open access
Published online on: https://journal.uii.ac.id/jurnalsnati/
Check for updates
DOI: 10.20885/snati.v4.i1.1
JURNAL SNATI Sains, Nalar, dan Aplikasi Teknologi Informasi
Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi
Vol. 4 No. 1 (2025) 1 - 8 ISSN Media Electronic: 2807-5935

# Analisis Perilaku Malware Menggunakan Pendekatan Analisis Statis dan Dinamis

*Malware Behavior Analysis Using Static and Dynamic Analysis Approaches*

Khansa Khalda[1], Dwi Kurnia Wibowo[2]
[1,2]Jurusan Informatika, Fakultas Teknik, Universitas Jenderal Soedirman, Banyumas, Indonesia
[1]khansa.khalda@mhs.unsoed.ac.id, [2]dwi.kurnia@unsoed.ac.id

***Abstract***

*Malware detection is a critical challenge in the evolving landscape of cybersecurity. This study explores the integration of static and dynamic analysis to enhance malware detection accuracy. Static analysis examines malware files without execution, providing insights into their metadata and structural attributes, while dynamic analysis observes behavior during execution in controlled environments. Using a dataset of 5000 samples, including ransomware, trojans, spyware, and worms, tools like IDA Pro, PE Studio, and sandbox platforms were employed. Results indicate that 87% of malware samples utilize code obfuscation to evade detection, and 95% exhibit suspicious runtime activities, such as registry modifications and encrypted network communications. A machine learning model (Deep Neural Networks, Random Forest, Support Vector Machine) trained on hybrid datasets achieved 97.8% accuracy with DNN, demonstrating superiority over single-method approaches. Challenges like high computational demands were addressed through cloud-based implementations.*

*Keywords: dynamic analysis; hybrid analysis; machine learning; malware detection; static analysis.*

**Abstrak**

Deteksi *malware* merupakan tantangan krusial dalam perkembangan keamanan siber. Penelitian ini mengeksplorasi integrasi analisis statis dan dinamis untuk meningkatkan akurasi deteksi *malware*. Analisis statis meneliti file *malware* tanpa eksekusi, memberikan wawasan tentang *metadata* dan atribut strukturalnya, sedangkan analisis dinamis mengamati perilaku *malware* selama eksekusi di lingkungan terkendali. Menggunakan *dataset* 5000 sampel, termasuk *ransomware*, *trojan*, *spyware*, dan *worm*, alat seperti IDA Pro, PE Studio, dan *platform sandbox* digunakan. Hasil menunjukkan 87% sampel *malware* menggunakan *code obfuscation* untuk menghindari deteksi, dan 95% menunjukkan aktivitas *runtime* mencurigakan, seperti modifikasi *registry* dan komunikasi jaringan terenkripsi. Model pembelajaran mesin (*Deep Neural Networks, Random Forest, Support Vector Machine*) yang dilatih pada *dataset hybrid* mencapai akurasi 96,4% dengan DNN, menunjukkan keunggulan dibandingkan pendekatan metode tunggal. Tantangan seperti kebutuhan komputasi tinggi diatasi melalui implementasi berbasis *cloud*.

Kata kunci: analisis dinamis; analisis *hybrid*; analisis statis; deteksi *malware*; pembelajaran mesin.

## 1. Pendahuluan

Keamanan dunia maya menjadi isu yang sangat penting di era digital, terutama dengan meningkatnya ancaman *malware* (*malicious software*). *Malware* adalah perangkat lunak berbahaya yang dirancang untuk mencuri informasi, merusak data, atau mengganggu operasional sistem tanpa izin pemiliknya. Jenis *malware* seperti *ransomware*, *spyware*, *trojan*, dan *worms* memiliki dampak signifikan terhadap keamanan. Selain itu, *malware* modern menggunakan teknik canggih seperti *code obfuscation*, *encryption*, dan *polymorphism* untuk menghindari deteksi. *Code obfuscation* digunakan untuk menyamarkan kode *malware* agar sulit dianalisis, sementara *encryption* mengenkripsi komunikasi antara *malware* dan *server Command and Control* (C&C), sehingga aktivitas mereka sulit dilacak. Teknik *polymorphism* dan *metamorphism* memungkinkan *malware* mengubah karakteristiknya setiap kali dijalankan, membuat pendekatan deteksi berbasis tanda tangan menjadi kurang efektif. Ancaman ini menyoroti urgensi untuk mengembangkan metode deteksi *malware* yang lebih adaptif dalam menghadapi kompleksitas ancaman siber [1], [2], [3], [4].

Ancaman *malware* tidak hanya terbatas pada perangkat seluler, tetapi juga mencakup infrastruktur kritis yang menjadi tulang punggung operasional suatu negara. Infrastruktur vital seperti jaringan listrik, sistem transportasi, dan layanan kesehatan telah menjadi target utama serangan siber karena

Received: 26-1-2024 | Revised: 31-12-2024 | Accepted: 07-01-2025 | Published: 15-01-2025

dampaknya yang luas dan signifikan. Serangan terhadap sektor ini tidak hanya dapat mengganggu layanan penting, tetapi juga berpotensi membahayakan nyawa manusia, terutama jika sistem yang diserang mengelola operasi medis, distribusi energi, atau kontrol lalu lintas.

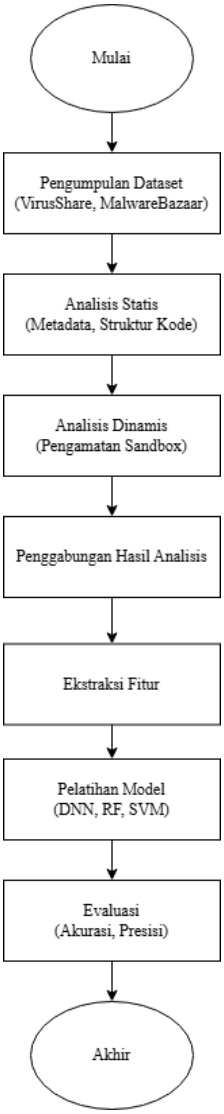
Deteksi *malware* dapat dilakukan melalui analisis statis dan dinamis, yang saling melengkapi. Analisis statis memeriksa file tanpa mengeksekusi kode, menggunakan alat seperti *disassembler* dan *decompiler* untuk mengidentifikasi struktur internal dengan cepat, namun lemah terhadap teknik penyamaran seperti *code obfuscation*. Sementara itu, analisis dinamis mengamati perilaku *malware* saat dijalankan dalam lingkungan terkendali, memberikan wawasan mendalam tentang aktivitas *runtime* seperti komunikasi jaringan dan modifikasi sistem, tetapi membutuhkan sumber daya besar dan rentan terhadap teknik *anti-sandboxing*. Untuk mengatasi kelemahan masing-masing, integrasi kedua pendekatan ini dalam metode *hybrid* diperlukan agar deteksi *malware* lebih komprehensif dan adaptif [2], [3], [4].

