Analisis Deteksi *Malware* Android menggunakan metode *Support Vector Machine & Random Forest*

Yitshak Wanli Sitorus [1],Parman Sukarno [2], Satria Mandala[3],
[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]yitshakwanli@students.telkomuniversity.ac.id, [2]psukarno@telkomuniversity.ac.id,
[3]satriamandala@telkomuniversity.ac.id,

Abstrak

Menurut data laporan kementerian komunikasi dan informatika, tahun 2018 pengguna aktif smartphone di Indonesia lebih dari 100 juta orang serta pada tahun yang sama data dari *statcounter* pengguna android di Indonesia sebanyak 90,85%. Tingginya penggunaan android membuat sistem operasi android menjadi target serangan malware. Malware merupakan sebuah sistem yang diprogram agar dapat menyusup ke sebuah sistem operasi, sebuah sistem operasi yang telah diserang malware dapat mengalami kerusakan dan bahkan dengan niat yang lebih jahat malware dapat digunakan untuk mencuri data-data penting. Rentannya serangan malware dan dapat merugikan para pengguna android sehingga diperlukan analisis lebih lanjut, oleh karena permasalah yang ada mendorong penelitian ini dilakukan dengan tujuan untuk deteksi dini. Pada kasus ini digunakan pendekatan *machine learning* untuk melakukan klasifikasi data serangan malware android. *Machine learning* yang digunakan dalam penelitian ini adalah *Support Vector Machine* (SVM) dan *Random Forest*. Kedua metode *machine learning* itu dipilih karena pada penelitian-penelitian sebelumnya terbukti kedua metode itu sangatlah efektif melakukan klasifikasi dengan menghasilkan akurasi yang tinggi. Pada makalah ini dilakukan perbandingan antara metode *Support Vector Machine* (SVM) dengan metode *Random Forest* dalam melakukan klasifikasi data, serta membandingkan hasil akurasi dengan penelitian sebelumnya. Proses klasifikasi menggunakan metode *Support Vector Machine* (SVM) menghasilkan nilai precision 97%, nilai recall 97%, dan nilai f1-score 97%, dan akurasi 96,23%, pada metode Random Forest menghasilkan nilai precision 99%, nilai recall 99%,nilai f1-score 99%, dan akurasi 98,99%. Menurut hasil percobaan, metode *Random Forest* lebih unggul dari metode *Support Vector Machine* (SVM) dan pendekatan yang diusulkan di penelitian ini memiliki hasil performansi matrik di atas 95% yang lebih baik daripada penelitian sebelumnya.

Kata kunci : ***Android, Malware, Machine learning, Support Vector Machine* (SVM), *Random Forest.***

Abstract

According to data from the ministry of communication and informatics, in 2018 active smartphone users in Indonesia more than 100 million people and in the same year data from the android user statcounter in Indonesia as much as 90.85%. The high use of android makes the android operating system a target for malware attacks. Malware is a system that is programmed to infiltrate an operating system, an operating system that has been attacked by malware can be damaged and even with more malicious intentions malware can be used to steal important data. The vulnerability of malware attacks and can harm android users so further analysis is needed, because the existing problems encourage this research to be done with the aim of early detection. In this case, a machine learning approach is used to classify android malware attack data. The machine learning used in this study is support vector machine (SVM) and random forest. Both machine learning methods were chosen because in previous studies it was proven that both methods are very effective at classifying with high accuracy. In this paper, a comparison between the Support Vector Machine (SVM) method and the Random Forest method in classifying data, as well as comparing the results of accuracy with previous research. The classification process using the Support Vector Machine (SVM) method produces a precision value of 97%, a recall value of 97%, and an f1-score value of 97%, and an accuracy of 96.23%, in the Random Forest method it produces a precision value of 99%, a recall value of 99%, an f1-score value of 99%, and an accuracy of 98.99%. According to the results of the experiment, the Random Forest method is superior to the Support Vector Machine (SVM) method and the approach proposed in this study has a higher matric performance result above 95% than previous studies.

Keywords : Android, Malware, Machine learning, Support Vector Machine (SVM), Random Forest.

1. Pendahuluan

1.1. Latar Belakang

Pada era industri 4.0 teknologi sangatlah berkembang dengan pesat, khususnya teknologi informasi. Teknologi ini digunakan untuk memudahkan pekerjaan manusia dan keandalan suatu aplikasi sangatlah dibutuhkan oleh manusia. Salah satu teknologi informasi yang sering digunakan oleh manusia adalah android. Pada tahun 2017, *Gertner* melaporkan penjualan Smartphone di seluruh Dunia naik 9 persen pada kuartal pertama dengan 380 juta perangkat seluler yang terjual dan perangkat seluler yang menjadi primadona di pasar adalah android dengan 84,1% [1]. Laporan Kementerian komunikasi dan informatika memperkirakan jumlah pengguna aktif smarfphone di indonesia pada 2018 melebihin 100 juta orang [2], beserta *statcounter* melaporkan tahun 2018 jumlah pengguna android di indonesia 91% [3], sehingga dapat perkirakan jumlah pengguna android di indonesia kurang lebih 91 juta orang. Meningkatnya pengguna android yang cukup besar membuat sistem operasi android menjadi target serangan malware, sebuah sistem operasi yang telah terserang malware dapat mengalami kerusakan dan bahkan dengan niat yang lebih jahat malware dapat digunakan untuk pencurian data-data penting yang ada di sistem.

Malicious Software atau sering dikenal sebagai malware merupakan sebuah software yang diprogram agar dapat menyusup ke sebuah sistem operasi yang dapat merusak cara kerja sistem dan bahkan digunakan untuk mencuri data-data penting pada perangkat korban. Malware memiliki beberapa jenis dan mempunyai cara kerja yang beragam, salah satu contoh serangan malware melalui aplikasi berbahaya yang melakukan akses *permission* secara ilegal tanpa izin dari pengguna dan sistem operasi. Oleh karena rentannya serangan malware dan merugikan para pengguna android sehingga diperlukan analisis lebih lanjut terhadap malware. Pada tahun 2019, perusahaan keamanan siber Kaspersky melaporkan bahwa malware yang terdeteksi sebanyak 556.486 dan pada tahun 2020 perusahaan Kaspersky juga melaporkan terjadi penurunan serangan malware di indonesia sebanyak 31,89% sehingga menjadi 378,973. Oleh hal ini menjadikan Indonesia negara dengan jumlah anacama malware terbesar yang terdeteksi se-Asia Tenggara dan menduduki posisi ke-4 secara global [4]

Analisis statis dilakukan dengan membongkar source code dari malware tersebut lalu mempelajari dan memahami perilaku jahat melalui source code, data dan file biner, sehingga proses analisis statis tidak memerlukan eksekusi terhadap malware. Analisis statis digunakan untuk mengekstrak karakteristik dan mengidentifikasi setiap aplikasi, menggunakan ApkTool 2.0.3 [5].

Berdasarkan data serangan malware dan pengguna android yang sangat banyak, ini menjadi sebuah masalah maka penelitian ini mengimplementasikan sebuah sistem *machine learning* untuk mengetahui suatu metode klasifikasi yang menghasilkan akurasi yang tinggi dalam melakukan deteksi dini terhadap serangan malware. Beberapa penelitian sebelumnya telah menggunakan beberapa metode *machine learning* seperti *Naïve bayes, Decision Tree, KNN* dan metode lainnya , seperti yang dilakukan oleh Lopez dkk [5] telah melakukan penelitian menggunakan beberapa metode *machine learning* dengan menghasilkan akurasi yang tidak lebih dari 95%. Dengan dikembangkan metode-metode *machine learning* untuk melakukan klasifikasi, sehingga pada penelitian ini mengimplementasikan sebuah sistem *machine learning* untuk mengetahui suatu metode *machine learning* yang menghasilkan performansi matrik yang lebih unggul dari penelitian [5] dalam melakukan deteksi dini terhadap serangan malware.

Penelitian ini menggunakan metode klasifikasi *Support Vector Machine* (SVM) dan *Random Forest* untuk mengetahui metode mana yang terbaik untuk melakukan klasifikasi dengan baik pada dataset malware serta membandingkan hasil perfomasi matrik dengan penelitian sebelumnya yang dilakukan oleh Lopez A. et al [5]. Hasil penelitian ini diharapkan dapat membantu *developer google play store* dan pengguna dalam mengunduh aplikasi di *google play store*. Aplikasi yang ingin dipublikasikan oleh perorangan maupun perusahaan dapat diketahui apakah aplikasi tersebut berbahaya atau tidak. Sehingga aplikasi yang ada di *google play store* lebih terjamin dari serangan malware dan aman untuk diunduh oleh pengguna.

Topik dan Batasannya

Pada tugas akhir ini membahas masalah bagaimana cara membatasi android terkena malware dengan mengklasifikasi akses izin (*permission*) yang dilakukan secara ilegal. Deteksi pada tugas akhir ini menggunakan metode linear *Support Vector Machine* dan *Random Forest*. Dataset yang digunakan untuk mendeteksi malware adalah "*Dataset malware/begingn permission Android*".

Tujuan

Tujuan dari penelitian ini berdasarkan masalah yang ada adalah untuk mengimplementasikan sebuah algoritma *Support Vector Machine* (SVM) dan *Random Forest* untuk mengklasifikasikan dataset *permission* android (Malware), menganalisis data dengan membuat fitur umum dan fitur ciri dari dataset yang dipilih. Serta membandingkan hasil performansi matrik antara kedua metode tersebut dan membandingkan hasil performansi matrik dengan penelitian sebelumnya.

