambang, S., Basuki, S., Akbi, D. R., & Sari, Z. (2018). Deteksi Malware Android Berdasarkan System Call Menggunakan Algoritma Support Vector Machine. *Seminar Nasional Teknologi Dan Rekayasa*, 157–165.

Khariwal, K., Singh, J., & Arora, A. (2020). IPDroid: Android malware detection using intents and permissions. *Proceedings of the World Conference on Smart Trends in Systems,*

Security and Sustainability, WS4 2020, 197–202. https://doi.org/10.1109/WorldS450073.2020.9210414

Kristianti, L. (2023, August 7). *BSSN sebut serangan siber malware marak akibat "software" bajakan*. Https://Www.Antaranews.Com/Berita/3670320/Bssn-Sebut-Serangan-Siber-Malware-Marak-Akibat-Software-Bajakan.

Manzil, H. H. R., & S, M. N. (2022). DynaMalDroid: Dynamic Analysis-Based Detection Framework for Android Malware Using Machine Learning Techniques. *IEEE International Conference on Knowledge Engineering and Communication Systems, ICKES 2022*, 1–6. https://doi.org/10.1109/ICKECS56523.2022.10060106

Masrudini, M., Sukamto, A. S., & Pratama, E. E. (2024). Rancang Bangun Aplikasi Manajemen Penyewaan Fasilitas Olahraga di Kota Pontianak. *JEPIN*, *10*(1), 95–103.

Odat, E., & Yaseen, Q. M. (2023). A Novel Machine Learning Approach for Android Malware Detection Based on the Co-Existence of Features. *IEEE Access*, *11*, 15471–15484. https://doi.org/10.1109/ACCESS.2023.3244656

Rawat, N., Amrita, & Singh Avjeet. (2022). Permission-Based Malware Detection in Android Using Machine Learning. *2022 International Conference on Knowledge Engineering and Communication Systems*, *22*(3), 1505–1517. https://doi.org/10.37896/YMER22.03/C4tabel

Rummler, J. (2021). *KtSh*. Https://Github.Com/Jaredrummler/KtSh.

Sharipuddin, Putra, R. S., Aulia, M. F., Maulana, S. A., & Jusia, P. A. (2024). Android Security: Malware Detection with Convolutional Neural Network and Feature Analysis. *Media Journal of General Computer Science*, *1*(1), 7–13.

Sinambela, S., Pangestu, A. R., & Feriyanto, R. (2020). Analisis Aplikasi Malware pada Android dengan Metode Statik. *ILKOMINFO*, *3*(2), 88–94.

Tarwireyi, P., Terzoli, A., & Adigun, M. O. (2022). BarkDroid: Android Malware Detection Using Bark Frequency Cepstral Coefficients. *Indonesian Journal of Information Systems (IJIS)*, *5*(1).

Winarnie. (2024). Pemanfaatan Sequential Convolutional Neural Networks pada Deteksi Malware Android. *Jurnal Ekonomi Dan Teknik Informatika*, *12*(1), 65–70.

Zhang, X., Mathur, A., Zhao, L., Rahmat, S., Niyaz, Q., Javaid, A., & Yang, X. (2022). An Early Detection of Android Malware Using System Calls based Machine Learning Model. *ARES '22: Proceedings of the 17th International Conference on Availability, Reliability and Security*, 1–9.