Pendekatan *hybrid*, yang menggabungkan analisis statis dan dinamis, telah terbukti memberikan hasil yang lebih komprehensif dalam mendeteksi *malware*. Metode ini tidak hanya mampu mengidentifikasi karakteristik malware yang tersembunyi tetapi juga menganalisis perilaku *runtime*-nya, sehingga memberikan pandangan menyeluruh tentang ancaman yang dihadapi. Beberapa penelitian mendukung efektivitas pendekatan ini. Misalnya, implementasi teknologi *deep learning* dengan arsitektur *Long Short-Term Memory* (LSTM) mencapai tingkat akurasi hingga 98,7% dalam mendeteksi *malware* berbasis Android. Selain itu, penggunaan *Indicator of Compromise* (IOC), seperti *hash file*, alamat IP, dan domain mencurigakan, telah menjadi alat penting untuk mengidentifikasi sistem yang terinfeksi dan memperkuat sistem *Cyber Threat Intelligence* (CTI). Dalam lanskap ancaman siber yang terus berkembang, pendekatan *hybrid* menjadi solusi penting untuk meningkatkan efektivitas dan efisiensi deteksi *malware* [2], [3], [4], [5], [6].

Penelitian ini mengintegrasikan analisis statis dan dinamis dalam pendekatan *hybrid* untuk memberikan pemahaman menyeluruh tentang karakteristik *malware* sekaligus mengatasi kelemahan masing-masing metode. Analisis statis mengevaluasi struktur internal *malware* tanpa menjalankannya, sementara analisis dinamis mengamati perilaku *malware* dalam lingkungan terkendali. Pendekatan ini ditingkatkan dengan teknologi *machine learning*, yang mampu meningkatkan efisiensi dan akurasi deteksi menggunakan *dataset* perilaku *malware*. Hasilnya, model ini diharapkan menjadi solusi adaptif dan efisien untuk mendeteksi *malware* yang semakin kompleks, serta memberikan kontribusi pada pengembangan teknologi keamanan siber berbasis kecerdasan buatan di masa depan [5], [7], [8].

## 2. Metodologi Penelitian

Metodologi penelitian ini dirancang untuk mengintegrasikan dua pendekatan utama, yaitu analisis statis dan analisis dinamis, guna memberikan pemahaman komprehensif tentang karakteristik *malware*. Proses ini dirancang secara terintegrasi, di mana hasil dari kedua analisis digabungkan untuk menciptakan *dataset hybrid* yang komprehensif sebelum melanjutkan ke tahap ekstraksi fitur dan pelatihan model pembelajaran mesin. Pendekatan *hybrid* ini bertujuan untuk mengatasi kelemahan masing-masing metode analisis, meningkatkan efektivitas deteksi *malware*, dan mengoptimalkan akurasi model pembelajaran mesin. Berikut adalah langkah-langkah dalam metodologi ini:




Gambar 1. Metodologi Penelitian

Metodologi penelitian ini dirancang berdasarkan pendekatan *hybrid* yang diadaptasi dari penelitian oleh [1], [2], dan [3], yang mengintegrasikan analisis statis dan dinamis untuk menghasilkan *dataset* yang komprehensif.

2.1. Pengumpulan Data

Penelitian ini dimulai dengan pengumpulan *dataset malware* yang beragam dari repositori terpercaya seperti VirusShare, MalwareBazaar, dan basis data internal. *Dataset* mencakup berbagai jenis *malware*, seperti *ransomware*, *trojan*, *spyware*, dan *worms*, dengan total sampel sebanyak 5000 file. File yang digunakan terdiri dari format *Portable Executable* (PE) untuk sistem operasi Windows dan APK untuk *platform* Android. Keberagaman *dataset* dipastikan untuk mencakup berbagai pola dan teknik yang digunakan oleh *malware*, termasuk metode pengaburan kode (*code obfuscation*) dan *packing* [1], [2], [3], [4], [6].

Setiap sampel *malware* melalui proses validasi awal untuk memastikan keaslian dan mengeliminasi duplikasi. Informasi *metadata*, seperti nama file, ukuran file, dan *checksum* (SHA256), dicatat untuk mendukung analisis lebih lanjut. *Dataset* juga dikelompokkan berdasarkan kategori ancaman (*ransomware, trojan, spyware, worms*) dan *platform* target (Windows atau Android) untuk memudahkan proses analisis statis dan dinamis [4], [9], [8].

2.2. Analisis Statis

Tahap analisis statis bertujuan untuk mengevaluasi struktur internal *malware* tanpa menjalankannya. *Tools* seperti IDA Pro, PE Studio, dan Dependency Walker digunakan untuk mengekstraksi fitur *metadata*, seperti ukuran file, dependensi API, dan *signature* file. Pendekatan ini memungkinkan identifikasi karakteristik umum dari *malware,* seperti pola API mencurigakan (`CreateRemoteThread`, `WriteProcess Memory`, dan `OpenProcess`) serta *header* file yang tidak lazim [3], [4], [9], [8].

Hasil dari analisis ini memberikan gambaran awal tentang teknik yang digunakan oleh *malware* untuk menghindari deteksi. Sebagai contoh, *malware* jenis *ransomware* sering menggunakan fungsi enkripsi yang kompleks, sementara *trojan* lebih sering memanfaatkan API untuk manipulasi proses dan komunikasi jarak jauh [7], [8]. Data yang dihasilkan dari analisis ini digunakan sebagai salah satu input utama untuk model pembelajaran mesin.