2. Studi Terkait

2.1 Penelitian terkait

Pada penelitian Lopez A. et al [5] membuat kerangka kerja yang menggunakan analisis statis untuk mendeteksi aplikasi berbahaya untuk android serta menggunakan teknik *machine learning*. Kerangka kerja yang diusulkan terdiri dari 4 fase: pengumpulan data, ekstraksi, pembuatan fitur, dan pelatihan & pengujian. Analisis statis digunakan untuk mengekstraksi fitur yang mengidentifikasi aplikasi. Pada penelitian yang dilakukan menggunakan 6 algoritma *machine learning*, klasifikasi yang digunakan dalam penelitian ini antara lain *Naive Bayes, Begging, KNN, SVM,* (*Stochastic Gradient Descent*) SGD, dan *Decision Tree*. Enam algoritma yang ditawarkan oleh penelitian ini menghasilkan akurasi, *Naïve Bayes* 90%, *Bagging* 93%, *KNN* 94%, *SVM* 94%, *SGD* 92%, dan *Decision Tree* 94%.

D. O. Sahin. et al [6] melakukan riset deteksi malware android berbasis *permission*. Mereka menyampaikan bahwa penelitian ini tidak seperti penelitian lainnya, mereka mengusulkan pendekatan bobot *permission*. Setiap *permission* diberikan skor yang berbeda melalui pendekatan ini, kemudian menerapkan algoritma KNN dan NB. Sumber dataset penelitian ini dari penelitian yang dilakukan oleh Lopez A. et al[5]. Metode yang mereka usulkan dilakukan perbandingan dengan studi-studi sebelumnya. Menurut hasil percobaan, pendekatan yang mereka usulkan memiliki hasil yang lebih baik dari penelitian sebelumnya, dengan hasil akurasi KNN 94,95% dan NB 96,63%.

Tao Ban et al [7] melakukan penelitian deteksi malware android menggunakan linear SVM. Pada penelitian ini mereka mengeksplorasi potensi fitur multi-modal untuk meningkatkan akurasi deteksi. Fitur yang diperiksa meliputi *permission*, API, dan APK. Fitur multi-modal ini dibuat untuk memfasilitasi pembelajaran dan pengujian yang efisien dengan pengklasifikasi SVM linear. Eksperimen menunjukkan bahwa metode yang diusulkan oleh peneliti mendapatkan akurasi lebih dari 94% dan hemat waktu dalam pengujian untuk data skala besar dan dimensi tinggi.

Hui-Juan Z et al [8] merancang sebuah *ensemble* sistem *machine learning* untuk mendeteksi malware berbasis *Random Forest* di perangkat android. 4 kelompok fitur yang dideteksi pada perangkat android antara lain *permission, sensitive API, system monitor event, permission-rate*. Klasifikasi *Random Forest ensemble* di *training* untuk mendeteksi apakah suatu aplikasi berpotensi berbahaya atau tidak. Kinerja metode yang diusulkan dievaluasi pada kumpulan data aktual menggunakan 10 *cross-validation*. Hasil dari penelitian yang dilakukan menunjukkan akurasi yang sangat tinggi sebesar 89%.

Junmei S. et al [13] merancang sebuah metode untuk mendeteksi malware android menggunakan *keywords vector* dan SVM. latar belakang penelitian ini dilakukan adalah mendeteksi malware baru dan perangkat lunak berbahaya menjadi masalah yang sulit. Makalah ini menyajikan metode untuk ekstraksi fitur kode sumber JAVA. Metode ini menggunakan *keywords vector* untuk menghitung korelasi antara kode seperti API, *permission* android, parameter umum dan kode sumber malware android. Kemudian SVM diterapkan ke sistem untuk mengakomodasi fungsi sampel perangkat lunak berbahaya dan mendeteksi perangkat lunak berbahaya serta malware. Cara ini berbeda dengan cara metode konvensional. Metode ini menggabungkan kerateristik yang jahat, kategori perangkat lunak dan lingkungan operasi untuk merekam perilaku perangkat lunak berbahaya. Percobaan menunjukkan bahwa metode ini efisien dan efektif dalam mendeteksi malware di platform android dengan hasil akurasi 88%.

Yuki T. et al [14] merancang sebuah sistem untuk mendeteksi malware *ransomware* menggunakan SVM. *Ransomware* merupakan perangkat lunak berbahaya yang mengenkripsi file di *device* korban dan meminta uang sebagai tebusan untuk mendekripsi file tersebut. Ini menjadi kerugian finansial individu maupun organisasi karena *ransomware* meningkat dari tahun ke tahun. Oleh karena itu mendeteksi *ransomware* merupakan hal yang penting, mereka mengusulkan sebuah skema deteksi *ransomware* menggunakan SVM. Ide skema yang diusulkan membiarkan SVM mempelajari API *ransomware* sebagai fitur-fitur sehingga SVM dapat mendeteksi *ransomware* yang tidak terlihat. Skema ini berbeda dengan sebelumnya karena skema mereka melihat riwayat panggilan API secara lebih rinci. Mereka menggunakan *testbeds ransomware* 276 dengan sandbox menunjukkan skema yang diusulkan meningkatkan deteksi *ransomware* dengan akurasi 97,48%.

Nikola M. et al [15] membangun sebuah metode klasifikasi malware android yang menggunakan *machine learning* untuk analisis statis malware android. Kemampuan android untuk mengakses informasi pribadi dan rahasia mengakibatkan perangkat ini menjadi sasaran pengembang malware. Pendekatan pertama didasarkan analisis *permission* dan lainnya analisis kode menggunakan model bag-of-words.

Model berbasis *permission* dan mengimplementasikan sebagai fitur dari aplikasi android OWASP yang diperoleh dari google *play store*. Evaluasi yang mereka dapatkan terhadap 2 pendekatan SVM dan *ensemble learning* menunjukkan akurasi SVM 95,1% dan *ensemble learning* 95,6%.

Alif P. et al [16] membangun sebuah sistem deteksi malware menggunakan metode *Random Forest* berdasarkan analisis forensik. Malware memiliki beberapa tujuan jahat berupa pencurian data, perusakan sistem, dan menolak layanan sehingga dibutuhkan sistem untuk deteksi malware tersebut. Pada proses analisis malware digunakan analisis statis dan dinamis sehingga karakteristik dari masing-masing perangkat lunak dapat diketahui. Data yang telah didapatkan diklasifikasikan menggunakan *Random Forest* untuk membedakan perangkat lunak jinak dan malware. Proses klasifikasi menggunakan *Random Forest* dengan data yang dibagi menjadi 80% *training* dan 20% *testing* menghasilkan akurasi 99% dan 0,8 detik untuk analisis statis dan untuk dinamis memiliki akurasi 76% dan 0,64 detik.

Ary Adhigana S. et al [17] melakukan penelitian menganalisis *ensembel* untuk mendeteksi malware pada *mobile devices*. Pada penelitian ini menggunakan *machine learning* dalam mendeteksi malware pada android. Pada penelitian ini menggunakan dataset Drebin, dimana dataset tersebut berbasis fitur perizinan pada aplikasi *mobile device*. Metode KNN, Random Forest, dan Naive Bayes dengan metode ensemble adalah metode yang digunakan penelitian. Hasil pengujian adalah metode ensemble menghasilkan tingkat akurasi 98,4%.

Mahendra D. et al [18] melakukan perbandingan SVM dan *modified balanced Random Forest* dalam deteksi pasien penyakit diabetes. Diabetes merupakan penyakit mematikan nomor 3 di indonesia dengan persentase sebesar 6,7%. Tingginya tingkat kematian akibat diabetes mendorong dilakukan penelitian ini dengan tujuan deteksi dini. Pada penelitian ini menggunakan pendekatan *machine learning* untuk klasifikasi data. Dalam makalah ini, dibahas perbandingan SVM dan *modified balanced Random Forest* (MBRF) untuk melakukan klasifikasi data pasien diabetes. Berdasarkan hasil eksperimen dalam proses pengujian sistem yang dibangun, diperoleh hasil performansi maksimum 87,94% untuk SVM dan 97,8 dengan menggunakan MBRF.

2.2 Metode Algoritma

Pada penelitian ini metode Algoritma yang digunakan adalah algoritma *Support Vector Machine* (SVM) dan *Random Forest*. Kedua metode ini dipilih karena pada penelitian-penelitian sebelumnya terbukti kedua metode tersebut sangatlah efektif melakukan klasifikasi. Pada bagian ke 2, hasil *literature review* yang telah dilakukan terhadap 10 *paper* bahwa metode *Support Vector Machine* (SVM) dan *Random Forest* memiliki hasil akurasi yang tinggi. Hasil dari penelitian yang dilakukan oleh Tao Ban et al [7] melakukan deteksi malware android menggunakan linear SVM menghasilkan akurasi 94%. Pada penelitian yang dilakukan oleh Alif P. et al [16] membangun sebuah sistem deteksi malware menggunakan *Random Forest* Menghasilkan akurasi 99%.

