Implementasi Sistem Deteksi Malware Berbasis *Random Forest* pada File *Portable Executable Windows*

Cindy Laura Tampubolon[1], Nur Cahyono Kushardianto[2]

[1, 2] Rekayasa Keamanan Siber, Politeknik Negeri Batam, Indonesia

Email: [1]cindyyy.laura@gmail.com, [2]anung@polibatam.ac.id

Received : Jun 9, 2025; Revised : Nov 20, 2025; Accepted : Nov 22, 2025; Published : Dec 11, 2025
Phone Number : +6281266144248

Abstract

This research addresses the limitations of signature based malware detection against new variants in Windows Portable Executable files, along with the absence of an automated system with real time monitoring that requires no user intervention. Limited and homogeneous benign samples also introduce bias and a high false positive rate. To address this, the research develops a Random Forest based malware detection system integrated with a web dashboard and event driven file monitoring. The contribution includes designing an automated system, extracting static features from PE headers without executing files, and presenting confidence scores and feature importance on a web interface. The method involves feature extraction with the pefile library, directory monitoring with the Watchdog library, and training a Random Forest model on the PE-Malware-Dataset containing malware samples from five families along with benign samples from the Windows system. The research began modeling with an exploration of multi class classification, but accuracy remained low because header features did not sufficiently distinguish between families, so the final approach was set as binary classification, malware versus benign. Real time testing results confirmed that the end to end pipeline runs automatically. Initial evaluation on a small test set showed the model captured all malware samples but produced false positives on benign files, resulting in high recall paired with low precision and a moderate F score. Most false positives came from third party administrative utilities with a Portable Executable structure similar to malware. In conclusion, the system functions as a proof of concept for real time detection based on Random Forest and a web dashboard, but real world performance remains limited by benign data bias, so diversifying benign samples is needed to reduce false positives.

Keywords : *Malware, Random Forest, Watchdog Python, Dashboard monitoring, Klasifikasi biner*

This work is an open access article licensed under a Creative Commons Attribution 4.0 International License.

1. INTRODUCTION

Sistem operasi Windows merupakan salah satu sistem operasi yang paling banyak digunakan di dunia, sehingga sering menjadi target utama penyebaran malware di berbagai sektor, mulai dari perkantoran hingga instansi pemerintahan [1]. Malware pada file Portable Executable (PE) Windows mencakup berbagai jenis perangkat lunak berbahaya seperti ransomware, spyware, backdoor, trojan, dan downloader, dengan dampak serius berupa pencurian data sensitif, kerusakan sistem, hingga gangguan operasional pengguna [2]. Windows menggunakan format file PE sebagai standar untuk seluruh program yang dapat dijalankan, termasuk file berekstensi .exe, .dll, dan .sys, sehingga file PE menjadi vektor serangan utama yang dieksploitasi oleh pelaku kejahatan siber [1].

Metode deteksi *malware* konvensional yang umumnya berbasis tanda tangan *(signature-based)* memiliki keterbatasan mendasar dalam menghadapi ancaman malware modern. Pendekatan ini hanya mampu mengenali malware yang pola atau tanda tangannya sudah diketahui sebelumnya, sehingga tidak efektif dalam mendeteksi varian *malware* baru yang belum terdaftar dalam basis data antivirus [3]. Keterbatasan ini mendorong pengembangan pendekatan berbasis *machine learning* yang lebih adaptif, karena mampu mempelajari pola karakteristik malware dari data historis secara otomatis dan dapat

mendeteksi ancaman baru yang belum pernah dikenal sebelumnya [3]. Salah satu algoritma machine learning yang terbukti efektif untuk klasifikasi malware adalah *Random Forest* yang unggul dalam menangani dataset berskala besar, memiliki waktu pelatihan singkat, menghasilkan akurasi prediksi tinggi, mampu mengukur feature importance, serta meminimalkan risiko *overfitting* [4].

Penelitian [1] mengkaji strategi perlindungan berlapis pada sistem operasi Windows menggunakan *Windows Defender*, *Firewall*, dan identifikasi peran *machine learning* untuk deteksi *on demand*, namun bersifat kajian literatur dan tidak menghasilkan sistem atau model klasifikasi malware secara konkret. Penelitian [2] melakukan analisis perilaku malware pada Windows 10 secara dinamis menggunakan Cuckoo Sandbox dengan tingkat deteksi VirusTotal mencapai 67%, namun pendekatan ini rentan terhadap teknik sandbox evasion dan belum mengintegrasikan machine learning untuk klasifikasi otomatis. Berdasarkan kajian tersebut, terdapat research gap yang belum terpenuhi, yaitu belum adanya sistem deteksi malware PE Windows yang bersifat otomatis, mencakup cakupan malware yang luas, dan dapat berjalan secara real-time tanpa intervensi pengguna.

Oleh karena itu, tujuan penelitian ini adalah merancang dan membangun sistem deteksi malware berbasis Random Forest pada file PE Windows yang diintegrasikan dengan antarmuka web sebagai dashboard monitoring berbasis *event-driven file system monitoring* menggunakan library *Watchdog Python.* Sistem mengekstrak fitur statis dari PE header secara otomatis menggunakan library pefile tanpa mengeksekusi file, sehingga proses analisis sepenuhnya aman dari risiko aktivasi payload malware. Kontribusi penelitian ini meliputi: (1) perancangan sistem deteksi malware otomatis yang terintegrasi dengan dashboard web real-time; (2) ekstraksi fitur statis dari PE header tanpa eksekusi file; (3) penyajian confidence score dan feature importance pada antarmuka web; dan (4) evaluasi kritis terhadap performa sistem, termasuk identifikasi sumber false positive dan bias dataset benign yang digunakan dalam pelatihan model.

2. STUDI LITERATUR

2.1. Penelitian Terkait

Tabel 1 menyajikan perbandingan penelitian terdahulu yang relevan.