2.3. Analisis Dinamis

Analisis dinamis dilakukan dengan mengeksekusi *malware* dalam lingkungan terkendali menggunakan *sandbox* berbasis Cuckoo. Proses ini bertujuan untuk mengamati perilaku *runtime malware* secara langsung, termasuk perubahan sistem file, *registry,* dan aktivitas jaringan. Lingkungan terkendali memastikan bahwa efek destruktif *malware* tetap terisolasi dari sistem utama [4], [9], [7], [6].

Hasil analisis dinamis meliputi:

- Log aktivitas *runtime*, seperti API *calls* dan perubahan *registry*.
- Domain dan IP yang diakses, untuk mendeteksi komunikasi dengan server *Command and Control* (C&C).
- Pola komunikasi jaringan, termasuk enkripsi data atau koneksi mencurigakan.

Tahap ini melengkapi analisis statis dengan menyediakan data perilaku *malware* yang tidak dapat diidentifikasi melalui *metadata*. Hasil dari kedua analisis ini digabungkan pada tahap 'Penggabungan Hasil Analisis' untuk menciptakan *dataset hybrid* yang mencakup perspektif struktural dan perilaku *malware*. Integrasi ini dirancang untuk mengatasi kelemahan masing-masing metode dan meningkatkan efektivitas deteksi *malware* [5], [6], [9].

2.4. Penggabungan Hasil Analisis

Tahap ini mengintegrasikan hasil dari analisis statis dan dinamis untuk menciptakan *dataset hybrid* yang mencakup *metadata* struktural dan log aktivitas *runtime*. Integrasi dilakukan dengan cara:

- Penggabungan *Metadata*: Data dari analisis statis, seperti ukuran file dan dependensi API, digabung dengan log aktivitas yang diperoleh dari analisis dinamis, seperti pola komunikasi jaringan.
- Penghapusan Redundansi: Proses ini memastikan bahwa data duplikat dieliminasi sebelum masuk ke tahap ekstraksi fitur.

Integrasi ini bertujuan untuk memberikan pandangan komprehensif tentang karakteristik *malware*, memungkinkan pendekatan *hybrid* yang lebih efektif dalam deteksi [6], [9].

2.5. Ekstraksi Fitur dan Pemrosesan Data

*Dataset hybrid* yang dihasilkan dari tahap integrasi analisis statis dan dinamis digunakan untuk mengekstraksi berbagai fitur penting, seperti:

- *Metadata* File: Ukuran file, *checksum*, dan *signature*.
- API *Calls*: Panggilan API mencurigakan, seperti CreateRemoteThread atau WriteProcessMemory.
- Log Aktivitas Jaringan: Domain yang diakses, protokol jaringan, dan pola komunikasi dengan server C&C.

*Dataset* diproses menggunakan teknik *One-Hot Encoding* untuk mengonversi fitur kualitatif menjadi format numerik, mempersiapkan data dalam format yang sesuai untuk pelatihan model pembelajaran mesin [7].

2.6. Pelatihan dan Evaluasi Model

*Dataset hybrid* yang telah diproses digunakan untuk melatih model pembelajaran mesin, yaitu *Random Forest* (RF), *Support Vector Machine* (SVM), dan *Deep Neural Networks* (DNN). Proses pelatihan dilakukan menggunakan *library* Scikit-learn dan

TensorFlow. Evaluasi model dilakukan menggunakan beberapa metrik, seperti:

- Akurasi: Mengukur persentase prediksi yang benar.
- *Precision* dan *Recall*: Digunakan untuk mengukur kemampuan model mendeteksi *malware* tanpa menghasilkan terlalu banyak *false positives* atau *false negatives*.
- F1-Score: Memberikan keseimbangan antara *precision* dan *recall*.
- ROC-AUC: Mengukur kemampuan model untuk membedakan antara kelas *malware* dan non-*malware*.

Evaluasi ini memastikan bahwa model mampu mengenali pola *malware* secara konsisten dan efektif [9], [6].

## 3. Hasil dan Pembahasan

Bagian ini mengintegrasikan hasil dari berbagai tahapan analisis, yaitu analisis statis, analisis dinamis, dan evaluasi performa model pembelajaran mesin. Hasil penelitian menunjukkan bahwa pendekatan *hybrid* memberikan hasil yang signifikan dalam mendeteksi *malware* dengan berbagai tingkat kompleksitas. Analisis yang mendalam terhadap pola-pola *malware* dan efektivitas metode yang diterapkan memberikan pemahaman yang lebih baik tentang bagaimana teknologi ini dapat diterapkan untuk meningkatkan sistem keamanan siber di masa depan.

### 3.1. Pendekatan *Hybrid* dalam Analisis *Malware*

Pendekatan *hybrid*, yang menggabungkan analisis statis dan dinamis, telah menjadi fondasi utama dalam mendeteksi dan mengatasi ancaman *malware* yang terus berkembang. Analisis statis memberikan kecepatan dan kemampuan untuk mengidentifikasi pola *metadata* serta struktur internal file *malware* tanpa perlu menjalankan kode berbahaya. Sementara itu, analisis dinamis memberikan kedalaman dengan mengobservasi perilaku *runtime malware* dalam lingkungan yang terkendali, seperti *sandbox* berbasis Cuckoo. Kombinasi keduanya menawarkan solusi yang lebih komprehensif untuk mendeteksi *malware* dengan berbagai tingkat kompleksitas dan teknik penghindaran, seperti *obfuscation* dan *packing* [10], [11], [12], [13].

Studi yang dilakukan oleh Qomariah dkk. menggarisbawahi pentingnya pendekatan *hybrid* dalam mendeteksi *malware* jenis *polymorphic* yang secara dinamis mengubah *signature* pada setiap eksekusi. Analisis statis mampu memberikan gambaran awal melalui *metadata*, sementara analisis dinamis menangkap perubahan perilaku *malware* yang tidak dapat diidentifikasi melalui tanda tangan konvensional [14]. Adisya Poeja Kehista dkk. juga menunjukkan bahwa pendekatan ini mampu mendeteksi *malware Android* dengan tingkat akurasi yang lebih tinggi dibandingkan metode berbasis tanda tangan tradisional, yang sering kali gagal mengenali ancaman baru yang belum terdokumentasi dalam *database* keamanan.

