CyberSecurity dan Forensik Digital e-ISSN: 2615-8442
Vol. 5, No. 1, Mei 2022, hlm. 1-5

ANALISIS STATIS DETEKSI *MALWARE* ANDROID MENGGUNAKAN ALGORITMA SUPERVISED MACHINE LEARNING

Raden Budiarto Hadiprakoso[1], Wahyu Rendra Aditya [2], Febriora Nevia Pramitha [3]

[1,2,3] Politeknik Siber dan Sandi Negara
Email: [1]raden.budiarto@poltekssn.ac.id, [2]wahyu.rendra@poltekssn.ac.id, [3]febriora.nevia@poltekssn.ac.id

Abstrak

Saat ini, pertumbuhan sistem operasi Android di perangkat *smartphone* sedang populer. Bagaimana pun, dibalik popularitas tersebut platform Android juga menjadi target peluang kejahatan dunia maya terhadap ancaman keamanan siber seperti *malware*. Mengidentifikasi *malware* ini sangat penting untuk menjaga keamanan dan privasi pengguna. Karena proses identifikasi *malware* yang semakin rumit, maka perlu digunakan machine learning untuk klasifikasi *malware*. Penelitian ini mengumpulkan fitur analisis statis dari aplikasi aman dan berbahaya. (*malware*). Dataset yang digunakan pada penelitian adalah dataset *malware* DREBIN yang merupakan dataset *malware* yang tersedia secara publik. Dataset tersebut terdiri dari fitur API CALL, system command, manifest permission, dan Intent. Data tersebut kemudian diproses menggunakan berbagai algoritma supervised machine learning di antaranya Support Vector Machine (SVM), Naive Bayes, Decision Tree dan K-Nearest Neighbors. Kami juga berkonsentrasi pada memaksimalkan pencapaian dengan mengevaluasi berbagai algoritma dan menyesuaikan beberapa konfigurasi untuk mendapatkan kombinasi terbaik dari hyper-parameter. Hasil eksperimen menunjukkan bahwa klasifikasi model SVM mendapatkan hasil terbaik dengan mencapai akurasi 96,94% dan nilai AUC (Area Under Curve) 95%.

Kata kunci: android, *malware*, machine learning, deteksi *malware*, analisis statis

STATIC ANALYSIS FOR ANDROID MALWARE DETECTION USING SUPERVISED MACHINE LEARNING ALGORITHM

Abstract

Currently, the growth of the Android operating system on smartphone devices is popular. However, behind this popularity, the Android platform is also a potential target for cybercrimes against cybersecurity threats such as malware. Identifying this malware is critical to maintaining user security and privacy. Because the malware identification process is getting more complicated, it is necessary to use machine learning for malware classification. This study collects the static analysis features of safe and malicious applications. (malware). The dataset used in this study is a DREBIN malware dataset which is a publicly available malware dataset. The dataset consists of the CALL API features, system commands, manifest permissions, and Intents. The data is then processed using various supervised machine learning algorithms including Support Vector Machine (SVM), Naive Bayes, Decision Tree and K-Nearest Neighbors. We also concentrate on maximizing performance by evaluating various algorithms and adjusting some configurations to get the best combination of hyper-parameters. The experimental results show that the SVM model classification gets the best results by achieving an accuracy of 96.94% and an AUC (Area Under Curve) value of 95%.

***Keywords**: android, malware, machine learning, malware detection, static analysis*

1. PENDAHULUAN

Perkembangan teknologi informasi saat ini telah membawa banyak kemajuan di berbagai bidang, salah satunya pada perangkat bergerak yaitu *smartphone*. Sebuah ponsel pintar pada dasarnya menyediakan aplikasi dan sistem operasi untuk memfasilitasi penggunanya. Beberapa sistem operasi yang digunakan antara lain iOS, Blackberry OS, dan Android. Dari beberapa sistem operasi yang ada, Android merupakan salah satu sistem operasi yang paling banyak digunakan. Menurut CEO Google, Erich Schmidt, sekitar 1,4 juta Android diaktifkan setiap hari (cycles, 2021).

Pada pertengahan 2020, platform Android mewakili 70,61% dari total pasar sistem operasi seluler berdasarkan NetMarketShare, yang menjadikannya sistem operasi seluler yang paling banyak digunakan (Stat, 2021). Namun, popularitas yang dimiliki Android tidak selalu berdampak positif bagi penggunanya. Bahkan dengan berbagai mekanisme perlindungan seperti Play Store Protection, masih ada berbagai laporan keberadaan *malware* di Play Store (Alzaylaee, Yerima & Sezer, 2020)..