2.2.1 Support Vector Machine (SVM)

Support Vector Machine atau yang sering disebut SVM adalah salah satu metode pada *machine learning* yang sering digunakan untuk mengklasifikasikan data. Support vector machine (SVM) pertama kali diperkenalkan pada tahun 1992 Bernhard Boser, Isabelle Guyon, Vladimir Vanik [9]. SVM memiliki konsep dasarnya merupakan gabungan dari teori komputasi, seperti hyperplane, kernel, dan konsep pendukung lainnya. Prinsip dasar SVM adalah klasifikasi linear, kemudian dikembangkan agar dapat bekerja pada problem non-linear dengan konsep *kernel trick* pada data berdimensi tinggi. SVM bersifat *Supervised Learning* ataupun dapat dikontrol data *training* dan *testing* untuk mendapatkan keakuratan klasifikasi. Konsep kerja dari SVM ketika pertama kali digunakan/dikembangkan oleh Boser, Guyon, Vapnik pada tahun 1992 hanya dapat menangani klasifikasi data yang memiliki 2 kelas tapi seiring waktu SVM telah dikembangkan dalam klasifikasi data yang memiliki lebih dari 2 kelas dengan *Pattern Recognition* pada ruang kerja berdimensi tinggi. Proses klasifikasi pada SVM adalah menentukan *hyperplane* terbaik untuk pemisah dua kelas data, data yang paling dekat dengan *hyperplane* disebut *support vector*. Untuk mendapatkan *hyperplane* terbaik dengan cara menentukan *max margin*. *Margin* adalah jarak antara *hyperplane* dengan *support vector.*

2.2.2 Random Forest

Random Forest merupakan salah satu algoritma *machine learning* untuk klasifikasi data dalam jumlah yang besar. *Random Forest* pertama kali diusulkan oleh Tin Kam Ho pada tahun 1995[10] dan *Random Forest* pertama kali dipublikasikan oleh Leo Breiman dengan beberapa persiapan pada tahun 2001[12]. *Random Forest* bersifat *ensemble learning*, *ensemble learning* adalah penggabungan dari beberapa algoritma *machine learning* untuk mendapatkan solusi prediksi yang lebih baik. Metode ini melakukan klasifikasi dengan penggabungan model *tree*. *Random Forest* mampu mengklasifikasi data yang memiliki atribut yang tidak lengkap dan cocok digunakan untuk pengklasifikasian data sampel yang banyak. Proses klasifikasi pada *Random Forest* melakukan *split* (memecah) data sampel yang ada ke dalam *decision tree* secara acak. Setelah *tree* terbentuk, pada tree terdapat *root* (akar), *internal node* (cabang-cabang), *leaf* (hasil kelas) [16]. kemudian diambil *tree* yang terbaik dengan menggunakan *Random Forest*, sehingga klasifikasi data menghasilkan akurasi yang terbaik. Pada pengujian *Random Forest* yang dilakukan jumlah pohon yang digunakan adalah 100 model pohon [17].

2.3 Permission di Dataset

Pada penelitian ini menggunakan dataset [11], terdiri dari 398 aplikasi dan 330 daftar *permission* sebagai fitur, setiap aplikasi dianalisis terhadap 330 *permission*. Aplikasi yang dianalisis diberikan variabel biner (1/0), jika terdapat *permission* pada aplikasi diberikan angka 1, jika tidak terdapat *permission* diberikan angka 0. Setelah dianalisis setiap aplikasi dideteksi menggunakan ApkTool, dari 398 aplikasi yang di deteksi terdapat 199 APK malware dan 199 APK jinak.

Pada APK malware terlihat sangat dominan memiliki *permission*. Dari 330 daftar *permission*, ada beberapa APK malware yang meminta akses ke fitur *permission*. Ada 10 *permission* dominan yang terdapat pada APK malware beserta dengan kegunaan *permission* tersebut [19][20], diantaranya:

- 97,9% meminta akses ke fitur “android.permission.INTERNET”, dari 199 APK malware 195 diantaranya terdapat android.permission.INTERNET. *Permission* ini aplikasi meminta akses ke jaringan. Tingkat perlindungan : Terbuka.
- 95,4% meminta akses ke fitur “android.permission.READ_PHONE_STATE”, dari 199 APK malware 190 diantaranya terdapat android.permission. READ_PHONE_STATE. *Permission* ini mengizinkan akses hanya baca ke status telepon, status panggilan yang sedang berlangsung dan daftar akun telepon apapun yang terdaftar di perangkat. Tingkat perlindungan : Tertutup.
- 83,9% meminta akses ke fitur “android.permission.ACCESS_NETWORK_STATE”, dari 199 APK malware 167 diantaranya terdapat android.permission.ACCESS_ NETWORK_STATE. *Permission* ini mengizinkan aplikasi mengakses informasi tentang jaringan. Tingkat perlindungan : Terbuka.
- 68,3% meminta akses ke fitur “android.permission.WRITE_EXTERNAL_STORAGE”, dari 199 APK malware 136 diantaranya terdapat android.permission.WRITE_ EXTERNAL_STORAGE. *Permission* ini mengizinkan aplikasi menulis ke penyimpanan eksternal. Tingkat perlindungan : Tertutup.
- 67,8% meminta akses ke fitur “android.permission.ACCESS_WIFI_STATE”, dari 199 APK malware 135 diantaranya terdapat android.permission.ACCESS_WIFI_STATE. *Permission* ini mengizinkan aplikasi mengakses informasi tentang jaringan WIFI. Tingkat perlindungan : Terbuka.
- 62,3% meminta akses ke fitur “android.permission.READ_SMS” dari 199 APK malware 124 diantaranya terdapat android.permission.READ_SMS. *Permission* ini mengizinkan aplikasi membaca pesan SMS. Tingkat perlindungan : Tertutup.
- 52,2% meminta akses ke fitur “android.permission.WRITE_SMS” dari 199 APK malware 104 diantaranya terdapat android.permission.WRITE_SMS. *Permission* ini mengizinkan aplikasi menulis pesan SMS. Tingkat perlindungan : Tertutup.
- 51,2% meminta akses ke fitur “android.permission.RECEIVE_BOOT_COMPLETED” dari 199 APK malware 102 diantaranya terdapat android.permission. RECEIVE_BOOT_COMPLETED. *Permission* ini mengizinkan aplikasi menerima

Intent.ACTION_BOOT_COMPLETED setelah sistem selesai booting. Tingkat perlindungan : Terbuka.

- 40,2% meminta akses ke fitur "android.permission.ACCESS_COARSE_LOCATION" dari 199 APK malware 80 diantaranya terdapat android.permission. ACCESS_COARSE_LOCATION. *Permission* ini mengizinkan aplikasi mengakses perkiraan lokasi. Tingkat Perlindungan : Tertutup.
- 35,1% meminta akses ke fitur "android.permission.CHANGE_WIFI_STATE" dari 199 APK malware 70 diantaranya terdapat android.permission.CHANGE_WIFI_STATE. *Permission* ini mengizinkan aplikasi mengubah status konektivitas WIFI. Tingkat perlindungan : Terbuka.

Pada 10 *permission* dominan yang terdapat di APK malware, 5 diantaranya memiliki tingkat perlindungan yang tertutup. APK yang memiliki *permission* yang memiliki tingkat perlindungan tertutup sangatlah dipantau. Sehingga diketahui APK memiliki *permission* itu untuk melakukan akses legal atau ilegal

Pada APK jinak terdapat beberapa *Permission* yang sangat dominan, Dari 330 daftar *permission*, ada beberapa APK jinak yang meminta akses ke fitur *permission*. ada 10 *permission* yang dominan terdapat pada APK malware, diantaranya:

- 52,2% meminta akses ke fitur "android.permission.INTERNET", dari 199 APK jinak 104 diantaranya terdapat android.permission.INTERNET. *Permission* ini aplikasi meminta akses ke jaringan. Tingkat perlindungan : Terbuka.
- 38,1% meminta akses ke fitur "android.permission.WRITE_EXTERNAL_STORAGE", dari 199 APK jinak 76 diantaranya terdapat android.permission.WRITE_ EXTERNAL_STORAGE. *Permission* ini mengizinkan aplikasi menulis ke penyimpanan eksternal. Tingkat perlindungan : Tertutup.
- 31,1% meminta akses ke fitur "android.permission.ACCESS_NETWORK_STATE", dari 199 APK jinak 62 diantaranya terdapat android.permission.ACCESS_ NETWORK_STATE. *Permission* ini mengizinkan aplikasi mengakses informasi tentang jaringan. Tingkat perlindungan : Terbuka.
- 18% meminta akses ke fitur "android.permission.WAKE_LOCK", dari 199 APK jinak 36 diantaranya terdapat android.permission.WAKE_LOCK. *Permission* ini memungkinkan pengguna untuk menjaga prosesor dari tidur atau layar peredupan. Tingkat perlindungan : Terbuka.
- 15% meminta akses ke fitur "android.permission.RECEIVE_BOOT_COMPLETED", dari 199 APK jinak 30 diantaranya terdapat android.permission. RECEIVE_BOOT_COMPLETED. *Permission* ini mengizinkan aplikasi menerima Intent.ACTION_BOOT_COMPLETED setelah sistem seselai booting. Tingkat perlindungan : Terbuka.
- 14,5% meminta akses ke fitur "android.permission.CHANGE_WIFI_STATE" dari 199 APK jinak 29 diantaranya terdapat android.permission.CHANGE_WIFI_STATE. *Permission* ini mengizinkan aplikasi mengubah status konektivitas WIFI. Tingkat perlindungan : Terbuka.
- 12% meminta akses ke fitur "android.permission.READ_PHONE_STATE", dari 199 APK jinak 24 diantaranya terdapat android.permission. READ_PHONE_STATE. *Permission* ini mengizinkan akses hanya baca ke status telepon, status panggilan yang sedang berlangsung dan daftar akun telepon apapun yang terdaftar di perangkat. Tingkat perlindungan : Tertutup.
- 10% meminta akses ke fitur "android.permission.VIBRATE" dari 199 APK jinak 21 diantaranya terdapat android.permission.VIBRATE. *Permission* ini mengizinkan akses ke vibrator. Tingkat perlindungan : Terbuka.
- 9% meminta akses ke fitur "android.permission.ACCESS_FINE_LOCATION" dari 199 APK jinak 18 diantaranya terdapat android.permission.ACCESS_FINE_

LOCATION. *Permission* ini mengizinkan aplikasi mengakses lokasi yang tepat. Tingkat perlindungan : Tertutup.