Table 1. Penelitian Terdahulu

Peneliti / Tahun	Metode	Hasil	Keterbatasan
Zulaeha, Z., Mulqiya, W.Z., & Rilvani, E. (2024)	Studi literatur, analisis pertahanan Windows, Windows Defender, Firewall, Machine Learning untuk deteksi on demand.	Memetakan kelemahan Windows dan mitigasi berlapis; mengidentifikasi peran ML dalam deteksi ancaman	Tidak menghasilkan sistem/model klasifikasi malware konkret; bersifat kajian teoritis
Adiyatma, M.R., Setiawan, H., & Tasmi (2025)	Dynamic Analysis, Cuckoo Sandbox v2.0.7, VirusTotal, Wireshark, Hash MD5 & SHA-1, VirtualBox	Mengungkap perilaku nyata malware (WannaCry, trojan generik); deteksi VirusTotal 67%	Rentan sandbox evasion; boros sumber daya; belum ada ML untuk klasifikasi otomatis
Matin, I.M.M., Agustin, M., Sugiarto, B., & Asri, A.N. (2023)	Decision Tree, Random Forest, AdaBoost, Bagging, Hist Gradient Boosting, Dataset ClaMP PE Windows,	Akurasi tertinggi 96,9% oleh Hist Gradient Boosting	Tidak menghasilkan aplikasi siap pakai; tanpa antarmuka pengguna

	Confusion Matrix, Cross Validation (CV=10)		
Alvanof, M., Bustami, & Dinata, R.K. (2024)	Random Forest, Python (Scikit-learn), Feature Selection, Dataset Ransomware (1380 sampel, 54 fitur), Confusion Matrix, Aplikasi berbasis Streamlit	Akurasi 98,79%; model terimplementasi dalam aplikasi web.	Fokus hanya ransomware; dataset kecil (1380 sampel); belum mencakup malware PE Windows umum.

Berdasarkan Tabel 1, belum adanya penelitian yang membangun sistem deteksi malware PE Windows berbasis Random Forest yang dilengkapi antarmuka web, mencakup berbagai jenis malware secara komprehensif, dan dapat dioperasikan secara on demand tanpa memerlukan lingkungan sandbox yang kompleks. Malware mencakup berbagai jenis perangkat lunak berbahaya seperti virus, *worm, trojan, ransomware, spyware,* dan *adware*, yang masing-masing memiliki karakteristik serangan berbeda namun seluruhnya menimbulkan risiko serius terhadap keamanan data dan integritas sistem operasi[1]

2.2 Landasan Teori

2.2.1 Malware

Malware adalah perangkat lunak yang dirancang secara khusus untuk menyusup, merusak, mencuri data dan mengganggu operasional system tanpa sepengetahuan pemilik system operasi [2]. Dalam penelitian ini, lima keluarga malware yang menjadi fokus adalah *Generic Malware, Spyware, Downloader, Ransomware,* dan *Backdoor*, yang kemudian disederhanakan menjadi klasifikasi biner *malware* versus *benign* pada tahap pemodelan akhir.

2.2.2. Sistem operasi windows dan file portable executable (PE)

Windows merupakan sistem operasi dengan tingkat adopsi tertinggi di dunia, menjadikannya target utama serangan malware di berbagai sektor [1]. Windows menggunakan format *Portable Executable* (PE) sebagai standar biner untuk seluruh program yang dapat dijalankan, termasuk file berekstensi .exe, .dll, dan .sys. File Portable Executable (PE) adalah format biner standar yang digunakan pada sistem operasi Windows untuk file yang dapat dieksekusi. Struktur file PE terdiri dari beberapa komponen header yang masing-masing menyimpan informasi penting mengenai karakteristik program. Struktur file PE terdiri atas tiga komponen header utama: (1) DOS Header berisi *magic number* MZ sebagai penanda file PE serta informasi kompatibilitas dengan sistem DOS lama; (2) COFF File Header menyimpan informasi dasar seperti arsitektur prosesor target, jumlah *section*, dan *timestamp* kompilasi; dan (3) Optional Header memuat informasi teknis detail seperti alamat *entry point*, ukuran kode, versi sistem operasi minimum, dan *subsystem*. Karena setiap file PE, baik *benign* maupun malware, harus mengikuti struktur ini, informasi pada header dapat diekstraksi menjadi fitur numerik yang merepresentasikan karakteristik file secara statis, tanpa perlu menjalankan file yang bersangkutan.

2.2.3 Algoritma Random Forest

Random Forest merupakan algoritma *ensemble machine learning* yang bekerja dengan membangun sejumlah pohon keputusan (*decision tree*) secara acak selama proses pelatihan, kemudian menggabungkan seluruh hasil prediksi melalui mekanisme *voting* mayoritas untuk menghasilkan prediksi akhir [4]. Keunggulan utama algoritma ini meliputi kemampuan menangani dataset berskala besar tanpa *preprocessing* yang kompleks, waktu pelatihan yang relatif singkat, akurasi prediksi yang tinggi, minimnya risiko *overfitting* karena menggabungkan prediksi dari banyak pohon keputusan yang berbeda, serta kemampuan mengukur tingkat kepentingan fitur (*feature importance*) yang membantu

memahami fitur mana yang paling berpengaruh dalam menentukan apakah sebuah file PE merupakan malware. Karena karakteristik tersebut, Random Forest dipilih sebagai algoritma utama dalam penelitian ini untuk membangun model klasifikasi malware berbasis fitur statis header PE.

2.2.4. CRISP-DM (Cross Industry Standard Process for Data Mining)

CRISP-DM adalah metodologi standar yang dirancang khusus untuk proyek berbasis *data mining* dan *machine learning*, mencakup enam fase yang saling berkorespondensi dengan tahapan penelitian, yaitu *Business Understanding* (memahami permasalahan dan tujuan penelitian), *Data Understanding* (mengumpulkan dan mengeksplorasi dataset), *Data Preparation* (membersihkan data dan mengekstraksi fitur), *Modeling* (melatih model klasifikasi), *Evaluation* (mengevaluasi performa model menggunakan metrik standar), dan *Deployment* (menerapkan model ke dalam sistem yang dapat digunakan secara nyata). CRISP-DM dipilih karena mencakup seluruh siklus penelitian, mulai dari pemahaman masalah hingga pembangunan produk sistem berbasis web, sehingga sesuai dengan tujuan penelitian yang tidak hanya menghasilkan model, tetapi juga sistem terapan.

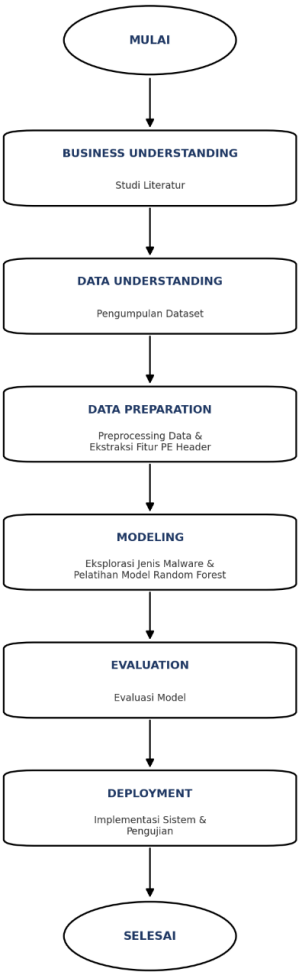
2.2.5. Static Analysis dan Library Pefile

Static analysis adalah teknik analisis file yang dilakukan tanpa mengeksekusi file tersebut, sehingga proses analisis sepenuhnya aman dan tidak berisiko mengaktifkan *payload* malware. Dalam penelitian ini, *static analysis* dilakukan menggunakan *library* pefile pada Python, yang memungkinkan pembacaan dan penguraian (*parsing*) struktur header PE termasuk DOS Header, COFF File Header, dan Optional Header secara langsung dari berkas biner, tanpa menjalankan file sebagai proses aktif. Fitur-fitur yang diekstraksi kemudian direpresentasikan dalam bentuk vektor numerik sebagai masukan bagi model Random Forest.