Sistem operasi seluler Android yang banyak digunakan ini dapat menjadi sasaran kejahatan. Menurut McAfee Mobile Threat Report, terdapat *malware* tipe

baru bernama PhantomLance, LeifAccess atau Shopper, yang baru teridentifikasi pada Mei 2020 dan sudah aktif secara global, terutama di Amerika Serikat dan Brazil dengan total 943 dan 286 serangan . Menurut Pavel Shoshin, PhantomLance adalah Trojan *backdoor* untuk Android di Google Play (Qiu et al., 2020). Pernyataan tersebut menunjukkan bahwa bahkan perusahaan seperti Google, yang sering berada di garis depan dalam menjaga ekosistem Android bebas dari *malware*, telah gagal mengambil tindakan untuk mencegah *malware* (Ma et al., 2019). Dari pernyataan di atas, *malware* Android semakin canggih untuk mengelak dari keamanan dan menyusup ke Google Play Store. *Malware* ini kemungkinan besar akan menyebar ke perangkat Android. Karena kecanggihan *malware* semakin meningkat dan sulit dideteksi, maka penting dilakukan deteksi *malware*. Banyak peneliti berusaha untuk mengurangi serangan siber pada *malware* melalui berbagai pendekatan (Odusami et al., 2018).

Berbagai teknik deteksi *malware* dikembangkan untuk melawan ekspansi ini. Di antara sistem deteksi yang paling umum adalah analisis statis (Taheri et al., 2020). Melalui pendekatan analisis statis, *malware* dikenali sebagai *signature* yang diekstrak dari sebuah aplikasi. Biasanya, *signature* ini berasal dari *payload*. Ketika *signature* telah dikenali, maka akan disimpan dalam basis data dan digunakan untuk mengklasifikasikan kasus *malware* yang baru berdasarkan basis data tersebut. (McLaughlin et al., 2017).

Pada sisi lain analisis dinamis adalah prosedur di mana *instance malware* diperiksa secara *real-time*. Saat berjalan di lingkungan virtual seperti emulator atau *sandbox*, aktivitas program yang berbahaya diamati. Program anti-virus mencoba menemukan apakah tindakan seperti replikasi file, duplikasi, atau bahkan penyamaran dilakukan. Setelah kode didekompilasi, perbandingan dibuat dengan kode *malware* yang diketahui untuk memastikan apakah aplikasi tersebut berbahaya. Kelemahan utama metode ini adalah ketidakmampuan untuk menemukan virus yang menggunakan prosedur baru untuk menjalankan aktivitas berbahaya serta tidak efisien dalam hal waktu dan sumber daya (Wang, Zhao and Wang, 2019).

Untuk penelitian ini, kami telah memfokuskan pada analisis statis untuk klasifikasi *malware*. Analisis statis adalah metode yang cepat dan mudah untuk mengidentifikasi malware tanpa menjalankan aplikasi atau mengamati perilaku run-time (Ma et al., 2019). Penelitian ini mengusulkan algoritma *machine laerning* untuk digunakan dalam deteksi *malware* dengan membandingkan beberapa algoritma. Empat algoritma akan dibandingkan, yaitu Support Vector Machine (SVM), Naive Bayes, Decision Tree, K-Nearest Neighbor (KNN).

2. TINJAUAN PUSTAKA

Dalam beberapa tahun, algoritma *Machine Learning* (ML) bertugas mengembangkan sistem cerdas dengan cara melatih mesin untuk membuat keputusan. Dengan *dataset* sebagai input dan *classifier* sebagai metodenya, ML mampu mengidentifikasi data baru yang memiliki kemiripan. Ada banyak algoritma ML yang dapat digunakan untuk membangun *framework* atau model ML dan masing-masing algoritma memiliki kelebihan masing-masing tergantung *dataset* dan fitur yang digunakan.

(Wang, Zhao and Wang, 2019) pada penelitiannya, membandingkan ML *classifier* yang termasuk ke dalam kelompok *Supervised. Supervised Machine Learning* merupakan pencarian algoritma yang memanfaatkan instans eksternal untuk menghasilkan hipotesis umum, yang kemudian digunakan untuk memprediksi kejadian masa depan Algoritma ML yang digunakan pada penelitian ini antara lain Decision Table, Random Forest (RF), Naïve Bayes (NB), Support Vector Machine (SVM), JRip, dan Decission Tree (J48) dan menggunakan *tool machine learning Waikato Environment for Knowledge Analysis* (WEKA). Hasilnya adalah SVM menjadi algoritma yang memiliki akurasi dan presisi paling baik.