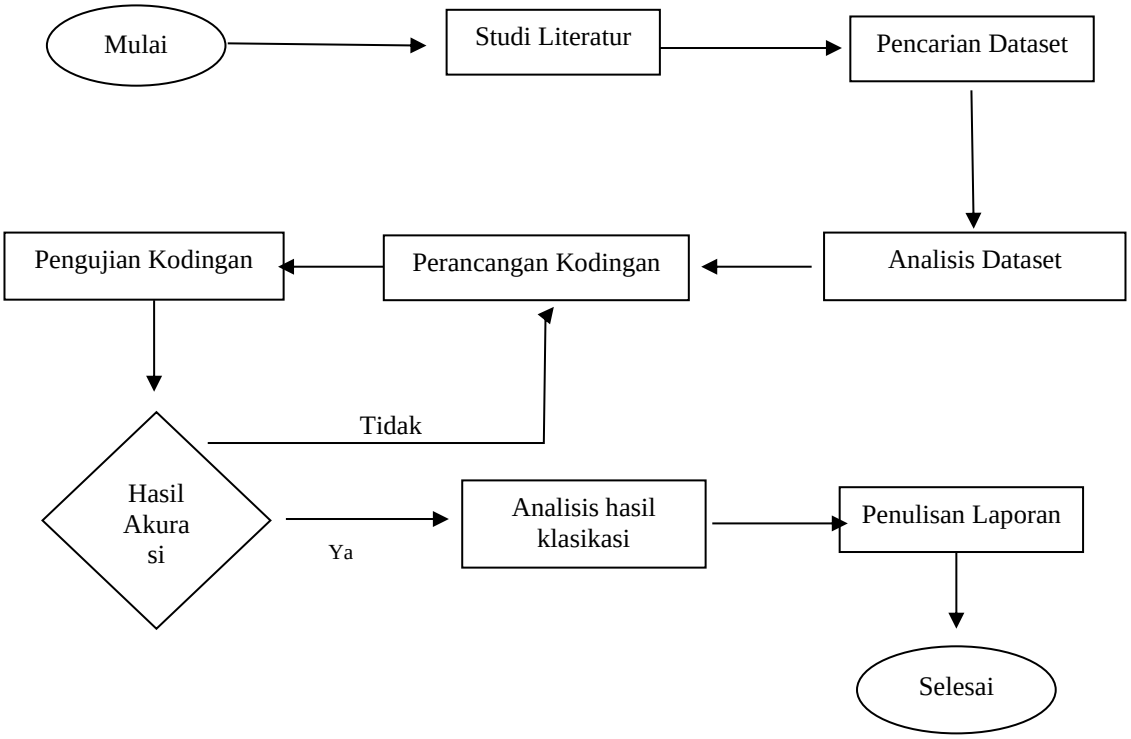
- 7,5% meminta akses ke fitur “android.permission.READ_EXTERNAL_STORAGE” dari 199 APK jinak 15 diantaranya terdapat android.permission. READ_EXTERNAL_STORAGE. *Permission* ini mengizinkan aplikasi membaca dari penyimpanan. Tingkat perlindungan : Tertutup.

Pada 10 *permission* dominan yang terdapat di APK malware, 4 diantaranya memiliki tingkat perlindungan yang tertutup. APK yang memiliki *permission* yang memiliki tingkat perlindungan tertutup sangat dipantau. Sehingga diketahui APK memiliki *permission* itu untuk melakukan akses legal atau ilegal

Dari penjelasan ini, kita dapat mengetahui kedua sampel aplikasi jinak dan malware menampilan bahwa sebagian besar aplikasi malware meminta akses ke fitur *permission* akses internet, mengetahui status telepon, memonitor jaringan, ke luar direktori dan mengelola semua aspek konektivitas WIFI. Untuk mengetahui lebih jauh kegunaan dan tingkat perlindungan dari sebuah *permission* dalam dataset ini dapat dicek di [19][20].

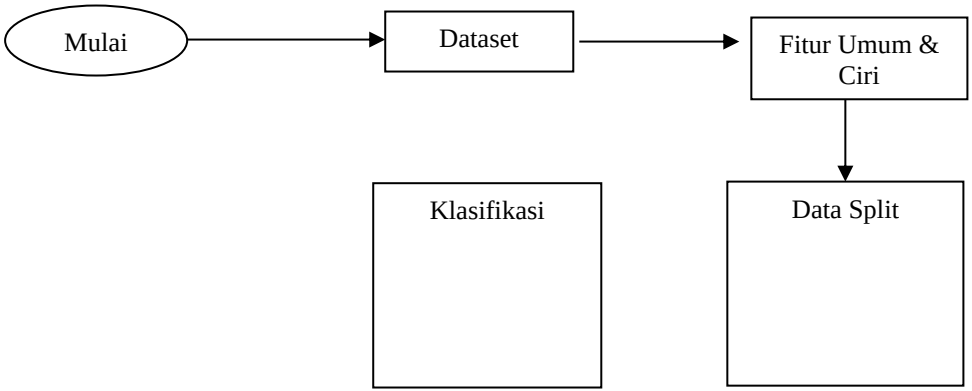
3. Sistem yang Dibangun

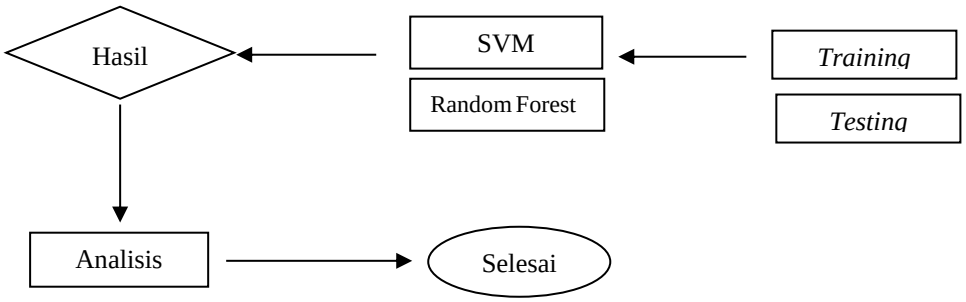
3.1 Framework Penelitian



Gambar 1. Framework Penelitian

3.2 Desain Sistem





Gambar 2. Desain Sistem

3.3 Dataset

Penelitian ini menggunakan dataset yang dipublikasikan oleh Lopez A. et al di web kaggle[11] dengan judul "*Dataset malware/begingn permission Android*" yang dipublikasikan pada tahun 2016. Pada dataset ini berisi 199 aplikasi malware dan 199 aplikasi jinak. 330 daftar *permission* telah diberikan sebagai fitur. Setiap aplikasi yang di analisis diberikan variabel biner (1 atau 0), jika terdapat *permission* di aplikasi diberikan angka 1, jika tidak terdapat *permission* diberikan angka 0. Setelah dianalisis *permission*, selanjutnya dilakukan deteksi terhadap aplikasi, setiap aplikasi yang di deteksi diberikan variabel biner di kolom *type*, jika aplikasi itu berbahaya diberikan angka 1 dan sebaliknya jika aplikasi itu tidak berbahaya diberikan angka 0. Lopez A. et al [5] mengklaim bahwa mereka melakukan deteksi 558 APK meskipun sampel yang dibagikan hanya 398 APK di web kaggle[11].

3.4 Fitur Dataset

Penelitian yang dilakukan oleh Lopez et al[5] yang sebagai paper rujukan, pada penelitian tersebut membangun sebuah metode algoritma dengan menggunakan semua data atribut sebagai fitur metode. Pada penelitian tersebut menggunakan semua atribut sebagai fitur metode dikarenakan penganalisa sengat berhati-hati untuk membuat variabel biner pada data atribut tersebut. Penganalisa menganalisis APK, apakah pada APK tersebut terdapat *permission* dari 330 daftar *permission* yang dimiliki. Sehingga penelitian ini menggunakan semua atribut sebagai fitur di dalam metode algoritma.