2.2.6. Library Watchdog

Watchdog adalah *library* Python yang digunakan untuk memantau perubahan pada sistem berkas (*file system*) secara *event-driven*. Watchdog dapat mendeteksi kejadian seperti pembuatan (*created*), modifikasi (*modified*), maupun penghapusan file pada suatu direktori secara langsung tanpa memerlukan pemeriksaan berkala (*polling*) oleh pengguna. Dalam penelitian ini, watchdog berperan sebagai *service background* yang memonitor direktori target secara kontinu; setiap kali file PE baru terdeteksi, watchdog memicu *pipeline* ekstraksi fitur dan prediksi *Random Forest* secara otomatis, sehingga proses deteksi berjalan secara *real-time* tanpa intervensi pengguna.

3. ANALISIS DAN PERANCANGAN



Gambar 1. Alur penelitian

Gambar 1 menggambarkan alur penelitian yang mengikuti enam fase metodologi CRISP-DM, mulai dari Business Understanding berupa studi literatur hingga Deployment berupa implementasi dan pengujian sistem. Setiap fase pada diagram berkorespondensi langsung dengan subbab pembahasan pada Bab 3, sehingga alur ini menjadi kerangka acuan penyusunan metodologi penelitian secara keseluruhan.

3.1. Data understanding

Tahap ini berkorespondensi sebagai fase pengumpulan data. Dataset utama yang digunakan dalam penelitian ini adalah PE-Malware-Dataset [2], terdiri dari 18.072 sampel file PE Windows yang seluruhnya merupakan sampel malware terverifikasi dan dikumpulkan dari malware bazaar menggunakan API. Dataset ini memiliki 52 fitur yang diekstrak dari komponen header PE Header.csv dan diklasifikasikan ke dalam lima kelas label malware, yaitu *Spyware, Ransomware, Downloader, Backdoor,* dan *Generic Malware*. Pelabelan kelas dilakukan menggunakan VirusTotal API dengan mekanisme *majority voting* untuk memastikan akurasi label setiap sampel [2]. Karena dataset tersebut tidak menyertakan sampel file *benign*, penelitian ini menambahkan dataset pelengkap berupa 4.039 sampel file .dll yang diambil dari direktori System32 Windows sebagai representasi kelas negatif. Penggabungan kedua sumber data menghasilkan dataset awal berjumlah 22.590 sampel dengan distribusi enam kelas terdiri atas lima kelas malware dan satu kelas *benign*. Distribusi jumlah sampel tiap kelas ditunjukkan pada Tabel 2. Untuk pendekatan klasifikasi biner, kelima keluarga malware digabung menjadi satu kelas malware, sehingga dataset akhir terdiri atas dua kelas, yaitu malware dan benign.

Table 2. Distribusi kelas dataset

Kelas	Jumlah Sampel	Persentase
Generic Malware	6.231	27,58 persen
Spyware	5.766	25,52 persen
Downloader	2.438	10,79 persen
Ransomware	2.376	10,52 persen
Backdoor	1.740	7,70 persen
Benign	4.039	17,88 persen
Total	**22.590**	**100 persen**

Distribusi jumlah sampel tiap kelas ditunjukkan pada Tabel 2. Tabel tersebut memperlihatkan bahwa kelas malware secara kolektif (Generic Malware, Spyware, Downloader, Ransomware, dan Backdoor) mendominasi dataset dengan total 82,12 persen sampel, sedangkan kelas Benign hanya menyumbang 17,88 persen.

3.2 Data Preparation

Setelah tahap pengumpulan data pada sub bab 3.1 menghasilkan dataset gabungan sebesar 22.590 sampel, tahap berikutnya adalah mempersiapkan data tersebut agar layak menjadi masukan model. Tahap ini mencakup empat langkah utama. Pertama, pembersihan data: data mentah hasil ekstraksi fitur diperiksa terhadap nilai kosong (*missing value*) dan nilai tidak valid hasil parsing PE header. Kedua, penyamaan format data agar seluruh fitur numerik konsisten sebagai masukan model. Ketiga, normalisasi fitur menggunakan metode *Min-Max Scaling* agar rentang nilai antar fitur sebanding. Keempat, pembagian dataset menjadi data latih dan data uji dengan komposisi 80:20. Ekstraksi fitur dilakukan secara statis menggunakan *library* Pefile Python tanpa mengeksekusi file, sehingga proses analisis sepenuhnya aman dan tidak berisiko mengaktifkan *payload* malware [3], [13], [14]. Fitur yang diekstrak meliputi tiga komponen utama header PE, terdiri atas DOS Header, COFF File Header, dan Optional Header. Seluruh fitur direpresentasikan dalam bentuk vektor numerik sebagai masukan proses pelatihan model.

Table 3. Hasil pembagian data

Subnet	Jumlah Sampel	Malware	Benign
Data latih	18.072	-	-
Data uji	4.518	3.710	808
Total	22.590		

3.2. Modelling

Penelitian ini mengawali pemodelan dengan eksplorasi multi-kelas terhadap lima keluarga malware. Eksplorasi bertujuan menguji kemampuan fitur header PE membedakan tiap keluarga. Hasil eksplorasi menunjukkan akurasi 32,07 persen dan F1-score makro 29,19 persen, dengan performa terendah pada kelas Backdoor (precision 0,13, recall 0,22) dan performa tertinggi pada kelas Generic Malware (precision 0,45, recall 0,39). Akurasi rendah ini mengindikasikan fitur header PE kurang mampu membedakan karakteristik antar keluarga malware secara spesifik, sehingga pendekatan akhir ditetapkan sebagai klasifikasi biner, malware lawan benign. Model *Random Forest* dibangun dengan library Scikit-learn Python memakai *hyperparameter* teroptimasi. *Random Forest* membangun sejumlah pohon keputusan secara acak pada proses pelatihan, lalu menggabungkan seluruh prediksi melalui *voting* mayoritas untuk menghasilkan prediksi akhir [4]. Algoritma ini unggul dalam menangani dataset berskala besar, memiliki waktu pelatihan singkat, menghasilkan akurasi tinggi, mengukur *feature importance*, dan meminimalkan risiko *overfitting*. Kombinasi *hyperparameter* terbaik dipilih berdasarkan metrik F1-score karena distribusi kelas tidak seimbang. Untuk menangani

ketidakseimbangan kelas, penelitian ini menerapkan SMOTE pada data latih dan parameter *class_weight='balanced'* pada RandomForestClassifier [8]. Model akhir disimpan dalam format .pkl dengan library *joblib* untuk tahap penerapan.