Pada tahun 2018 (Fan et al., 2018), melakukan penelitian lain mengenai *malware detection*. (McLaughlin et al., 2017)Fan membangun *malware detection framework* yang di dalamnya mengemas enam *machine learning classifiers*, SVM, Decision Tree (C4.5), MLP, NB, K-KN dan Bagging predictor. Untuk menghitung performa ML algoritma tersebut, Chen membagi menjadi dua kategori yaitu kategori pertama yang berisikan fitur *permission* dan *sensitive* API *calls* dan kategori kedua yang berisikan fitur *permission, sensitive, API calls, sequence*, dan *dynamic behaviour*. Pada kategori pertama performa terbaik diperoleh oleh K-NN dan MLP dengan rata-rata akurasi mencapai di atas 91.00%. Sedangkan pada kategori kedua, SVM dan K-NN memperoleh akurasi tertinggi dengan nilai rata-rata 93,80% dan 93,80%. Dari segi eksekusi waktu semua algoritma berjalan di bawah satu menit, kecuali (MLP) yang dengan multi-layer-nya memakan waktu lebih lama. Secara umum pada penelitian ini K-NN merupakan *classifier* dengan performa terbaik yang membutuhkan sumber komputasi yang kecil.

Pada tahun 2020, dilakukan penelitian oleh (Zhang et al., 2020) deteksi *malware* yang berbasis pada *machine learning* menggunakan analisis dinamis pada aplikasi android. Pada akhirnya, dibuat sebuah aplikasi, untuk mengekstrak informasi-informasi yang terdapat pada suatu aplikasi android. Pada aplikasi tersebut dilatih beberapa algoritma klasifikasi untuk melihat performa terbaik pada akurasi dan kecepatannya. Pada aplikasi tersebut didapatkan bahwa penelitian tersebut mendapatkan nilai akurasi sebesar 97% untuk mendeteksi *malware* tak terlihat dari data yang telah disiapkan.

Pada penelitian yang dilakukan oleh (Feng et al., 2018), melakukan studi analisis dinamis yang mana menggunakan syscall-capture untuk menangkap dan menganalisis perilaku jejak system-call yang dibuat oleh setiap aplikasi selama menjalankan aplikasi tersebut. Pada akhirnya, didapatkan tingkat akurasi sebesar 85% dengan menggunakan algoritma *decission tree* dan tingkat akurasi 88% dengan menggunakan algoritma RF.

Pada penelitian yang dilakukan oleh (Lashkari et al., 2018), dilakukan deteksi *malware* pada *smartphone* yang menggunakan sistem operasi android. Penelitian ini mencakup pembahasan mengenai bagaimana mengembangkan sistem deteksi *malware* yang dapat mendeteksi berbagai jenis *malware*, bagaimana mendeteksi *malware* sebelum proses instalasi melalui beberapa tahap. Tahap pertama yaitu mengekstrak string dari aplikasi android dan mengeksplorasi file android manifest.xml. tahap kedua yaitu memisahkan kata kunci string dari android manifest.xml. tahap ketiga yaitu mengklasifikasikan *malware* dan aplikasi sah dengan menggunakan kata kunci dan string sebagai fitur input. Tahap terakhir yaitu mengidentifikasi aplikasi berbahaya dan aplikasi aman. Hasil dari penelitian ini didapatkan bahwa SVM merupakan algoritma paling cocok dengan *binary* dibandingkan dengan algoritma lain dan model yang diusulkan meningkatkan akurasi hingga 85,51%.

Secara keseluruhan yang membedakan penelitian ini dengan penelitian sebelumnya adalah kami menggunakan pendekatan analisis statis. Pendekatan ini dipilih karena proses deteksi yang cepat serta lebih ringan pada memori mengingat lingkungan implementasi pada *mobile*.