3.4.1 Fitur Umum

Melalui pengembangan analisis *permission*, Lopez A. et al[5] melakukan deteksi menggunakan ApkTool 2.0.3 ke 558 APK terhadap daftar 330 *permission*. Hasil analisis *permission* diberikan variabel biner dengan sangat teliti untuk masing-masing *permission* dari daftar. Nilai biner diberikan tergantung hasil deteksi APK terdapat *permission* atau tidak.

$$\begin{aligned} Ri = 1 \quad & \text{Terdapat } permission \text{ di APK} \\ Ri = 0 \quad & \text{Tidak terdapat } permission \text{ di APK} \end{aligned} \tag{1}$$

Setelah selesai memberikan variabel biner terhadap setiap *permission*, selanjutnya diberikan label kelas/type dari APK, kelas malware atau tidak.

$$\begin{aligned} Ci = 1 \quad & \text{Jika APK berbahaya} \\ Ci = 0 \quad & \text{Jika APK tidak berbahaya} \end{aligned} \tag{2}$$

Setiap aplikasi diberikan vektor yang didefinisikan sebagai V = {R1, R2, … , R330, C} yang berisi informasi variabel *permission* dan label kelas/type.

Permission pada APK bisa sangat berbahaya karena *permission* bisa saja bersifat *stowaway* (penumpang gelap). *Stowaway* (penumpang gelap) adalah aplikasi yang mencoba melakukan operasi yang tidak memiliki izin dari pengguna dan sistem operasi. Aplikasi beroperasi dibawah sistem berbasis *permission*, dimana aplikasi harus diberikan izin oleh pengguna dan sistem operasi ke berbagai area fungsionalitas sebelum aplikasi dapat digunakan. Contohnya lokasi GPS, informasi, kontak, atau melakukan panggilan telepon.

3.4.2 Fitur Ciri

Pada bagian ini menjelaskan fitur ciri dari dataset. Pada penelitian ini, semua atribut dataset digunakan sebagai fitur, fitur ciri ini digunakan ke dalam metode untuk dideteksi. Mendeteksi APK malware atau jinak. Pada tabel 1 adalah 10 *permission* yang hanya ada di APK malware dan 10 *permission* yang hanya ada di APK jinak.

Tabel 1. 10 *Permission* hanya ada di APK Malware dan Jinak

Malware		**Jinak**	
Permission	APK	Permission	APK
android.permission. RECEIVE_MMS	13	android.permission. AUTHENTICATE_ACCOUNTS	7
android.permission. .RECEIVE_WAP_PUSH	11	android.permission.NFC	7
com.android.launcher.permission. WRITE_SETTINGS	9	android.permission. USE_CREDENTIALS	7
android.permission. GLOBAL_SEARCH_CONTROL	8	android.permission. READ_CALL_LOG	4
android.permission. SET_WALLPAPER_HINTS	8	android.permission. READ_SYNC_STATS	4
android.permission. CHANGE_CONFIGURATION	6	android.permission. MANAGE_ACCOUNTS	3
android.permission. GET_PACKAGE_SIZE	4	android.permission. READ_CALENDAR	3
android.permission. ACCESS_CACHE_FILESYSTEM	2	android.permission. SYSTEM_ALERT_WINDOW	3
android.permission. ACCESS_WIMAX_STATE	2	android.permission. WRITE_CALENDAR	3
android.permission. CHANGE_WIMAX_STATE	2	com.android.alarm.permission. SET_ALARM	3

Pada top 10 *permission* tersebut hanya ada di aplikasi yang terdeteksi malware dan setiap *pemission* memiliki kegunaan [19][20],antara lain:

- 13 APK meminta akses ke fitur “android.permission.RECEIVE_MMS”. *Permission* ini Mengizinkan aplikasi memantau pesan MMS yang masuk. Tingkat perlindungan: Tertutup.
- 11 APK meminta akses ke fitur “android.permission.RECEIVE_WAP_PUSH”. *Permission* ini Mengizinkan aplikasi menerima pesan *push* WAP. Tingkat perlindungan: Tertutup.
- 9 APK meminta akses ke fitur “com.android.permission.WRITE_SETTINGS”. *Permission* ini Mengizinkan aplikasi mengubah setelan dan pintasan di beranda. Tingkat perlindungan: Tertutup.
- 8 APK meminta akses ke fitur “android.permission.GLOBAL_SEARCH _CONTROL”. *Permission* ini dapat digunakan pada penyedia konten untuk memungkinkan sistem mengontrol pencarian global mengakses data mereka. Tingkat perlindungan: Tanda Tangan|Hak Istimewa.
- 8 APK meminta akses ke fitur “android.permission.SET_WALLPAPER_HINTS”. *Permission* ini Mengizinkan aplikasi menyetel petunjuk wallpaper. Tingkat perlindungan: Terbuka.
- 6 APK meminta akses ke fitur “android.permission.CHANGE_ CONFIGURATION” *Permission* ini Mengizinkan aplikasi mengubah konfigurasi saat ini,seperti lokal. Tingkat perlindungan: Tanda Tangan|Hak Istimewa|Pengembangan .

- 4 APK meminta akses ke fitur "android.permission.GET_PACKAGE_SIZE". *Permission* ini Mengizinkan aplikasi mengetahui ruang yang digunakan oleh paket apa pun. Tingkat perlindungan: Terbuka.
- 2 APK meminta akses ke fitur "android.permission.ACCESS_CACHE_ FILESYSTEM". *Permission* ini Mengizinkan aplikasi mengetahui ruang yang digunakan oleh paket apa pun. Tingkat perlindungan: Terbuka.
- 2 APK meminta akses ke fitur "android.permission.ACCESS_WIMAX_STATE". *Permission* ini Mengizinkan aplikasi menentukan apakah WIMAX diaktifkan dan informasi tentang jaringan WIMAX apapun yang terhubung. Tingkat perlindungan: Tertutup.
- 2 APK meminta akses ke fitur "android.permission.CHANGE_WIMAX_STATE". *Permission* ini Mengizinkan aplikasi menghubungkan dan memutuskan sambungan ponsel dari jaringan WIMAX . Tingkat perlindungan: Tertutup.

Pada 10 *permission* malware memiliki kegunaannya masing-masing dan memiliki Tingkat perlindungan. Pada 10 *permission* malware kebanyakan memiliki tingkat perlindungan tertutup, tanda tangan/hak istimewa|pengembangan. *Permission* tersebut sangat dipantau, kemungkinan *permission* tersebut melakukan akses ilegal untuk kepentingan tertentu sehingga APK itu dilakukan deteksi menggunakan ApkTool, setelah dilakukan deteksi menggunakan *tools* tersebut didapatkan hasil bahwa aplikasi yang terdapat *permission* tersebut terdeteksi berbahaya dan diberikan label berbahaya ataupun APK tersebut terdeteksi malware.

Pada 10 *permission* jinak tersebut hanya ada di aplikasi yang terdeteksi jinak dan setiap *pemission* memiliki kegunaan [19][20],antara lain:

- 7 APK meminta akses ke fitur "android.permission.AUTHENTICATE _ACCOUNTS". *Permission* ini Mengizinkan aplikasi menggunakan kemampuan mengautentikasi akun. Tingkat perlindungan: Terbuka.
- 7 APK meminta akses ke fitur "android.permission.NFC". *Permission* ini Mengizinkan aplikasi melakukan operasi I/O melalui NFC. Tingkat perlindungan: Terbuka.
- 7 APK meminta akses ke fitur "android.permission.USE_CREDENTIALS". *Permission* ini Mengizinkan aplikasi melakukan operasi I/O melalui NFC. Tingkat perlindungan: Terbuka.
- 4 APK meminta akses ke fitur "android.permission.READ_CALL_LOG". *Permission* ini Mengizinkan aplikasi membaca log panggilan pengguna. Tingkat perlindungan: Terbuka.
- 4 APK meminta akses ke fitur "android.permission.READ_SYNC_STATS". *Permission* ini Mengizinkan aplikasi membaca statistik sinkronisasi. Tingkat perlindungan: Terbuka.
- 3 APK meminta akses ke fitur "android.permission.MANAGE_ACCOUNTS". *Permission* ini Mengizinkan aplikasi melakukan operasi seperti menambah dan menghapus akun, serta menghapus sandinya. Tingkat perlindungan: Terbuka.
- 3 APK meminta akses ke fitur "android.permission.READ_CALENDAR *Permission* ini Mengizinkan aplikasi membaca data kelender pengguna. Tingkat perlindungan: Tertutup.
- 3 APK meminta akses ke fitur "android.permission.SYSTEM_ALERT_WINDOW *Permission* ini Mengizinkan aplikasi membuat jendela menggunakan tipe. Tingkat perlindungan: Tertutup.
- 3 APK meminta akses ke fitur "android.permission.WRITE_CALENDAR". *Permission* ini Mengizinkan aplikasi menulis data kelender pengguna. Tingkat perlindungan: Tertutup.
- 3 APK meminta akses ke fitur "android.permission.SET_ALARM". *Permission* ini Mengizinkan aplikasi menyetel alarm bagi pengguna. Tingkat perlindungan: Terbuka.

Pada 10 *permission* jinak memiliki kegunaannya masing-masing dan memiliki Tingkat perlindungan. Pada 10 *permission* jinak kebanyakan memiliki tingkat perlindungan terbuka dan beberapa *permission* memiliki tingkat keamanan tertutup. Tingkat keamanan tertutup sangat dipantau.

Pada 10 *permission* jinak yang terdapat di beberapa APK, dideteksi bahwa *permission* itu tidak melakukan akses ilegal sehingga APK yang terdapat *permission* tersebut diberikan label jinak ataupun APK tersebut aman.