3.3. Evaluation

Model yang telah dilatih pada subbab 3.3 selanjutnya diuji performanya menggunakan data uji yang telah dipisahkan sejak subbab 3.2, sebelum diterapkan pada tahap deployment. Evaluasi menggunakan data uji sebesar 20 persen melalui *confusion matrix*. *Confusion matrix* menghasilkan empat metrik, terdiri atas *accuracy, precision, recall*, dan *F1-score*. Konteks deteksi malware menempatkan *recall* kelas malware sebagai prioritas utama, karena false negative berisiko tinggi [5]. Evaluasi menekankan pencapaian nilai *recall* malware tinggi agar kemampuan deteksi ancaman optimal. Kelas malware ditetapkan sebagai kelas positif, sehingga recall mengukur proporsi malware terdeteksi dengan benar.

Table 4. Metrik evaluasi model

Metrik	Rumus	Keterangan
Accuracy	(TP+TN) / (TP+TN+FP+FN)	Proporsi prediksi benar terhadap seluruh data
Precision	TP / (TP+FP)	Kemampuan menghindari false positive
Recall	TP / (TP+FN)	Kemampuan mendeteksi seluruh malware
F1-Score	2 x P x R / (P+R)	Keseimbangan precision dan recall

Keterangan: TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative, P = Precision, R = Recall

Evaluasi dilakukan pada data uji sejumlah 4.518 sampel menghasilkan confusion matrix sebagai berikut:

Table 5. Confusion matrix hasil evaluasi offline

Prediksi	Malware	Aman
Malware	TP = 3709	FN = 1
Aman	FP = 4	TN = 804

Berdasarkan confusion matrix tersebut, diperoleh nilai metrik evaluasi sebagai berikut :

Table 6. Hasil evaluasi metrik model

Metrik	Nilai
Accuracy	99.89%
Precision	99,89%
Recall	99,97%
F1-Score	99,93%

3.4. Deployment

Model terlatih diterapkan dalam arsitektur *event-driven*. *Service background* berbasis library *watchdog* Python memonitor direktori target secara kontinu. Saat file PE terdeteksi, sistem otomatis mengekstrak fitur statis dan menjalankan prediksi *Random Forest*. Antarmuka web berperan sebagai *dashboard monitoring* untuk menampilkan *log*, *alert*, *confidence score*, dan *feature importance*. Untuk mencegah *data leakage* antara proses pelatihan dan pengujian, sampel dataset diorganisasi ke dalam dua partisi independen. Partisi pertama adalah *training pool* berjumlah 18.072 sampel yang digunakan untuk

melatih model. Partisi kedua adalah data uji offline berjumlah 4.518 sampel (20 persen dari total dataset), digunakan untuk evaluasi performa model secara otomatis melalui *confusion matrix* (lihat Tabel 4 dan Tabel 5, bagian Evaluation). Skema partisi ini memastikan bahwa performa model pada evaluasi offline mencerminkan kemampuan generalisasi terhadap data uji yang tidak dilihat model selama pelatihan, meskipun tetap berasal dari distribusi dataset yang sama.

Sebagai validasi tambahan di luar distribusi dataset pelatihan, penelitian ini juga melakukan pengujian real-time menggunakan sampel independen dari luar dataset, yaitu 100 sampel malware dari repositori theZoo dan 21 sampel benign pihak ketiga (lihat bagian Hasil, subbab 4.1). Pengujian ini bertujuan menilai generalisasi model terhadap sumber data yang sama sekali berbeda dari data pelatihan, bukan sekadar mengulang evaluasi pada distribusi yang sama. Pengujian *end-to-end* dilakukan pada *virtual machine* terisolasi (VirtualBox) tanpa koneksi internet dengan *snapshot baseline* sebagai titik pemulihan. Perlu ditegaskan bahwa keamanan pengujian pada penelitian ini bertumpu pada dua lapis. Pertama, metode deteksi bersifat *static analysis* murni tanpa pernah mengeksekusi berkas sampel sehingga *payload malware* tidak pernah aktif berjalan pada tahap klasifikasi. Kedua, proses penyalinan dan pencatatan hasil tetap dilakukan di dalam mesin virtual yang terisolasi dari jaringan dengan *snapshot baseline* yang dipulihkan setelah setiap sampel diproses sebagai lapisan *containment* tambahan untuk berjaga-jaga terhadap Risiko *human eror* misalnya sampel tidak sengaja dijalankan secara manual.

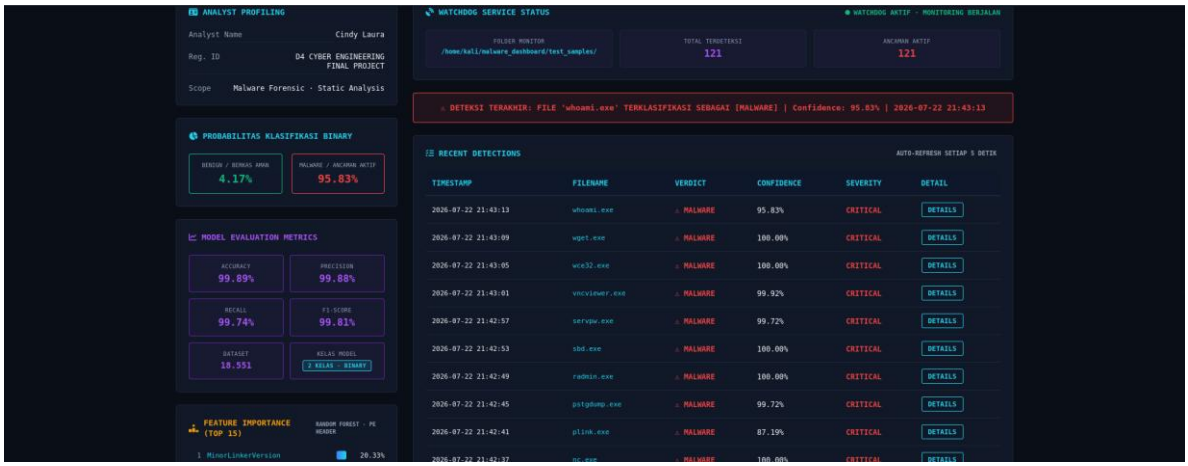
Dengan demikian, perlu ditegaskan bahwa isolasi VirtualBox pada penelitian ini berbeda secara fundamental dari behavioral sandbox seperti Cuckoo Sandbox atau CAPE Sandbox, yang secara sengaja mengeksekusi berkas malware untuk mengamati perilakunya (dynamic analysis). Isolasi VirtualBox dalam penelitian ini semata-mata berfungsi sebagai lapisan containment tambahan yang mendukung proses static analysis: berkas sampel tidak pernah dieksekusi, dan mesin virtual hanya berperan sebagai lingkungan aman untuk mengantisipasi risiko human error, misalnya apabila sampel tidak sengaja dijalankan secara manual di luar alur pipeline yang seharusnya.