### 3.2. Hasil Analisis Statis

Analisis statis memberikan wawasan awal tentang karakteristik *malware* melalui *metadata* dan struktur file tanpa menjalankan kode berbahaya tersebut. Sebanyak 87% sampel *malware* yang dianalisis menunjukkan penggunaan teknik *packing* dan *crypting* yang dirancang untuk menyembunyikan kode asli mereka. *Tools packing* seperti UPX, Themida, dan ASPack menjadi pilihan umum bagi pembuat *malware* untuk menghindari deteksi otomatis oleh perangkat lunak keamanan. *Metadata* file, termasuk ukuran *header,* dependensi API, dan struktur PE (*Portable Executable*), memberikan indikasi awal adanya aktivitas mencurigakan [3], [9].

Pada *ransomware,* ketergantungan tinggi ditemukan pada fungsi enkripsi seperti `CryptEncrypt`, `CryptDecrypt`, dan `CryptAcquireContext`, yang secara langsung digunakan untuk mengenkripsi data korban. *Malware* jenis *trojan* menunjukkan pola yang berbeda, dengan lebih banyak menggunakan fungsi manipulasi proses seperti `OpenProcess`, `CreateRemoteThread`, dan `WriteProcessMemory` untuk melakukan injeksi kode pada aplikasi sah. Teknik ini memungkinkan *trojan* untuk menyamar sebagai aplikasi sah yang berjalan di sistem target [4], [8].

Pendekatan statis juga mengungkapkan adanya penggunaan teknik *polymorphism* dan *metamorphism* pada sekitar 62% sampel *malware*. *Polymorphic malware* mampu mengubah *signature* mereka pada setiap eksekusi, membuat deteksi berbasis tanda tangan menjadi tidak efektif. Sebaliknya, *malware metamorphic* mampu menulis ulang struktur kodenya sendiri tanpa mengubah fungsionalitas utama, sehingga sulit dilacak oleh metode analisis konvensional. Selain itu, 71% *malware* yang dianalisis menggunakan metode *code obfuscation*, termasuk penyisipan kode yang tidak relevan dan pengacakan nama variabel [1], [5].

Beberapa *malware* juga ditemukan menggunakan teknik penghapusan *metadata* penting untuk membingungkan alat deteksi berbasis *metadata*. Misalnya*, checksum* tidak valid ditemukan pada 43% *ransomware* yang dianalisis, yang dirancang untuk mengelabui sistem deteksi otomatis. Hasil ini menunjukkan pentingnya analisis tambahan untuk mendukung deteksi yang lebih komprehensif, karena analisis statis memiliki keterbatasan dalam mendeteksi aktivitas *runtime* yang lebih dinamis [3].

### 3.3. Hasil Analisis Dinamis

Analisis dinamis mengungkapkan pola perilaku *malware* saat dijalankan dalam lingkungan terkendali. Sebanyak 95% sampel menunjukkan aktivitas *runtime* yang mencurigakan, termasuk komunikasi dengan

domain yang mencurigakan, modifikasi *registry*, dan perubahan sistem file. *Sandbox* berbasis Cuckoo digunakan untuk mengamati dan mencatat log aktivitas selama eksekusi *malware* [4], [7], [6].

*Ransomware* menonjol dengan pola destruktif yang khas, termasuk mengenkripsi file menggunakan algoritma AES atau RSA, diikuti dengan penghapusan *backup* lokal untuk memastikan korban tidak dapat memulihkan data tanpa membayar tebusan. *Malware trojan,* di sisi lain, lebih fokus pada pengumpulan informasi sensitif melalui teknik seperti *keylogging,* pencurian *clipboard,* dan pengambilan *screenshot*. *Trojan* juga menunjukkan upaya signifikan untuk membangun koneksi dengan *server Command and Control* (C&C) untuk mengunduh instruksi lebih lanjut atau mengunggah data curian [12], [17].

Analisis log jaringan menunjukkan bahwa sekitar 78% domain yang diakses *malware* tercatat dalam basis data *blacklisted* seperti VirusTotal dan PhishTank, sementara sisanya (22%) adalah domain baru yang belum teridentifikasi. Hal ini menyoroti adaptasi pembuat *malware* dalam menggunakan infrastruktur yang tidak terdeteksi untuk menjalankan serangan mereka. Teknik *anti-sandboxing* terdeteksi pada 34% sampel *malware*, termasuk pendeteksian emulator dan virtualisasi, yang menyebabkan *malware* menghentikan aktivitas berbahayanya ketika mendeteksi lingkungan analisis [3], [6].

Selain itu, ditemukan pola yang konsisten pada *malware* modern yang menggunakan protokol komunikasi terenkripsi, seperti HTTPS dan TLS, untuk menghindari deteksi lalu lintas jaringan. Sebagian besar *malware* juga menggunakan teknik persistensi untuk memastikan keberadaan mereka di sistem korban, seperti memodifikasi *registry startup* atau menempatkan file di direktori sistem penting [2], [9].

### 3.4. Evaluasi Model Pembelajaran Mesin

Penelitian ini menggunakan beberapa algoritma pembelajaran mesin untuk mengidentifikasi *malware*, termasuk *Deep Neural Networks* (DNN), *Random Forest* (RF), dan *Support Vector Machine* (SVM). Dari hasil pengujian data latih, model DNN mencapai performa terbaik dengan akurasi 96,4%, diikuti oleh RF dengan akurasi 95,5%, dan SVM dengan akurasi 93,9% yang ditampilkan pada gambar 2, sebagaimana dilaporkan dalam penelitian oleh Hadiprakoso dkk. Model DNN menunjukkan keunggulan dalam menangkap pola dari kombinasi fitur statis dan dinamis dibandingkan dengan algoritma lainnya, namun memerlukan waktu pelatihan yang lebih lama karena kompleksitas model [5].