Tabel 2. Total dan Rata-rata permission Malware &

	Malware	Jinak
Total APK	199	199
Total Permission	2429	657
Rata-rata Permission	12,2	3,3

Pada tabel 2 ini menunjukkan total seluruh dari *permission* yang ada di 199 aplikasi malware & jinak, jumlah rata-rata seluruh *permission* yang ada di 199 aplikasi malware & jinak. Pada tabel 2 juga menunjukkan bahwa total dari seluruh *permission* dan rata-rata *permission* lebih besar pada APK malware dibandingkan pada APK jinak.

3.5 Split Data

Pada tahap ini dilakukan *split* atau membagi data. Data dibagi menjadi dua bagian yaitu data *training* dan *testing*. Data *testing* digunakan untuk menjadi bahan pembelajaran ataupun latihan pada metode *machine learning* sehingga metode dapat bekerja dengan baik ketika melakukan klasifikasi. Data *testing* digunakan data yang dijadikan sebagai data uji terhadap metode sehingga dapat mengetahui seberapa akurat metode yang digunakan.

Pada penelitian melakukan *split* terhadap 398 baris dataset sebanyak tiga kali, tiga skenario itu digunakan sebagai data *training* dan data *testing* terhadap metode *machine learning*. Skenario pertama melakukan *split* menjadi 90% data *training* (latihan) dan 10% data *testing* (uji), skenario kedua terdapat 80% data *training* (latihan) dan 20% data *testing* (uji), skenario ketiga terdapat 70% data *training* (latihan) dan 30% data *testing* (uji).

3.6 Klasifikasi

Metode klasifikasi yang digunakan pada penelitian ini adalah metode *Support Vector Machine* (SVM) dan *Random Forest*. Seperti yang dijelaskan pada bagian 2, bahwa kedua metode ini sangat direkomendasikan karena pada penelitian sebelumnya terbukti menghasilkan akurasi yang tinggi. Metode *Support Vector Machine* (SVM) sangat sesuai untuk digunakan pada penelitian ini karena dataset yang digunakan berisi angka biner, memiliki 2 kelas. Metode *Random Forest* digunakan pada penelitian ini karena memiliki sifat *ensemble learning*, membangun pohon keputusan.

3.7 *Confusion Matrix*

Confusion matrix digunakan untuk menilai hasil kinerja pengklasifikasi dalam penelitian ini. *Confusion matrix* yang digunakan sama seperti metrik uji pada penelitian-penelitian klasifikasi yang telah ada sebelumnya. Penelitian ini menggunakan ukuran penilaian sebagai berikut :

- *Accuracy*

  Nilai *accuracy* (akurasi) adalah nilai untuk mengetahui seberapa akurat sistem mengklasifikasikan data tersebut secara benar.

  $$Accuracy = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}} \quad (3)$$

- *Precision*

  Nilai *precision* (presisi) adalah nilai untuk mengetahui jumlah data positif yang diklasifikasikan secara benar dibagi total data yang diklasifikasi positif.

  $$Precision = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \quad (4)$$

- *Recall*

Nilai *recall* adalah nilai untuk mengetahui berapa persen data kategori positif yang diklasifikasikan dengan benar oleh sistem.

$$Recall = \frac{\text{TP}}{\text{TP+FN}} \quad (5)$$

- *F1-Score*

  Nilai *F1-score* adalah nilai *harmonic mean* dari presisi dan recall. Nilai terbaik *f1-score* adalah 1.0 dan terburuknya adalah 0. Jika Nila f1-score punya skor baik maka mengindikasikan bahwa metode klasifikasi yang dibangun memiliki presisi dan *recall*.

$$F1\ Score = 2\text{x}\frac{\text{Precision x Recall}}{\text{Precision+ Recall}} \quad (6)$$

Tabel 3. Confusion Matrix

Kelas	Terklasifikasi Positif	Terklasifikasi Nagatif
Positif	TP (True Positive)	FN (False Negative)
Nagatif	FP (False Positive)	TN (True Negative)

Berdasarkan nilai TP (*True Positive*), TN (*True Negative*), FP (*False Positive*), FN (*False Negative*) dapat diperoleh nilai akurasi, presisi, dan *recall*.

3.8 Analisis

Setelah didapatkan hasil akurasi dari kedua metode *machine learning* yang diimplementasikan. Pada tahap ini dilakukan proses penyelidikan terhadap hasil performansi dari klasifikasi untuk mengetahui metode *machine learning* mana yang terbaik untuk melakukan klasifikasi pada dataset tersebut.

4. Evaluasi.

Pada penelitian ini menggunakan semua atribut sebagai fitur di dalam metode algoritma seperti yang telah dijelaskan pada bagian 3.3. Penelitian ini dilakukan tiga skenario seperti yang telah dijelaskan pada bagian 3.4, tiga skenario itu antara lain, skenario pertama dengan data *training* 90% dan 10% data *testing*, skenario kedua dengan data *training* 80% dan *testing* 20%, dan skenario ketiga dengan data *training* 70% dan *testing* 30%. Tiga skenario digunakan bertujuan untuk melihat performansi metode dalam menangani ketiga skenario, sehingga mengetahui metode yang terbaik.

4.1 Hasil Pengujian

Pada penelitian ini mendapatkan hasil pengujian bahwa metode *Random Forest* dalam ketiga skenario yang ada mendapatkan hasil akurasi lebih unggul dari pada metode *Support Vector Machine* (SVM) dalam melakukan klasifikasi data. Dengan menghasilkan akurasi pada skenario pertama *Random Forest* 98,99% sedangan *Support Vector Machine* (SVM) 96,73%. Hal ini terjadi karena cara kerja *Random Forest* yang membangun beberapa model *tree* dan memilih *tree* yang terbaik untuk memutuskan prediksi *Random Forest*.

4.2 Analisis Hasil Pengujian

4.2.1 Skenario Pertama

Pada pengujian skenario pertama dengan rasio data *training* 90% dan data *testing* 10% delakukan pengujian terhadap metode SVM dan *Random Forest*. Pada skenario ini dilakukan 10 kali pencobaan setiap metode.

Tabel 4. Performansi matrik Support Vector Machine

	Support Vector Machine						
	Precision		Recall		F1-score		Accurasy
	0	1	0	1	0	1	
Pencobaan 1	96%	96%	96%	96%	96%	96%	96,23%
Pencobaan 2	96%	96%	96%	96%	96%	96%	96,23%
Pencobaan 3	97%	96%	96%	97%	97%	97%	96,73%
Pencobaan 4	97%	95%	96%	97%	97%	96%	96,48%
Pencobaan 5	97%	96%	96%	97%	97%	97%	96,73%
Pencobaan 6	96%	96%	96%	96%	96%	96%	95,97%
Pencobaan 7	96%	96%	96%	96%	96%	96%	96,23%
Pencobaan 8	96%	96%	96%	96%	96%	96%	96,23%
Pencobaan 9	97%	96%	96%	97%	97%	97%	96,73%
Pencobaan 10	97%	95%	95%	97%	96%	96%	95,97%

Tabel 5. Performansi matrik Random Forest

	Random Forest						
	Precision		Recall		F1-score		Accurasy
	0	1	0	1	0	1	
Pencobaan 1	98%	98%	98%	98%	98%	98%	98,49%
Pencobaan 2	99%	98%	98%	99%	98%	98%	98,49%
Pencobaan 3	100%	98%	98%	100%	99%	99%	98,99%
Pencobaan 4	100%	97%	98%	100%	99%	99%	98,74%
Pencobaan 5	99%	97%	98%	99%	99%	98%	98,49%
Pencobaan 6	99%	98%	98%	99%	99%	99%	98,74%
Pencobaan 7	100%	98%	98%	100%	99%	99%	98,99%
Pencobaan 8	99%	99%	99%	99%	99%	99%	98,99%
Pencobaan 9	99%	98%	98%	99%	99%	99%	98,74%
Pencobaan 10	99%	96%	97%	99%	98%	98%	97,73%

Pada tabel 4, menunjukkan hasil performansi matrik skenario pertama dengan menggunakan metode SVM. Hasil performansi tertinggi pada skenario ini mendapatkan hasil akurasi 96,73% , *precision* 97%, *recall* 97%, *f1-score* 97% di pencobaan 3,5 dan 9.

Pada tabel 5, menunjukkan hasil performansi matrik skenario pertama dengan menggunakan metode *Random Forest*. Hasil performansi tertinggi pada skenario ini mendapatkan hasil akurasi 98,99% ,*precision* 99%, *recall* 99%, *f1-score* 99% di pencobaan 3 dan 7.

Dalam pengujian ini pada skenario pertama ini mendapatkan hasil bahwa *Random Forest* lebih unggul dari metode SVM. Hasil Akurasi dari metode *Rendom Forest* unggul 2,26% dibandingkan dengan metode SVM. Jumlah nilai *precision* di *Rendom Forest* sangan baik dengan hasil 99% sedangkan pada metode SVM dengan hasil 97%. Jumlah nilai *recall* di *Random Forest* 99% sedangkan pada metode SVM 97% dan nilai *f1-score* di *Rendom Forest* 99% sedangkan pada metode SVM 97%.

4.2.2 Skenario Kedua

Pada pengujian skenario kedua dengan rasio data *training* 80% dan data *testing* 20% delakukan pengujian terhadap metode SVM dan *Random Forest*. Pada skenario ini dilakukan 10 kali pencobaan setiap metode.