Pengujian fungsional menggunakan minimal 30 sampai 50 sampel per kelas, sedangkan evaluasi statistik menggunakan *test set* terpisah berjumlah minimal 3.000 sampel dengan target *recall* minimal 95 persen per kelas malware. Realisasi evaluasi statistik mencapai 4.518 sampel data uji, melampaui target yang ditetapkan.

4. HASIL

4.1. Hasil pengujian fungsional Real-time

Sistem diuji pada lingkungan terisolasi dengan direktori monitoring aktif. Library *watchdog* mendeteksi setiap file PE baru, lalu sistem mengekstrak fitur statis dan menjalankan prediksi *Random Forest* secara otomatis. Pengujian dilakukan terhadap 121 sampel, terdiri atas 100 sampel malware dan 21 sampel benign. Setiap hasil tercatat pada log dengan informasi nama file, ukuran, status deteksi, *confidence score*, dan waktu eksekusi. Pengujian membuktikan alur *end-to-end* berjalan tanpa intervensi pengguna, mulai dari deteksi file sampai pencatatan log dan tampilan dashboard, sebagaimana didukung oleh catatan *audit trail* sistem. Tabel 7 menampilkan sebagian hasil deteksi real-time.



Gambar 2. Dashboard monitoring

Gambar 2 memperlihatkan antarmuka dashboard monitoring yang menampilkan status layanan watchdog, probabilitas klasifikasi biner, metrik evaluasi model, daftar feature importance teratas, serta log deteksi real-time secara berurutan berdasarkan waktu. Contoh pada gambar menunjukkan deteksi file whoami.exe yang diklasifikasikan sebagai malware dengan confidence 95,83 persen, konsisten dengan hasil pada Tabel 7 dan mengilustrasikan bagaimana sistem menyajikan informasi deteksi kepada pengguna secara langsung tanpa intervensi manual.

Table 7. Contoh hasil deteksi real-time

Nama File	Ukuran	Hasil	Confidence
whoami.exe	65,0 KB	Malware	95,83 persen
nc.exe	58,0 KB	Malware	100,00 persen
plink.exe	818,3 KB	Malware	87,19 persen
wget.exe	301,5 KB	Malware	100,00 persen
vncviewer.exe	356,0 KB	Malware	99,92 persen
Wc32.exe	4790,7 KB	Malware	100,00 persen

Tabel 7 menampilkan sebagian hasil deteksi real-time. Kelima sampel pertama pada tabel merupakan berkas benign pihak ketiga (whoami.exe, nc.exe, plink.exe, wget.exe, vncviewer.exe) yang seluruhnya salah diklasifikasikan sebagai malware dengan confidence tinggi, sedangkan Wc32.exe merupakan sampel malware yang terdeteksi benar.

Sampel malware yang digunakan dalam pengujian real-time diperoleh dari repositori akademik theZoo (github.com/ytisf/theZoo), sebuah koleksi malware yang dikurasi khusus untuk keperluan riset dan pendidikan keamanan siber. Sampel didistribusikan dalam format ZIP terenkripsi dengan password 'infected' sebagai standar keamanan. Dari repositori ini diperoleh 100 sampel file PE Windows yang valid, mencakup keluarga malware FannyWorm, CryptoLocker, WannaCry, ZeroAccess, Ransomware.Rex, dan Wiper.BEEP. Keaslian setiap sampel dapat diverifikasi melalui nilai SHA256 yang tercatat di audit trail sistem. Pemilihan sampel malware dari repositori *thezoo* tidak dilakukan secara acak murni, melainkan mengikuti 3 kriteria: pertama, berkas berformat *portable Executable* yang valid dan dapat di parsing oleh Pustaka *pefile*. Kedua, mewakili keluarga malware yang relevan dengan kelima label pada dataset pelatihan sehingga dipilih sampel dari keluarga *FannyWorm, CryptoLocker, WannaCry, ZeroAccess, Ransomware.* Ketiga, memiliki Riwayat verifikasi publik berupa hash *SHA256* yang tercatat sehingga keasliannya dapat dipertanggung jawabkan secara akademik. Seluruh pengujian

dilakukan di VM Kali Linux terisolasi tanpa mengeksekusi file — hanya header PE yang dibaca secara statis.

4.2. Hasil evaluasi metrik

Setelah subbab 4.1 memastikan alur end-to-end berjalan otomatis, subbab ini mengukur seberapa akurat hasil klasifikasi tersebut dibandingkan label ground truth. Evaluasi real-time dilakukan pada 121 sampel dengan label asli diketahui. *Confusion matrix* hasil pengujian:

Table 8. Confusion matrix real-time

Prediksi	Malware	Aman
Malware	TP = 100	FN = 0
Aman	FP = 21	TN = 0

Table 9. Hasil evaluasi metrik real-time

Metrik	Nilai
Accuracy	82,64%
Precision	82,64%
Recall	100%
F1-Score	90,50%

Nilai Accuracy dan Precision yang identik (82,64%) bukan kekeliruan penulisan, melainkan konsekuensi matematis dari TN=0 dan FN=0: pada kondisi tersebut, Accuracy = TP/(TP+FP) yang secara aljabar setara dengan Precision.