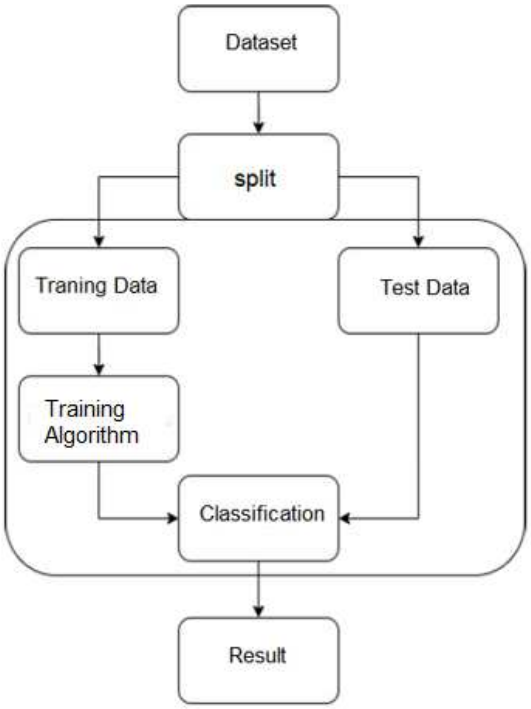
3. METODE PENELITIAN

Metode penelitian ini dibagi menjadi beberapa tahapan seperti yang digambarkan pada Gambar 1 di antaranya adalah pengumpulan data penelitian, pengolahan data, pengujian klasifikasi, dan perbandingan algoritma. Berikut ini adalah gambaran dari masing-masing tahapan tersebut.

Tahap pertama adalah pengumpulan data. Kami mengumpulkan kumpulan data dari proyek DREBIN *malware* (Arp et al., 2014). Data tersebut berisi 215 atribut fitur yang diekstrak dari 3799 aplikasi (1260 aplikasi *malware* dan 2539 aplikasi jinak). Atribut fitur pada dataset terdiri dari atribut analisis statis yang berupa *signature* dari aplikasi. Dataset berisi atribut statis seperti izin manifes, tanda tangan perintah, maksud. Selain itu juga terdapat atribut API call yang merupakan hasil analisis dinamik. Dataset ini akan digunakan sebagai data latih dan data uji untuk algoritma pembelajaran mesin.

Tahap kedua adalah pengolahan data. Dataset yang ada kemudian dibagi untuk data latih dan data uji. Proporsi data latih dan data uji adalah 80% untuk data latih atau 3039 data dan 20% untuk data validasi atau sebanyak 760 data. Satu kolom mewakili satu fitur dan frekuensinya, sedangkan satu baris mewakili satu aplikasi android, dan kolom terakhir adalah jenis kelas (1 untuk aplikasi *malware* dan 0 untuk aplikasi jinak). Data dibagi dengan cara *cross validation*, di mana data dibagi merata untuk pelatihan dan pengujian tanpa saling tumpang tindih. Pada penelitian ini kami menggunakan 5-fold cross validation.

Tahap ketiga adalah pengujian klasifikasi. Dalam penelitian ini, data akan digunakan algoritma SVM, NB, Decision Tree, dan K-NN. Setiap algoritma akan menggunakan data latih dengan proporsi yang sama yaitu 80% untuk data latih dan 20% untuk data uji.



Gambar 1. Alur penelitian

Berikut adalah gambaran proses pada tahap ini:

1. Data latih diperoleh dari pemisahan acak dari dataset yang diambil 80% dari seluruh dataset.
2. Data uji pemisahan acak diperoleh dari dataset yang diambil 20% dari seluruh dataset.
3. Pelatihan algoritma adalah proses untuk melatih algoritma klasifikasi dengan data pelatihan.
4. Klasifikasi pada tahap ini dilakukan pengujian antara data latih dan data uji, dengan menghitung akurasi klasifikasi. Hasilnya adalah tingkat akurasi dalam mengklasifikasikan aplikasi.

Langkah selanjutnya adalah perbandingan algoritma. Hasil dari tahap ketiga adalah akurasi dalam memprediksi aplikasi, termasuk *malware* atau tidak. Tingkat akurasi dari algoritma SVM, NB, Decision Tree, dan K-NN akan dibandingkan. Dari perbandingan tersebut didapatkan algoritma yang dinilai lebih efektif dalam mengklasifikasikan aplikasi, baik itu *malware* maupun bukan.