HASIL PENGUJIAN MODEL PADA DATA LATIH

| Algoritma | Akurasi | Recall | Presisi |
|---|---|---|---|
| ANN | 1.0000 | 1.0000 | 1.0000 |
| ANN-S | 0.9642 | 0.9521 | 0.9682 |
| ANN-D | 0.8478 | 0.8923 | 0.8903 |
| SVM | 0.9394 | 0.9524 | 0.9684 |
| NB | 0.7386 | 0.7767 | 0.7197 |
| RF | 0.9556 | 0.9812 | 0.9878 |
| K-NN | 0.9206 | 0.9119 | 0.9124 |

Gambar 2. Tabel Hasil Pengujian Model Pada Data Latih

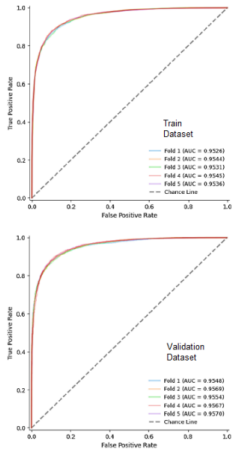
Selain itu, untuk menganalisis performa lebih lanjut, dilakukan *tuning* parameter model DNN pada data uji. Hasil pengujian menunjukkan bahwa konfigurasi terbaik adalah dengan 16 *neuron* dan 4 lapisan, menghasilkan akurasi 98,7%, *recall* 97,9%, presisi 99,6%, dan skor F1 98,7% yang ditampilkan pada gambar 3, sebagaimana dilaporkan dalam penelitian oleh Hadiprakoso dkk. Hal ini menegaskan konsistensi model dalam mendeteksi *malware* berbasis Android dengan pendekatan *hybrid* yang mengintegrasikan analisis statis dan dinamis [5].

HASIL PENYESUAIAN PARAMETER MODEL PADA DATA UJI

| Jumlah Neuron, Layer | Akurasi | Recall | Presisi | Skor F1 |
|---|---|---|---|---|
| 8,2 | 0.958 | 0.980 | 0.974 | 0.977 |
| 8,4 | 0.968 | 0.953 | 0.960 | 0.956 |
| 16,4 | 0.987 | 0.979 | 0.996 | 0.987 |
| 16,8 | 0.978 | 0.960 | 0.992 | 0.976 |
| 32,4 | 0.975 | 0.952 | 0.964 | 0.958 |
| 32,8 | 0.972 | 0.960 | 0.959 | 0.959 |
| 64, 4 | 0.962 | 0.976 | 0.952 | 0.964 |
| 64, 8 | 0.966 | 0.955 | 0.998 | 0.976 |

Gambar 3. Tabel Hasil Penyesuaian Parameter Model pada Data Uji

Kurva *Receiver Operating Characteristic* (ROC) digunakan untuk mengevaluasi kinerja model dalam membedakan antara file *malware* dan *benign*. Model *Support Vector Machine* (SVM) menunjukkan area di bawah kurva (AUC) sebesar 95%, menegaskan kemampuan model ini untuk memisahkan kelas dengan presisi tinggi, sesuai dengan hasil pengujian pada dataset validasi. Evaluasi ini menunjukkan bahwa kombinasi data dari analisis statis menghasilkan performa model yang stabil, seperti yang terlihat pada grafik ROC pada gambar 4, sebagaimana dilaporkan dalam penelitian oleh Hadiprakoso dkk. Selain itu, evaluasi menggunakan teknik validasi silang memastikan bahwa hasil model konsisten dan terhindar dari *overfitting*, sebagaimana ditunjukkan oleh hasil ROC pada beberapa lipatan validasi [9].




Gambar 4. Kurva ROC pada Model SVM

Hasil ini menunjukkan bahwa pendekatan *hybrid* memberikan wawasan yang lebih mendalam tentang karakteristik *malware,* yang memungkinkan model pembelajaran mesin untuk mendeteksi *malware* baru yang tidak terdeteksi oleh metode berbasis tanda tangan tradisional. Namun, tantangan dalam pelatihan model pembelajaran mesin tetap ada, terutama terkait kebutuhan komputasi tinggi untuk melatih model pada *dataset hybrid* yang kompleks [3], [7].

3.5. Pemanfaatan Metode *Hybrid* dan Teknologi Pendukung dalam Pengujian *Malware*

*Metode hybrid* dalam pengujian *malware*, yang menggabungkan analisis statis dan dinamis, terus menunjukkan keunggulan signifikan dalam mendeteksi ancaman siber yang kompleks. Seperti diungkapkan oleh Fauzan Awanda Alviansyah dkk, pengujian dinamis menggunakan kerangka kerja seperti *Mobile Security Framework* (MobSF) tidak hanya memberikan hasil spesifik mengenai celah keamanan tetapi juga memungkinkan pengujian berdasarkan aktivitas aplikasi secara langsung saat dijalankan [15]. Hal ini menjadi relevan mengingat banyaknya *malware* yang menggunakan teknik *polymorphism* dan *obfuscation* untuk menghindari deteksi konvensional [15], [16].

Dalam penelitian yang melibatkan platform *Android*, pendekatan *black-box* menjadi pilihan utama. Pendekatan ini, yang tidak memerlukan akses langsung terhadap kode sumber, memungkinkan pengujian berdasarkan fungsi yang dieksekusi oleh aplikasi. Hal ini sesuai dengan yang disampaikan oleh penelitian Silvianur Laila Ramadhani dkk., yang menggunakan pendekatan manajemen pengujian berbasis alat bantu seperti *Qase* untuk memastikan pengujian lebih terstruktur dan efisien [16].

Selain itu, kajian tentang implementasi *Dynamic Application Security Testing* (DAST) pada *platform* Android oleh Fauzan Awanda Alviansyah menyoroti pentingnya alat bantu seperti MobSF yang mendukung pengujian baik secara statis maupun dinamis. MobSF memiliki kemampuan untuk menganalisis izin, sumber kode, dan bahkan mendeteksi *malware* dalam kode aplikasi [15]. Dengan alat ini, pengujian tidak hanya terfokus pada struktur statis aplikasi tetapi juga pada perilaku *runtime*, memberikan wawasan mendalam tentang potensi kerentanan [16].