. Tabel 6. Performansi matrik Support Vector Machine

	Support Vector Machine						
	Precision		Recall		F1-score		Accurasy
	0	1	0	1	0	1	
Pencobaan 1	95%	96%	96%	96%	96%	96%	95,97%
Pencobaan 2	95%	95%	95%	95%	95%	95%	95,47%
Pencobaan 3	96%	95%	96%	96%	96%	96%	95,97%
Pencobaan 4	96%	95%	96%	96%	96%	96%	95,97%
Pencobaan 5	94%	96%	96%	95%	95%	96%	95,47%
Pencobaan 6	96%	96%	96%	96%	96%	96%	96,23%
Pencobaan 7	96%	96%	96%	96%	96%	96%	95,97%
Pencobaan 8	97%	96%	96%	97%	96%	96%	96,48%
Pencobaan 9	95%	96%	96%	95%	96%	96%	95,72%
Pencobaan 10	96%	96%	96%	96%	96%	96%	96,23%

Tabel 7. Performansi matrik Random Forest

	Random Forest						
	Precision		Recall		F1-score		Accurasy
	0	1	0	1	0	1	
Pencobaan 1	98%	99%	99%	98%	99%	99%	98,74%
Pencobaan 2	99%	96%	97%	99%	98%	98%	97,73%
Pencobaan 3	100%	98%	98%	100%	99%	99%	98,99%
Pencobaan 4	98%	98%	98%	98%	98%	98%	98,24%

Pencobaan 5	98%	97%	97%	98%	98%	98%	97,73%
Pencobaan 6	99%	97%	98%	99%	98%	98%	98,24%
Pencobaan 7	99%	97%	97%	99%	98%	98%	97,93%
Pencobaan 8	99%	98%	98%	99%	99%	99%	98,74%
Pencobaan 9	97%	98%	98%	98%	98%	98%	97,98%
Pencobaan 10	98%	97%	97%	98%	97%	97%	97,48%

Pada tabel 6, menunjukkan hasil performansi matrik skenario kedua dengan menggunakan metode SVM. Hasil performansi tertinggi pada skenario ini mendapatkan hasil akurasi 96,48% ,*precision* 97%, *recall* 97%, *f1-score* 96% di pencobaan 8.

Pada tabel 7, menunjukkan hasil performansi matrik skenario kedua dengan menggunakan metode *Random Forest*. Hasil performansi tertinggi pada skenario ini mendapatkan hasil akurasi 98,99% , *precision* 100%, *recall* 100%, *f1-score* 99% di pencobaan 3.

Dalam pengujian ini pada skenario kedua ini mendapatkan hasil bahwa *Random Forest* lebih unggul dari metode SVM. Hasil Akurasi dari metode *Rendom Forest* unggul 2,51% dibandingkan dengan metode SVM. Jumlah nilai *precision* di *Rendom Forest* sangan baik dengan hasil 100% sedangkan pada metode SVM dengan hasil 97%. Jumlah nilai *recall* di *Random Forest* 100% sedangkan pada metode SVM 97% dan nilai *f1-score* di *Rendom Forest* 99% sedangkan pada metode SVM 97%.

4.2.3 Skenario Ketiga

Pada pengujian skenario Ketiga dengan rasio data *training* 70% dan data *testing* 30% delakukan pengujian terhadap metode SVM dan *Random Forest*. Pada skenario ini dilakukan 10 kali pencobaan setiap metode.

Tabel 8. Performansi matrik Support Vector Machine

	Support Vector Machine						
	Precision		Recall		F1-score		Accurasy
	0	1	0	1	0	1	
Pencobaan 1	96%	96%	96%	96%	96%	96%	96,23%
Pencobaan 2	95%	96%	96%	95%	96%	96%	95,72%
Pencobaan 3	96%	95%	96%	96%	96%	96%	95,97%
Pencobaan 4	96%	94%	95%	96%	96%	95%	95,47%
Pencobaan 5	97%	94%	94%	97%	96%	96%	95,72%
Pencobaan 6	96%	96%	96%	96%	96%	96%	96,23%
Pencobaan 7	97%	95%	96%	97%	97%	96%	96,48%
Pencobaan 8	94%	96%	96%	94%	95%	95%	95,22%
Pencobaan 9	94%	96%	96%	95%	95%	96%	95,47%
Pencobaan 10	96%	95%	95%	96%	96%	96%	95,72%

Tabel 9. Performansi matrik Random Forest

	Random Forest						
	Precision		Recall		F1-score		Accurasy
	0	1	0	1	0	1	
Pencobaan 1	98%	95%	96%	98%	97%	97%	96,98%
Pencobaan 2	98%	97%	97%	98%	98%	98%	97,73%
Pencobaan 3	99%	96%	96%	99%	98%	97%	97,48%
Pencobaan 4	98%	94%	95%	96%	96%	95%	95,47%
Pencobaan 5	98%	94%	94%	98%	96%	96%	96,23%
Pencobaan 6	98%	98%	98%	98%	98%	98%	98,24%
Pencobaan 7	99%	95%	95%	99%	97%	97%	97,23%
Pencobaan 8	96%	97%	97%	96%	97%	97%	96,73%
Pencobaan 9	97%	98%	98%	97%	97%	97%	97,48%
Pencobaan 10	96%	96%	96%	96%	96%	96%	96,48%

Pada tabel 8, menunjukkan hasil performansi matrik skenario ketiga dengan menggunakan metode SVM. Hasil performansi tertinggi pada skenario ini mendapatkan hasil akurasi 96,48% ,*precision* 97%, *recall* 97%, *f1-score* 97% di pencobaan 7.

Pada tabel 9, menunjukkan hasil performansi matrik skenario ketiga dengan menggunakan metode *Random Forest*. Hasil performansi tertinggi pada skenario ini mendapatkan hasil akurasi 98,24% , *precision* 98%, *recall* 98%, *f1-score* 98% di pencobaan 3 dan 9.

Dalam pengujian ini pada skenario pertama ini mendapatkan hasil bahwa *Random Forest* lebih unggul dari metode SVM. Hasil Akurasi dari metode *Rendom Forest* unggul 1,76% dibandingkan dengan metode SVM. Jumlah nilai *precision* di *Rendom Forest* dengan hasil 98% sedangkan pada metode SVM dengan hasil 97%. Jumlah nilai *recall* di *Random Forest* 98% sedangkan pada metode SVM 97% dan nilai *f1-score* di *Rendom Forest* 98% sedangkan pada metode SVM 97%. Jumlah nilai *precision , recall ,* dan *f1-score* pada metode SVM berbeda 1% dari metode *Random Forest*.

Dari ketiga skenario, Random Forest lebih diunggulkan dari metode SVM. Hal ini terjadi karena sifat dari Random Forest yang ensemble learning , yang menggunakan ataupun mambangun beberapa model tree. Dari 100 tree yang dibangun dipilih beberapa tree yang terbaik dan menghitung rata-rata hasil prediksi tree sehingga menghasilkan prediksi Random Forest yang tinggi.

4.3 Analisis Hasil Komparasi

Pada pengujian ini dilakukan komparasi dengan penelitian [5]. 4 metode melakukan komparasi dengan mengkomparasi hasil akurasi, *precision, recall* dan *f1-score*. Penelitian [5] menggunakan metode *machine learning* dengan rasio data 70% data *training* dan 30% data *testing*. Metode yang dikomparasi yaitu: SVM, KNN, NB, dan *Decision Tree*.

Tabel 10. Performansi matrik rasio data 70% : 30%

Metode Algoritma	*Precision*		*Recall*		*F1-score*		Akurasi
	0	1	0	1	0	1	
SVM	97%	93%	93%	97%	95%	95%	95,22%
KNN	96%	87%	88%	96%	92%	91%	91,70%
NB	94%	88%	89%	94%	92%	91%	91,45%
Decision Tree	99%	93%	94%	99%	96%	96%	96,23%

Penelitian [5], metode SVM hasil performansi matriks yang didapatkan akurasi 94%, *precision* 0 dan 1 mendapatkan hasil 93% dan 95%, *recall* 0 dan 1 mendapatkan hasil 93% dan 94%, *f1-score* 0 dan 1 mendapatkan hasil 94% dan 94%. Sedangkan pada penelitian ini mendapatkan hasil akurasi 95,22% lebih unggul 1,22%, *precision* 0 dan 1 mendapatkan hasil 97% dan 93% lebih unggul 3% pada *precision* 0, *recall* 0 dan 1 mendapatkan hasil 93% dan 97% lebih unggul 3% pada *recall* 1, *f1-score* 0 dan 1 mendapatkan hasil 95% dan 95% lebih unggul 1%.

Metode KNN hasil performansi matriks yang didapatkan pada penelitan [5], akurasi 94%, *precision* 0 dan 1 mendapatkan hasil 94% dan 95%, *recall* 0 dan 1 mendapatkan hasil 94% dan 95%, *f1-score* 0 dan 1 mendapatkan hasil 94% dan 94%. Sedangkan pada penelitian ini mendapatkan hasil akurasi, *precision, recall,* dan *f1-score* yang lebih rendah dari penelitian [5].

Metode NB hasil performansi matriks yang didapatkan pada penelitian [5] akurasi 90%, *precision* 0 dan 1 mendapatkan hasil 93% dan 88%, *recall* 0 dan 1 mendapatkan hasil 88% dan 92%, *f1-score* 0 dan 1 mendapatkan hasil 90% dan 90%. Sedangkan pada penelitian ini mendapatkan hasil akurasi, *precision, recall,* dan *f1-score* yang lebih unggul dari penelitian [5]. Pada penelitian ini akurasi mendapatkan hasil 91,45% lebih unggul 1,45%, *precision* 0 dan 1 mendapatkan hasil 94% dan 88% lebih unggul 1% pada *precision* 0, *recall* 0 dan 1 mendapatkan hasil 89% dan 94% lebih unggul 2% pada *recall* 1, *f1-score* 0 dan 1 mendapatkan hasil 92% dan 91% lebih unggul 2% pada *f1-score* 0.