4.3. Interpretasi awal hasil

Recall mencapai 100 persen, sehingga sistem mendeteksi seluruh 100 sampel malware tanpa terlewat, selaras dengan prioritas deteksi malware karena *false negative* berisiko tinggi [6]. Namun, pada 21 sampel benign yang di uji seluruhnya salah di klasifikasikan sebagai malware(*specificity* 0 persen pada sampel ini) menghasilkan *precision* 82,64 persen dan F1-score 90,50 persen. Mengingat jumlah sampel benign yang terbatas. Analisis log menunjukkan seluruh *false positive* berasal dari perkakas administrasi dan *dual-use* seperti nc.exe, plink.exe, wget.exe, vncviewer.exe, dan sejenisnya, serta utilitas standar Windows (whoami.exe). Perkakas tersebut memiliki karakteristik struktur PE mirip malware, misalnya pemakaian fungsi jaringan dan kompresi, sehingga sulit dibedakan secara statis.

Penyebab utama diduga keterbatasan data benign pada pelatihan: kelas benign pada data latih hanya berasal dari file .dll System32, sehingga model tidak mengenali variasi file benign lain seperti perkakas jaringan dan instaler pihak ketiga. Temuan ini berbeda dari penelitian Alvanof dan Dinata dengan precision tinggi pada satu jenis malware [7], dan Matin dkk. dengan akurasi 96,9 persen pada dataset relatif seimbang [8]; perbedaan muncul karena cakupan benign penelitian ini lebih sempit. Arah perbaikan mencakup penambahan data benign bervariasi, penyetelan ulang ambang keputusan, dan penerapan teknik penyeimbang kelas seperti *random under-sampling* [9].

4.4. Perbandingan dengan penelitian terdahulu

Untuk menempatkan hasil pada subbab 4.2 dan 4.3 dalam konteks yang lebih luas, subbab ini membandingkan performa dan luaran sistem dengan penelitian sejenis. Tabel 10 membandingkan penelitian ini dengan penelitian terdahulu. Penelitian Matin dkk [3]. dan Alvanof serta Dinata [4] mencapai akurasi tinggi, namun keduanya berhenti pada model atau aplikasi unggah on-demand. Penelitian ini menghasilkan produk dashboard real-time berbasis *event-driven* dengan cakupan lima keluarga malware ditambah kelas benign. Nilai metrik real-time penelitian ini diukur dari 121 sampel (100 malware, 21 benign).

Table 10. Perbandingan dengan penelitian terdahulu

Aspek	Matin dkk. [3]	Alvanof & Dinata [4]	Penelitian ini
Algoritma	Ensemble (Hist Gradient Boosting)	Random Forest	Random Forest
Cakupan malware	Deteksi malware PE	Ransomware	Lima keluarga malware + benign
Performa	Akurasi 96,9 persen	Akurasi 98,79 persen	Recall 100 persen, Precision 82,64 persen, Specificity 0 persen (n=121)
Bentuk keluaran	Model evaluasi	Aplikasi web Streamlit (on-demand)	Dashboard real-time event-driven

4.5. Bias dataset dan interpretasi akurasi

Kesenjangan antara akurasi offline yang tinggi (subbab 3.4) dan precision real-time yang lebih rendah (subbab 4.2–4.3) mendorong penelusuran lebih lanjut terhadap kemungkinan bias pada dataset pelatihan. Pada evaluasi offline dengan test set sebesar 4.518 sampel model mencapai akurasi 99,89 persen. Angka tinggi ini perlu dimaknai secara kritis. Komposisi data menempatkan 18.551 sampel malware dari beragam sumber melawan 4.039 sampel benign yang seluruhnya berasal dari direktori System32 Windows. Akibatnya, model cenderung belajar membedakan file buatan Microsoft dengan toolchain resmi dari file malware berbagai sumber, bukan mengenali karakteristik malware secara langsung.

Indikasi bias terlihat pada *feature importance* [15], dengan fitur dominan berupa MinorLinkerVersion(0,203), MajorImageVersion (0,129) , dan MajorOperatingSystemVersion(0,126). Ketiga fitur tersebut adalah informasi versi linker dan sistem operasi pada proses kompilasi, bukan ciri perilaku berbahaya. Dominasi fitur versi menandakan model menangkap artefak penyusunan dataset, bukan sinyal keamanan sebenarnya. Pengujian real-time memperkuat dugaan ini: pada pengujian 121 sampel, precision tercatat 82,64 persen namun specificity 0 persen, karena seluruh 21 sampel benign pihak ketiga di luar System32 tetap salah terklasifikasi sebagai malware. Mengingat kecilnya ukuran sampel benign uji angka *specificity* 0 persen ini bersifat indikatif dan memerlukan validasi lanjutan pada sampel benign yang lebih besar dan lebih beragam sebelum dapat digeneralisasi sebagai karakteristik permanen sistem.

4.6. Keterbatasan penelitian

Temuan bias pada sub bab 4.5 menegaskan perlunya membaca hasil penelitian ini secara proporsional, dengan mempertimbangkan sejumlah keterbatasan metodologis berikut. Penelitian ini memiliki beberapa keterbatasan. Pertama, dataset benign pelatihan bersifat homogen karena seluruhnya berasal dari System32, sehingga model bias terhadap asal-usul *build* file; jumlah sampel benign juga jauh lebih kecil dibanding malware. Kedua, pengujian real-time menggunakan 121 sampel (100 malware, 21 benign) — melampaui target minimal untuk kelas malware (30–50 sampel), namun jumlah sampel benign (21) masih di bawah target tersebut, dan justru kelas inilah yang menjadi sumber seluruh kesalahan klasifikasi (*specificity* 0 persen). Ketiga, enam dari 21 sampel benign pada pengujian real-time merupakan sampel yang sama dengan pengujian awal berskala kecil, sehingga variasi sampel benign yang benar-benar baru lebih terbatas dari jumlah total yang terlihat. Keempat, file malware mentah sulit diperoleh karena keterbatasan legal, etika, dan akses repositori terverifikasi, sehingga keterbatasan sampel menjadi temuan metodologis, bukan kendala waktu. Kelima, terdapat potensi *feature mismatch* antara sampel pengujian real-time dan data pelatihan yang belum diverifikasi secara statistik. Keenam, isolasi berbasis *VirtualBox* pada penelitian ini berfungsi sebagai lapisan *containment* untuk *static analysis* bukan sebagai *sandbox* analisis perilaku yang tahan terhadap Teknik *anti-sandbox*.