4. HASIL DAN PEMBAHASAN

Setelah data terkumpul, kami menggunakan Google Colab untuk memprosesnya. Kami menggunakan lingkungan pengujian dengan perangkat prosesor i3-2328M dengan RAM 10GB: Pada tahap *preprocessing* data dengan teknik SMOTE didapatkan jumlah kelas sampel yang sama, yaitu 2539 di setiap kelas. Validasi silang 5 kali lipat bertingkat membagi data secara merata dengan 80% data pelatihan, dan sisanya untuk validasi. Teknik yang digunakan untuk pelatihan dan validasi adalah SVM, K-NN, Decision Tree dan NB. Tabel 1 menjelaskan hasil akurasi dan latensi pelatihan rata-rata.

Tabel 1. Hasil pengujian

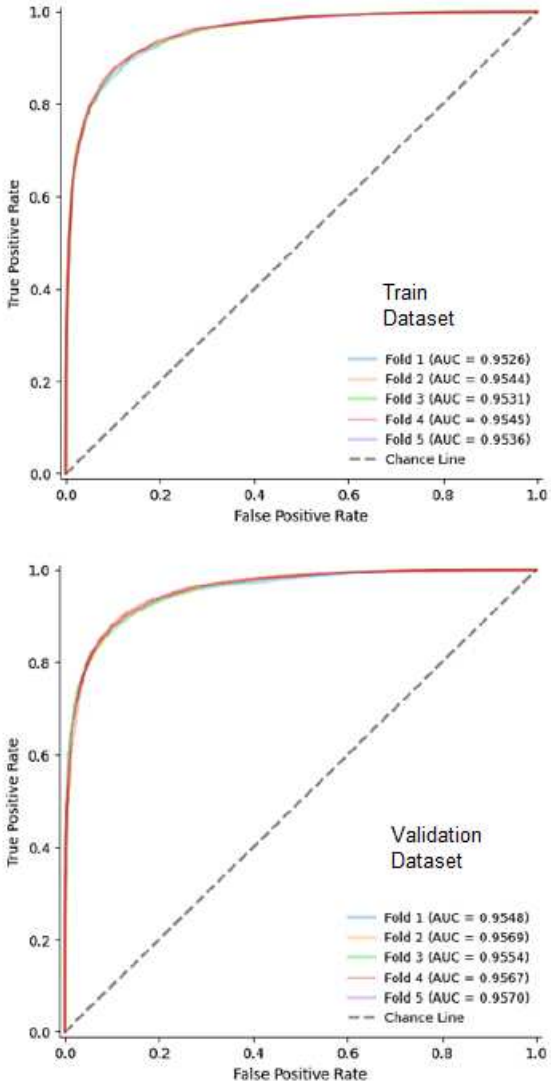
Algoritma	Akurasi	Latensi
SVM	0.969471	4.497
Naive Bayes	0.738667	1.247
Decision Tree	0.923684	2.754
K-NN	0.928667	1.549

Dari tabel akurasi di atas, diketahui bahwa algoritma SVM memiliki rata-rata akurasi 96,94 % yang juga merupakan nilai tertinggi dibandingkan ketiga algoritma lainnya. Selain itu, algoritma KNN juga memiliki tingkat akurasi rata-rata yang mendekati akurasi rata-rata algoritma SVM.

SVM menunjukkan hasil deteksi *malware* terbaik. SVM menunjukkan hasil terbaik di sini agaknya karena datasetnya tidak terlalu besar, dan datanya bersih dari *noise*. SVM cenderung bekerja lebih baik pada data yang tidak terlalu besar dan bebas *noise*. Faktor lainnya adalah ukuran fitur dari dataset yang cukup besar dengan 215 fitur karena SVM lebih efektif untuk dataset dengan dimensi yang besar.

Algoritma NB tidak menghasilkan skor akurasi yang tinggi karena cenderung bekerja lebih efektif pada data pelatihan yang lebih sedikit. Algoritma ini juga memiliki waktu latensi terpendek karena set data pelatihan hanya disimpan dalam memori dan digunakan kembali selama prediksi. Ini menghasilkan waktu pelatihan yang singkat tetapi waktu pengujian lebih lama. Cara kerja yang sama juga berlaku untuk algoritma K-NN. Oleh karena itu, algoritma ini memiliki waktu pelatihan yang relatif singkat. Algoritma K-NN menunjukkan hasil yang cukup baik, hal ini disebabkan karena sebaran data non linier karena algoritma ini terkenal dengan klasifikasi data non linier.