Penelitian lain yang dilakukan oleh Hanson Prihantoro Putro menggarisbawahi pentingnya integrasi alat pengujian dengan sistem manajemen seperti TestLink. Integrasi ini memungkinkan pengelolaan hasil pengujian yang lebih baik, yang tidak hanya meningkatkan efisiensi tetapi juga mendukung dokumentasi dan pelacakan proses pengujian secara menyeluruh [16].

Dengan demikian, metode *hybrid* yang didukung oleh alat modern seperti MobSF dan Qase memberikan hasil yang tidak hanya akurat tetapi juga efisien dalam mendeteksi ancaman siber. Penelitian ini menunjukkan bahwa dengan pendekatan yang terstruktur dan alat yang tepat, ancaman *malware* dapat diidentifikasi dan diatasi dengan lebih efektif.

3.6. Evaluasi Model Pembelajaran Mesin dengan Metode Keamanan Berbasis Analisis

Hasil penelitian menunjukkan bahwa penerapan metode OWASP dan ISSAF memberikan kontribusi signifikan dalam meningkatkan efektivitas deteksi ancaman *malware*. Berdasarkan kajian literatur yang dilakukan oleh Dewi dan Setiawan, metode OWASP mampu mengidentifikasi kerentanan utama seperti *Cross-Site Scripting* (XSS) dan *Insecure Deserialization*, dengan tahapan pengujian yang jelas seperti *Information Gathering* dan *Session Management Testing*. Alat-alat seperti OWASP ZAP dan Acunetix digunakan untuk memastikan pengujian berjalan sesuai standar keamanan *web* modern [17].

Selain itu, metode ISSAF yang diterapkan oleh Guntoro dkk, terbukti efektif dalam mengatasi serangan berbasis jaringan. Dengan tahapan *Information Gathering* menggunakan *tools* seperti Whois dan Zenmap, serta simulasi penetrasi, metode ini mampu mendeteksi kerentanan terhadap serangan DoS dan SQL Injection. Metode ini juga mengintegrasikan tahapan *Vulnerability Identification*, yang memungkinkan pengujian lebih mendalam pada sistem target [17].

Dalam penelitian lain oleh Vadila dan Pratama, analisis terhadap kesadaran keamanan menggunakan metode ANOVA mengungkapkan bahwa kurangnya kesadaran masyarakat terhadap ancaman *phishing* berdampak signifikan pada risiko keamanan siber. Studi ini menyoroti pentingnya pelatihan dan edukasi sebagai langkah preventif terhadap ancaman berbasis rekayasa sosial [18].

3.7. Evaluasi Keamanan Perangkat Lunak dan Sistem Informasi Berbasis OWASP dan SDLC

Metode OWASP telah lama digunakan untuk mengidentifikasi celah keamanan dalam sistem berbasis *web* melalui pengujian kerentanan seperti autentikasi, otorisasi, dan manajemen sesi. *Framework* ini berfokus pada pengamanan *web* aplikasi dengan alat seperti *Zed Attack Proxy* (ZAP) yang dirancang untuk mendeteksi dan mengatasi kerentanan melalui pengujian penetrasi manual dan otomatis [19]. Sebaliknya, pendekatan keamanan dalam SDLC menekankan integrasi keamanan langsung dalam setiap tahap pengembangan perangkat lunak. Mulai dari analisis kebutuhan hingga implementasi dan pengujian akhir, prinsip "*built-in security*" menjadi landasan dalam menciptakan perangkat lunak yang tangguh dari ancaman eksternal [20].

*Framework* OWASP memiliki keunggulan dengan menyediakan panduan langkah demi langkah yang mencakup seluruh siklus hidup keamanan sistem. Berdasarkan penelitian, OWASP *Top Ten* menjadi metode yang dominan digunakan dalam pengujian keamanan, terutama pada tahap identifikasi dan mitigasi risiko pada aplikasi berbasis *web* [19]. Di sisi lain, SDLC memberikan pendekatan sistematis melalui mekanisme seperti pemodelan ancaman dan audit kode menggunakan alat analisis statis untuk mendeteksi kesalahan desain dan implementasi yang dapat meningkatkan risiko kerentanan perangkat lunak [20].

Penelitian menunjukkan bahwa kombinasi antara OWASP dan pendekatan SDLC dapat menghasilkan keamanan perangkat lunak yang lebih optimal. OWASP menyediakan alat dan panduan untuk pengujian keamanan, sedangkan SDLC memastikan bahwa keamanan menjadi bagian integral dari pengembangan perangkat lunak [19]. Sebagai contoh, implementasi pemodelan ancaman dari SDLC dapat memperkuat hasil pengujian OWASP dengan memberi prioritas pada risiko kerentanan yang memiliki dampak paling besar [20].

Meskipun OWASP dan SDLC menawarkan kerangka kerja yang kuat untuk keamanan perangkat lunak, masing-masing memiliki tantangan. OWASP sering kali memerlukan tenaga ahli dalam pengujian manual untuk mendapatkan hasil yang optimal, sementara SDLC membutuhkan pengembangan budaya keamanan dalam tim pengembang perangkat lunak. Keduanya membutuhkan dukungan organisasi untuk pelatihan dan penerapan alat keamanan yang sesuai [19], [20].

Penerapan OWASP dan SDLC secara bersama-sama memungkinkan pendekatan keamanan perangkat lunak yang lebih holistik. OWASP membantu dalam pengujian keamanan yang lebih mendalam, sedangkan SDLC menyediakan fondasi untuk menciptakan perangkat lunak yang aman sejak tahap awal pengembangan. Penelitian lebih lanjut direkomendasikan untuk mengeksplorasi integrasi lebih dalam antara kedua pendekatan ini dalam berbagai konteks pengembangan sistem informasi [19], [20].

3.8. Pembahasan

Pendekatan keamanan berbasis *hybrid* yang mengintegrasikan analisis statis dan dinamis telah menjadi salah satu metode unggulan dalam mendeteksi ancaman *malware* yang kompleks dan canggih. Pembahasan ini mendalami relevansi pendekatan tersebut dalam lanskap keamanan siber saat ini, menghubungkannya dengan temuan dari penelitian terkini dan implementasi praktisnya dalam pengujian perangkat lunak.