Keterbatasan ini menjadi relevan apabila penelitian ini diperluas kearah *dynamic analysis* pada penelitian selanjutnya.Sampel malware pada pengujian real-time diperoleh dari repositori theZoo dan mencakup keluarga seperti WannaCry, ZeroAccess, dan Wiper.BEEP, yang tidak selalu berkorespondensi langsung dengan lima label keluarga pada data pelatihan (Generic Malware, Spyware, Downloader, Ransomware, Backdoor). Kesamaan distribusi fitur header PE antara sampel theZoo dan data latih belum diuji secara statistik, sehingga terdapat kemungkinan model belajar membedakan berkas berdasarkan asal-usul kompilasi atau *toolchain* alih-alih pola perilaku malware yang sebenarnya. Dengan demikian, nilai recall 100 persen pada pengujian theZoo perlu dimaknai secara hati-hati bukan mustahil model cenderung menandai berkas apa pun di luar ekosistem System32 sebagai malware, terlepas dari apakah berkas tersebut benar-benar berbahaya. Verifikasi kesamaan *feature space* antara sumber data pelatihan dan data pengujian menjadi arah penting bagi penelitian lanjutan, guna memastikan bahwa performa deteksi yang dilaporkan mencerminkan kemampuan generalisasi yang valid, bukan artefak perbedaan sumber data.

Seluruh pengujian berjalan pada virtual machine Kali terisolasi, dan perhitungan metrik dilakukan otomatis dari berkas CSV. Atas dasar keterbatasan tersebut, sistem ini diposisikan sebagai *proof of concept*, bukan sistem siap pakai pada lingkungan *production.*

5. PEMBAHASAN

Recall mencapai 100 persen sedangkan precision hanya 82,64 persen, menunjukkan sistem kuat mendeteksi malware (*threat coverage*) namun masih lemah membedakan file benign (*discriminative power*), karena model cenderung menandai karakteristik struktural PE yang tidak lazim sebagai ancaman, termasuk pada perkakas administratif yang sah [3]. Mayoritas *false positive* berasal dari perkakas *dual-use* seperti nc.exe, plink.exe, wget.exe, dan vncviewer.exe, bahkan whoami.exe, karena struktur PE-nya menyerupai malware. Akar masalahnya adalah data benign pelatihan yang homogen, hanya dari direktori System32, sehingga model tidak mengenali variasi file benign dari luar ekosistem Microsoft. Akurasi offline 99,89 persen juga tidak mencerminkan kemampuan deteksi sebenarnya. Analisis *feature importance* [15] menunjukkan fitur dominan adalah metadata versi *toolchain* (MinorLinkerVersion, MajorImageVersion, MajorOperatingSystemVersion), bukan indikator perilaku berbahaya, sehingga model diduga belajar membedakan asal-usul *build* file, bukan karakteristik malware itu sendiri. Kesenjangan tajam antara akurasi offline dan precision real-time menegaskan bahwa akurasi tinggi pada data uji sejenis data latih tidak menjamin generalisasi di dunia nyata.

Dibandingkan Matin dkk. dengan akurasi 96,9 persen [3] dan Alvanof serta Dinata dengan akurasi 98,79 persen **[**4], nilai precision penelitian ini tampak lebih rendah, namun hal ini disebabkan oleh perbedaan desain evaluasi: penelitian terdahulu hanya diuji secara offline pada distribusi yang sama dengan data latih, sedangkan penelitian ini turut diuji secara real-time dengan sampel benign di luar distribusi pelatihan. Berbeda dari Zulaeha dkk. yang bersifat kajian literatur tanpa model konkret [1] dan Adiyatma serta Setiawan yang rentan terhadap *sandbox evasion* [2], penelitian ini menghasilkan sistem *event-driven* yang terintegrasi dengan dashboard dan berjalan otomatis tanpa intervensi pengguna. Precision yang belum optimal pada pengujian *real-time* berisiko menimbulkan *alert fatigue* pada pengguna, sehingga diversifikasi data benign dan penyesuaian ambang keputusan menjadi prasyarat sebelum sistem layak diterapkan pada lingkungan produksi. Meski demikian, arsitektur *event-driven* yang sudah mendukung otomatisasi penuh tetap menjadi nilai tambah dibandingkan pendekatan unggah *on-demand* pada penelitian pembanding.

Pengujian real-time pada penelitian ini baru menggunakan 121 sampel *(100 malware, 21 benign)*. Untuk kelas malware jumlah ini melampaui target fungsional minimal 30-50 sampel per kelas, namun untuk kelas *benign* jumlah 21 sampel masih jauh dari target tersebut dan enam diantaranya merupakan sampel yang sama dengan pengujian awal berskala kecil sehingga variasi sampel *benign* yang benar-benar baru lebih terbatas dari jumlah total yang terlihat. Oleh karena itu, nilai *specifciity* 0 persen dan *precision* 82,64 persen pada pengujian ini sebaiknya dimaknai sebagai temuan awal yang

mengidentifikasikan arah masalah bukan sebagai generalisasi definitive atas kemampuan sistem membedakan seluruh populasi file *benign*.

6. KESIMPULAN

Penelitian ini berhasil merancang dan membangun sistem deteksi malware berbasis Random Forest pada file Portable Executable Windows yang terintegrasi dengan dashboard web dan pemantauan direktori secara *event-driven* menggunakan library watchdog dan pefile, sehingga seluruh alur mulai dari deteksi file, ekstraksi fitur, klasifikasi, hingga penyajian hasil berjalan otomatis tanpa intervensi pengguna. Pengujian real-time terhadap 121 sampel, terdiri atas 100 sampel malware dan 21 sampel benign, pada lingkungan virtual machine Kali Linux terisolasi menghasilkan Recall 100%, Precision 82,64%, Accuracy 82,64%, dan F1-Score 90,50%. Nilai Recall yang sempurna membuktikan sistem mampu mengenali seluruh keluarga malware yang diujikan, tanpa satupun ancaman yang lolos, sehingga tujuan utama penelitian sebagai sistem deteksi yang dapat diandalkan mendeteksi ancaman tercapai.

Meskipun demikian, pada pengujian real-time dengan 21 sampel benign seluruhnya salah diklasifikasikan sebagai malware. Mengingat ukuran sampel yang masih terbatas dan Sebagian di antaranya bukan sampel baru, temuan ini merupakan indikasi awal bahwa kemampuan sistem membedakan file aman masih perlu diperbaiki bukan Kesimpulan definitif yang berlaku umum untuk seluruh populasi file benign. Analisis lebih lanjut mengidentifikasi tiga faktor penyebab: bias dataset benign yang homogen karena seluruhnya berasal dari direktori System32 Windows, dominasi fitur metadata versi kompilasi seperti MinorLinkerVersion, MajorImageVersion, dan MajorOperatingSystemVersion pada feature importance yang mengindikasikan model mempelajari pola *spurious correlation* antara asal-usul *build* file dan bukan karakteristik perilaku berbahaya, serta potensi *over-correction* akibat penerapan SMOTE dan class_weight='balanced' secara bersamaan. Dengan demikian, sistem ini diposisikan sebagai *proof of concept* yang membuktikan kelayakan arsitektur deteksi malware berbasis Random Forest yang terintegrasi dan otomatis, namun belum siap diterapkan pada lingkungan produksi tanpa perbaikan lebih lanjut pada kualitas dan keberagaman data benign.