Metode *Decision Tress* hasil performansi matriks yang didapatkan pada penelitian [5] akurasi 94%, *precision* 0 dan 1 mendapatkan hasil 93% dan 95%, *recall* 0 dan 1 mendapatkan hasil 95% dan 91%, *f1-score* 0 dan 1 mendapatkan hasil 94% dan 94%. Sedangkan pada penelitian ini mendapatkan hasil akurasi, *precision, recall,* dan *f1-score* yang lebih unggul dari penelitian [5]. Pada penelitian ini akurasi mendapatkan hasil 96,23% lebih unggul 2,23%, *precision* 0 dan 1 mendapatkan hasil 99% dan 93% lebih unggul 6% pada *precision* 0, *recall* 0 dan 1 mendapatkan hasil 84% dan 99% lebih unggul 8% pada *recall* 1, *f1-score* 0 dan 1 mendapatkan hasil 96% dan 96% lebih unggul 2%.

4.4 Aplikasi Android

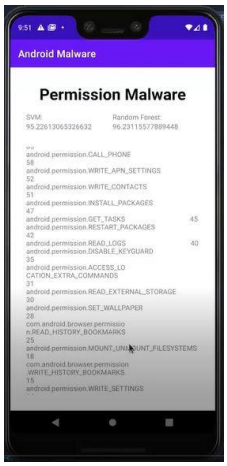
Pada tahap ini membuat aplikasi android menggunakan android studio. Aplikasi android ini melakukan deteksi malware menggunakan metode *Support Vector Machine* dan *Random Forest*.

4.4.1 Layar pertama

Gambar 3. Layar Pertama Aplikasi Android

Pada gambar 3 ini adalah layar pertama pada aplikasi deteksi malware ini, pada bagian ini ada *button scanning* untuk melakukan deteksi malware menggunakan metode *Support Vector Machine* (SVM) dan *Random Forest*.

4.4.2 Layar ke-dua



Gambar 4. Layar Kedua Aplikasi Android

Pada gambar 4 ini menampilkan layar kedua pada aplikasi deteksi malware. Layar kedua ini menampilkan hasil deteksi dari layar pertama. Pada layar ini menampilkan hasil deteksi menggunakan metode SVM 95,22% dan *Random Forest* 96,23%, serta menampilkan *permission* yang terdapat pada aplikasi yang terdeteksi malware.

5. Kesimpulan

Penelitian ini dilakukan berdasarkan tingginya serangan malware yang terjadi di Indonesia. Pada tahun 2019, terdeteksi 556.486 malware android di Indonesia, menjadikan Indonesia menduduki posisi pertama serangan malware terdeteksi terbanyak se-Asia Tenggara dan posisi keempat secara global. Penelitian ini melakukan deteksi malware dengan menggunakan pendekatan *machine learning* dalam proses klasifikasi.

Penelitian yang telah dilakukan mengusulkan perbandingan algoritma *support vector machine* (SVM) dan *Random Forest* yang digunakan dalam proses klasifikasi terhadap dataset malware yang bersumber dari web kaggle, serta membandingkan hasil akurasi dengan penelitian sebelumnya yang dilakukan oleh Lopez A et al [5], maka menghasilkan kesimpulan sebagai berikut:

1. Pada pengujian terhadap ketiga skenario yang ada didapatkan hasil bahwa metode *Random Forest* lebih unggul dibandingkan dengan metode SVM untuk kasus ini. Dengan hasil akurasi *Random Forest* sampai 98,99% sedangkan metode SVM mendapatkan hasil akurasi sampai 96,23%.
2. Pada *permission* dataset memiliki 330 daftar *permission* sebagai fitur yang memiliki tingkat keamanan perlindungan yang beragam. *Permission* pada aplikasi malware kebanyakan memiliki tingkat keamanan tertutup. Sebagai sampel 10 daftar *permission* di aplikasi malware 7 diantaranya memiliki tingkat keamanan tertutup, serta jumlah *permission* dan rata-rata *permission* pada aplikasi malware lebih besar dibandingkan aplikasi jinak.
3. Berdasarkan Hasil performansi matriks metode algoritma SVM, NB, dan *Decision Tree* yang digunakan di penelitian ini mendapatkan hasil yang lebih unggul dari penelitian yang dilakukan oleh Lopez A. et al [5], Pada penelitian ini menghasilkan akurasi yang lebih unggul 1,22% dengan SVM, 1,45% dengan NB dan 2,23% dengan *Decision Tree* dari penelitian sebelumnya. Pada penelitian yang dilakukan sebelumnya oleh Lopez A. et al mendapatkan hasil akurasi yang paling baik 94% dengan menggunakan metode SVM dan *Decision Tree* sedangkan NB 90%.

Saran untuk penelitian selanjutnya, Berdasarkan hasil pada penelitian ini, peneliti menyarankan metode *Random Forest* digunakan untuk melakukan klasifikasi data yang binary. Hal ini karena hasil akurasi dari metode *Random Forest* menghasilkan akurasi 98,99%. Serta peneliti juga menyarankan penelitian yang mendatang menggunakan dataset yang terbaru karena serangan ataupun fitur dari malware android selalu di update oleh orang-orang yang tidak bertanggung jawab.

Daftar Pustaka

[1] Egham, 2017,"Gerter Says Worldwide Sales of Smartphone Grew 9 Percent in First Quarter of 2017", https://www.gartner.com/newsroom/id/3725117 akses: Juli 2021

[2] Kementerian Komunikasi dan Informatika, 2018, "Indonesia Raksasa Teknologi Digital Asia", https://kominfo.go.id/content/detail/6095/indonesia-raksasa-teknologi-digital-asia/0/sorotan_media akse: Juli 2021

[3] Startcounter Global Stats, 2018, "Mobile Operating System Market Share Indonesia", https://gs.statcounter.com/os-market-share/mobile/indonesia/2018 , akses: Juli 2021

[4] Imantoko Kurniadi, 2021, "Serangan Mobile Malware terdeteksi Menurun di Indonesia" https://selular.id/2021/04/serangan-mobile-malware-terdeteksi-menurun-di-indonesia/ , akses: Juli 2021

[5] C. C. U. Lopez, A. N. Cadavid, " Framework for malware analysis in Android",.i2t Research Group , Icesi University, Cali, Colombia, 2016.

[6] D. O. Sahin, O. E. Kural, S. Akleylek, E. Kilic, "New Result on Permission Based Static Analysis For Android Malware", Department of Computer Engineering , Ondokuz Mayis University, 2018.

[7] T. Ban, T. Takahashi, S. Guo, D. Inoue, K Nakao, "Integration of Multi-modal Features For Android Malware Detection Using Linear SVM", Information and Communication Tachnology, Koganei, Tokyo, 2016.

[8] H. J. Zhu, T.H. Jiang, Bo Ma, Z.H. You, W.L. Shi, Li Cheng, "HEMD: a highly efficient random forest based malware detection framework for Android", The Xianjiang Technical Institute of Pyisics and Chemistry, Chinese Academy of Sciences, China, 2016.

[9] Boser B., Guyon I., Vapnik V., "Proceedings of the fifth annual workshop on Computational learning theory", COLT, 1992

[10] Ho Tin Kam, "Random Decision Forests", AT&T Bell Laboratories 600 Moutain Avenue, Murray Hill, USA , 1995

[11] https://www.kaggle.com/xwolf12/datasetandroidpermissions akses: Juni 2020

[12] Leo Breiman, "Random Forests", Statis Department, University of California, Berkeley, 2001.

[13] Junmei S., Kai Y., X Liu , Chunlei Y., Yaoyin F., "Malware Detection on Android Smarthone using Keywords Vector and SVM ", Hangzhou Institute of Sorvice Engineering, Hangzhou Normal University, China, 2017.

[14] Yuki T., Kazuya S., Satoshi F., "Detecting Ransomware using Support Vector Machines", Department of Electrical Engineering and Computer Science, Tokyo Metropolita University, Japan, 2018.

[15] Nikola M., Ali D., Kim-Kwang R. C., "Machine learning aided Android malware classification", School of Computer Science, University of Manchester, UK, 2017.

[16] Alif P. B.A, Yudha P., M. Faris R., "Deteksi *Malware* Menggunakan Metode Random Forest Berdasarkan Analisa Forensik", Prodi S1 Teknik Komputer, Universitas Telkom, 2021.

[17] Ary Adhigana S., Parma S., Erwid M. J., "Analisis Metode Ensemble untuk Mendeteksi Malware pada *Mobile Devices*". Fakultas Informatika, Universitas Telkom, 2019.

[18] Mahendra D. P., M. Irvan T., Adnan I. H., Adiwijaya, "Perbandingan Support Vector Machine dan Modified Balanced Random Forest dalam Deteksi Penyakit Diabetes", Rekayasa Sistem dan Teknologi Infomasi, Universitas Telkom, 2019.

[19] https://developer.android.com/reference/android/Manifest.permission

[20] http://androidpermissions.com/