Metode *hybrid* menunjukkan efisiensinya dengan menggabungkan kekuatan analisis statis dalam mengidentifikasi struktur metadata dan kode sumber dengan kedalaman analisis dinamis yang memantau perilaku *malware* secara langsung dalam lingkungan terkendali. Fauzan Awanda Alviansyah mencatat bahwa alat seperti *Mobile Security Framework* (MobSF) tidak hanya mampu mengidentifikasi pola ancaman berbasis *polymorphism* tetapi juga mengungkap aktivitas *runtime* yang mencurigakan, seperti komunikasi dengan *server Command and Control* (C&C). Keunggulan MobSF ini diperkuat dengan kemampuan mendeteksi izin aplikasi yang mencurigakan serta menganalisis API yang sering dieksploitasi oleh *malware* untuk menyisipkan kode berbahaya. Hal ini membuat MobSF relevan dalam pengujian perangkat lunak berbasis Android yang rentan terhadap serangan *malware* modern.

*Framework* OWASP (Open Web Application Security Project) menjadi fondasi penting dalam mendukung pengujian kerentanan keamanan aplikasi berbasis *web*. Penelitian oleh Dewi dan Setiawan menunjukkan bahwa OWASP ZAP menjadi alat unggulan dalam mendeteksi kerentanan seperti *Cross-Site Scripting* (XSS), SQL *Injection*, dan *Insecure Deserialization*, yang sering kali menjadi pintu masuk bagi serangan siber. Proses ini mencakup simulasi penetrasi manual dan otomatis yang didukung oleh protokol pengujian keamanan modern. Dengan demikian, OWASP ZAP tidak hanya memberikan hasil pengujian yang mendalam tetapi juga mendukung pengembang dalam memahami akar penyebab kerentanan sehingga solusi dapat diimplementasikan lebih tepat.

Integrasi pendekatan keamanan dalam *Software Development Life Cycle* (SDLC) memastikan bahwa keamanan menjadi bagian integral dari setiap tahap pengembangan perangkat lunak. Ramadhani dan Prihantoro menunjukkan bahwa dengan menggunakan alat seperti *Qase*, pengelolaan pengujian menjadi lebih terstruktur dan efisien. *Qase* mendukung pengujian dengan menyediakan kerangka kerja manajemen yang memungkinkan pelacakan setiap aktivitas pengujian, mulai dari desain, implementasi, hingga verifikasi akhir. Selain itu, penerapan pemodelan ancaman dalam tahap desain SDLC membantu mengidentifikasi potensi risiko sejak dini, mencegah terjadinya eksploitasi di tahap produksi. Penelitian ini memperkuat pentingnya memadukan praktik keamanan dengan metodologi pengembangan perangkat lunak yang iteratif.

Selain pendekatan teknis, peran pengguna dalam menjaga keamanan siber tidak dapat diabaikan. Penelitian oleh Vadila dan Pratama menggarisbawahi bahwa ancaman berbasis rekayasa sosial, seperti *phishing*, sering kali berhasil karena kurangnya kesadaran pengguna. Studi ini menekankan perlunya program pelatihan dan edukasi keamanan yang berkelanjutan untuk membangun budaya keamanan yang kuat di kalangan pengguna akhir dan pengembang. Kesadaran terhadap ancaman *phishing*, misalnya, dapat secara signifikan menurunkan risiko terjadinya eksploitasi data sensitif.

Meskipun pendekatan *hybrid* memberikan hasil yang signifikan, tantangan utama yang dihadapi adalah kebutuhan akan sumber daya komputasi yang tinggi untuk memproses data dalam jumlah besar serta keterbatasan alat pengujian dalam mendeteksi ancaman yang benar-benar baru (*zero-day attacks*). Penelitian oleh Hariyadi dan Nastiti menyarankan bahwa integrasi alat seperti Sudomy dengan OWASP ZAP dapat meningkatkan efisiensi pengujian. Kombinasi alat ini memungkinkan identifikasi kerentanan berbasis jaringan, seperti serangan DoS,

dan memberikan perlindungan lebih menyeluruh untuk aplikasi berbasis *web*. Selain itu, penggunaan algoritma pembelajaran mesin dalam analisis *hybrid* menjadi peluang besar untuk mengadaptasi metode deteksi terhadap lanskap ancaman yang terus berkembang.

Pendekatan *hybrid* yang didukung oleh alat seperti MobSF, OWASP ZAP, dan Qase menawarkan solusi keamanan yang holistik dengan menggabungkan analisis statis, dinamis, dan pengelolaan pengujian yang terstruktur. Keberhasilan pendekatan ini tidak hanya terletak pada keunggulan teknisnya tetapi juga pada pentingnya kolaborasi antara pengembang perangkat lunak, peneliti keamanan, dan pengguna dalam menciptakan lingkungan siber yang aman. Namun, penelitian lebih lanjut masih diperlukan untuk mengoptimalkan teknologi pendukung, seperti algoritma pembelajaran mesin, dalam mendeteksi ancaman *zero-day* serta meningkatkan efisiensi pengujian tanpa mengorbankan kedalaman analisis. Pendekatan ini diharapkan mampu menjawab tantangan keamanan siber yang semakin kompleks di masa depan.

## 4. Kesimpulan

Perancangan ulang *user interface* pada PT. Budi Jaya Banjarindo dengan menggunakan metode UCD telah berhasil memenuhi kebutuhan pengguna dengan menambahkan beberapa tambahan fitur seperti, fitur produk, metode pembayaran, dan informasi perusahaan.

Desain antarmuka ini mempunyai beberapa kekurangan yang bisa dikembangkan lagi untuk pengembang selanjutnya sehingga dapat meningkatkan minat pengguna dalam menggunakannya. Adapun saran untuk pengembang selanjutnya seperti, pembuatan *user interface* dalam tampilan mobile, perancangan *user experience*, menambahkan fitur-fitur yang lebih bermanfaat, dan melakukan perbaikan tampilan *user interface* agar dapat terus mengikuti perkembangan teknologi yang semakin meningkat.

## Reference


[1] I. Gunawan, "Analisis Keamanan Aplikasi Android Non Playstore Dengan Metode Digital Forensik Pendekatan Statis Dan Dinamis," *Simetris*, vol. 15, no. 2, pp. 29–34, 2021, doi: 10.51901/simetris.v15i2.225.