Hasil pengujian berikutnya ditunjukkan pada gambar 2 yang merupakan kurva ROC (*Receiver Operating Characteristics*) dari model SVM. Kurva ROC digunakan untuk menilai kinerja masalah klasifikasi. *Area Under Curve* (AUC) adalah ukuran daerah di bawah kurva ROC, semakin luas area ini menunjukkan model klasifikasi yang diusulkan semakin baik. ROC merupakan representasi grafis dari hubungan antara sensitivitas dan spesifisitas. Nilai ini menunjukkan seberapa baik kemampuan model untuk memisahkan kelas. Gambar 2 menggambarkan AUC yang cukup lebar sekitar 95%. Nilai tersebut menunjukkan bahwa model yang dihasilkan sesuai untuk kategorisasi input. Selain itu pada gambar 2 juga menunjukkan luas AUC pada *dataset* pelatihan dan validasi relatif stabil. Hasil ini menunjukkan bahwa model yang diusulkan sudah fit.



Gambar 2. Kurva ROC pada model SVM

5. KESIMPULAN DAN SARAN

Popularitas sistem operasi Android telah menjadikannya target banyak kejahatan, seperti *malware*. Berbagai mesin pendeteksi *malware* mulai bermunculan dengan berbagai metode deteksi statis atau dinamis dan algoritma pembelajaran mesin. Dalam studi ini, deteksi *malware* menggunakan metode analisis statis dan algoritma *machine learning*. Hasil yang ditampilkan membuktikan bahwa model kami memberikan pencapaian akurasi yang tinggi yakni 96,94% dengan menggunakan algoritma SVM. Hasil pengujian kurva ROC juga menunjukkan bahwa model memiliki luas AUC di kisaran 95%. Untuk pengembangan penelitian lebih lanjut, algoritma *deep learning* dapat diselidiki untuk meningkatkan kemampuan deteksi *malware* pada platform Android.

DAFTAR PUSTAKA

Alzaylaee, M.K., Yerima, S.Y. and Sezer, S., 2020. DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, 89, p.101663.

Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H. and Rieck, K., 2014. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. Symposium on Network and Distributed System Security (NDSS). . https://doi.org/10.14722/ndss.2014.23247.

Cycles, D., 2021. Topic: Android. [online] Statista. Available at: <https://www.statista.com/topics/876/android/> [Accessed 11 Oct. 2021].

Fan, M., Liu, J., Luo, X., Chen, K., Tian, Z., Zheng, Q. and Liu, T., 2018. Android malware familial classification and representative sample selection via frequent subgraph analysis. IEEE Transactions on Information Forensics and Security, 13(8), pp.1890–1905.

Feng, P., Ma, J., Sun, C., Xu, X. and Ma, Y., 2018. A novel dynamic Android malware detection system with ensemble learning. IEEE Access, 6, pp.30996–31011.

Lashkari, A.H., Kadir, A.F.A., Taheri, L. and Ghorbani, A.A., 2018. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In: 2018 International Carnahan Conference on Security Technology (ICCST). IEEE.pp.1–7.

Ma, Z., Ge, H., Liu, Y., Zhao, M. and Ma, J., 2019. A combination method for android malware detection based on control flow graphs and machine learning algorithms. IEEE access, 7, pp.21235–21245.

McLaughlin, N., Martinez del Rincon, J., Kang, B., Yerima, S., Miller, P., Sezer, S., Safaei, Y., Trickel, E., Zhao, Z. and Doupé, A., 2017. Deep android malware detection. In: Proceedings of the seventh ACM on conference on data and application security and privacy. pp.301–308.

Odusami, M., Abayomi-Alli, O., Misra, S., Shobayo, O., Damasevicius, R. and Maskeliunas, R., 2018. Android malware detection: A survey. In: International conference on applied informatics. Springer.pp.255–266.

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S. and Xiang, Y., 2020. A survey of Android malware detection with deep neural models. ACM Computing Surveys (CSUR), 53(6), pp.1–36.

Taheri, R., Ghahramani, M., Javidan, R., Shojafar, M., Pooranian, Z. and Conti, M., 2020. Similarity-based Android malware detection using Hamming distance of static binary features. Future Generation Computer Systems, 105, pp.230–247.

Wang, W., Zhao, M. and Wang, J., 2019. Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. Journal of Ambient Intelligence and Humanized Computing, 10(8), pp.3035–3043.

Zhang, X., Zhang, Y., Zhong, M., Ding, D., Cao, Y., Zhang, Y., Zhang, M. and Yang, M., 2020. Enhancing state-of-the-art classifiers with API semantics to detect evolved android malware. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp.757–770.

.