Penelitian selanjutnya disarankan untuk mendiversifikasi dataset benign dari minimal empat sumber berbeda, mencakup sistem operasi, aplikasi komersial, tools administrasi, dan installer pihak ketiga, guna menghindari bias kompilasi yang teridentifikasi pada penelitian ini. Secara khusus, sampel benign uji sebaiknya turut mencakup berkas *executable* umum bawaan Windows (misalnya notepad.exe, explorer.exe, calc.exe) serta perangkat lunak populer pihak ketiga, bukan hanya perkakas administratif bergaya Kali Linux (nc.exe, plink.exe, wget.exe, vncviewer.exe) yang secara struktural memang menyerupai malware karena karakteristik jaringan dan kompresinya; pemilihan sampel benign uji yang kurang representatif inilah yang diduga menjadi penyebab utama specificity 0 persen pada pengujian real-time penelitian ini. Evaluasi feature importance juga perlu memverifikasi relevansi keamanan dari fitur yang dominan, bukan sekadar relevansi statistik, serta pemilihan satu mekanisme penyeimbangan kelas guna menghindari risiko over-correction. Sebagai arah pengembangan lanjutan, penelitian berikutnya juga dapat membandingkan secara langsung sampel *legitimate* dari suatu utilitas (misalnya whoami.exe versi asli Windows) dengan varian malware yang menyamar menggunakan nama serupa, guna menguji apakah model mampu membedakan keduanya berdasarkan karakteristik struktural PE yang sebenarnya, bukan sekadar nama atau ukuran berkas. Langkah-langkah tersebut diharapkan dapat meningkatkan precision sistem tanpa mengorbankan recall yang telah tercapai, sehingga sistem deteksi malware berbasis Random Forest ini dapat berkembang dari tahap proof of concept menuju sistem yang lebih andal untuk diterapkan pada lingkungan nyata..

REFERENCES

[1] Z. Zulaeha, W. Z. Mulqiya, and E. Rilvani, "Peningkatan Perlindungan pada Sistem Operasi Windows terhadap Gangguan Malware," Jurnal Sistem Informasi dan Ilmu Komputer, vol. 2, no. 4, pp. 165–177, 2024, doi: 10.59581/jusiik-widyakarya.v3i1.4440.

[2] M. R. Adiyatma and H. Setiawan, "Analisis Sampel Malware pada Sistem Operasi Windows 10 Menggunakan Cuckoo Sandbox," Jurnal Informatika dan Teknik Elektro Terapan, vol. 13, no. 3, 2025, doi: 10.23960/jitet.v13i3.6556.

[3] I. M. M. Matin, M. Agustin, B. Sugiarto, and A. N. Asri, "Deteksi Malware Menggunakan Machine Learning dengan Metode Ensemble," Prosiding Sains Nasional dan Teknologi, vol. 13, no. 1, pp. 265–270, Nov. 2023, doi: 10.36499/psnst.v13i1.9224.

[4] M. Mahendra Alvanof, Bustami, and R. Kesuma Dinata, "Penerapan Algoritma Random Forest dalam Deteksi dan Klasifikasi Ransomware," Jurnal Elektronika dan Teknologi Informasi, vol. 5, no. 2, pp. 23–31, Sep. 2024.

[5] A. Nugroho and C. Umam, "Comparison of Random Forest and Neural Network for Portable Executable Malware Classification," Indonesian Journal of Innovation Studies, vol. 27, no. 1, Jan. 2026, doi: 10.21070/ijins.v27i1.1883.

[6] F. A. Rafrastara, R. A. Pramunendar, D. P. Prabowo, E. Kartikadarma, and U. Sudibyo, "Optimasi Algoritma Random Forest menggunakan Principal Component Analysis untuk Deteksi Malware," Jurnal Teknologi dan Sistem Informasi Bisnis, vol. 5, no. 3, pp. 217–223, Jul. 2023, doi: 10.47233/jteksis.v5i3.854.

[7] R. R. Sani, F. A. Rafrastara, and W. Ghozi, "Integrating Ensemble Learning and Information Gain for Malware Detection based on Static and Dynamic Features," Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, vol. 10, no. 1, Feb. 2025, doi: 10.22219/kinetik.v10i1.2051.

[8] N. H. Putri, "Deteksi dan Prioritas Mitigasi Malware Menggunakan Random Forest dan Metode MCDM," Jurnal Informatika dan Teknik Elektro Terapan, vol. 13, no. 3S1, Oct. 2025, doi: 10.23960/jitet.v13i3S1.7762.

[9] D. S. Bhayangkara, H. D. Putranto, F. Toriq, and F. Wijayanto, "Analisis Static Malware Menggunakan Algoritma Random Forest Machine Learning," Jurnal Teknologi Informasi, vol. 9, no. 2, pp. 172–176, Dec. 2023.

[10] M. R. Adiyatma and H. Setiawan, "Analisis Sampel Malware pada Sistem Operasi Windows 10 Menggunakan Cuckoo Sandbox," Jurnal Informatika dan Teknik Elektro Terapan, vol. 13, no. 3, 2025, doi: 10.23960/jitet.v13i3.6556.

[11] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A systematic literature review on Windows malware detection: Techniques, research issues, and future directions," Journal of Systems and Software, vol. 209, Mar. 2024, doi: 10.1016/j.jss.2023.111921.

[12] M. I. Yousuf, I. Anwer, T. Shakir, M. Siddiqui, and M. Shahid, "Multi-feature Dataset for Windows PE Malware Classification," arXiv preprint arXiv:2210.16285, Oct. 2022, doi: 10.48550/arXiv.2210.16285.

[13] C. K. Yuk and C. J. Seo, "Static Analysis and Machine Learning-based Malware Detection System using PE Header Feature Values," International Journal of Innovative Research and Scientific Studies, vol. 5, no. 4, pp. 281-288, 2022.

[14] M. I. Yousuf, I. Anwer, T. Shakir, M. Siddiqui, and M. Shahid, "Windows Malware Detection Based on Static Analysis with Multiple Features,". PMCID: PMC10280383.

[15] Barnes, Gabryella, Ghafarian, Ahmad, "Machine Learning-Based Static Ransomware Detection Using PE Header Features and SHAP Interpretation," Journal of Cybersecurity and Privacy, vol. 6, no. 2, art. 58, 2026, doi: 10.3390/jcp6020058.