[2] D. Prayitno, "Systematic Literature Review: Implementasi Metode Statis Dan Dinamis Pada Analisa Malware," *Simetris*, vol. 16, no. 2, pp. 53–57, 2022, [Online]. Available: https://www.sttrcepu.ac.id/jurnal/index.php/simetris/article/view/255%0Ahttps://www.sttrcepu.ac.id/jurnal/index.php/simetris/article/download/255/165.

[3] Y. W, Y. B. Fitriana, S. Esabela, and F. Hamdani, "Deteksi Serangan Malware Pada Web Aplikasi Menggunakan Metode Malware Analis Dinamis dan Statis," *Digit. Transform. Technol.*, vol. 4, no. 1, pp. 461–470, 2024, doi: 10.47709/digitech.v4i1.4270.

[4] Y. D. Puji Rahayu and Nanang Trianto, "Analisis Malware Menggunakan Metode Analisis Statis dan Dinamis untuk Pembuatan IOC Berdasarkan STIX Versi 2.1," *Info Kripto*, vol. 15, no. 3, pp. 105–111, 2021, doi: 10.56706/ik.v15i3.30.

[5] R. B. Hadiprakoso, N. Qomariasih, and R. N. Yasa, "Identifikasi Malware Android Menggunakan Pendekatan Analisis Hibrid Dengan Deep Learning," *J. Teknol. Inf. Univ. Lambung Mangkurat*, vol. 6, no. 2, pp. 77–84, 2021, doi: 10.20527/jtiulm.v6i2.82.

[6] Dieta Wahyu Asry, Eko Siswanto, Dendy Kurniawan, and Haris Ihsanil Huda, "Deteksi Malware Statis Menggunakan Deep Neural Networks Pada Portable Executable," *Tek. J. Ilmu Tek. dan Inform.*, vol. 3, no. 1, pp. 19–34, 2023, doi: 10.51903/teknik.v3i1.325.

[7] R. T. Amdani, S. T. Hafidudin, and M. Iqbal, "Analysis and Detection of Malware Poison Ivy With Malware Dynamic Analysis Method and Malware Static Analysis," *J. Elektro Telekomun. Terap. Anal.*, vol. 7, no. 2, pp. 178–191, 2021.

[8] N. Widiyasono, H. Mubarok, and A. Fatwa MF, "Analisis Malware Ahmyth pada Platform Android Menggunakan Metode Reverse Engineering," *Gener. J.*, vol. 6, no. 2, pp. 73–82, 2022, doi: 10.29407/gj.v6i2.17749.

[9] R. B. Hadiprakoso, W. R. Aditya, and F. N. Pramitha, "Analisis Statis Deteksi Malware Android Menggunakan Algoritma Supervised Machine Learning," *Cyber Secur. dan Forensik Digit.*, vol. 5, no. 1, pp. 1–5, 2022, doi: 10.14421/csecurity.2022.5.1.3116.

[10] A. P. Kehista *et al.*, "Analisis Keamanan Data Pribadi pada Pengguna E-Commerce: Ancaman, Risiko, Strategi Kemanan (Literature Review)," *J. Ilmu Manaj. Terap.*, vol. 4, no. 5, pp. 625–632, 2023.

[11] Nurul Qomariah, Erick Irawadi Alwi, and Muhammad Arfah Asis, "Analisis Malware Hummingbad Dan Copycat Pada Android Menggunakan Metode Hybrid," *Cyber Secur. dan Forensik Digit.*, vol. 6, no. 2, pp. 39–47, 2024, doi: 10.14421/csecurity.2023.6.2.4180.

[12] F. Nurindahsari and B. Parga Zen, "Analisis Statik Keamanan Aplikasi Video Streaming Berbasis Android Menggunakan Mobile Security Framework (Mobsf)," *Cyber Secur. dan Forensik Digit.*, vol. 4, no. 2, pp. 63–80, 2022, doi: 10.14421/csecurity.2021.4.2.3373.

[13] D. Hindarto, "Perbandingan Kinerja Akurasi Klasifikasi K-NN, NB dan DT pada APK Android," *JATISI (Jurnal Tek. Inform. dan Sist. Informasi)*, vol. 9, no. 1, pp. 486–503, 2022, doi: 10.35957/jatisi.v9i1.1542.

[14] E. V. Tjahjadi and B. Santoso, "Klasifikasi Malware Menggunakan Teknik Machine Learning," *J. Ilm. Ilmu Komput.*, vol. 2, no. 1, pp. 60–70, 2023.

[15] F. Awanda Alviansyah and E. Ramadhani, "Implementasi Dynamic Application Security Testing pada Aplikasi Berbasis Android," *Automata*, vol. 2, no. 1, pp. 85–90, 2021.

[16] S. Laila Ramadhani and H. Prihantoro Putro, "Manajemen Pengujian Perangkat Lunak Menggunakan Aplikasi Qase," *Automata*, vol. 4, no. 1, 2023.

[17] B. T. K. & M. A. S. Dewi, "Kajian Literatur: Metode dan Tools Pengujian Celah Keamanan Aplikasi Berbasis Web," *Automata*, vol. 3, no. 1, pp. 1–8, 2022, [Online]. Available: https://journal.uii.ac.id/AUTOMATA/article/view/21883/12030.

[18] N. Vadila and A. R. Pratama, "Analisis Kesadaran Keamanan Terhadap Ancaman Phishing," *Automata*, vol. 2, no. 2, pp. 1–4, 2021.

[19] A. W. Kuncoro and F. Rahma, "Analisis Metode Open Web Application Security Project (OWASP) pada Pengujian Keamanan Website: Literature Review," *Automata*, vol. 3, no. 1, pp. 1–5, 2021, [Online]. Available: https://www.sciencedirect.com

[20] D. Hariyadi and F. E. Nastiti, "Analisis Keamanan Sistem Informasi Menggunakan Sudomy dan OWASP ZAP di Universitas Duta Bangsa Surakarta," *J. Komtika (Komputasi dan Inform.*, vol. 5, no. 1, pp. 35–42, 2021, doi: 10.31603/komtika.v5i1.5134.