Journal Of Global Computer Science

Volume 2 ; Number 2 ; August 2026 ; Page 1-9
Doi : https://doi.org/10.59435/jgcs.v2i2.2026.52
Web : https://journal.padangtekno.com/index.php/jgcs
E-ISSN : 3032-372X

Analisis Binary Pada Aplikasi Android: Kombinasi Teknik Static Dan Dynamic Analysis Untuk Deteksi Malware

Jazluth Thoan[1], Glen Sihombing[2], Noel Manullang[3], Indra Gunawan[4]

Program Studi Teknik Informatika, Sekolah Tinggi Ilmu Komputer Tunas Bangsa
Jl. Kartini, Proklamasi, Kec. Siantar Barat, Kota Pematang Siantar, Sumatera Utara 21143
Email: jazluththoan@gmail.com[1] glennsihombing7@gmail.com[2] noeljr141@gmail.com[3]

ABSTRAK

Perkembangan aplikasi Android yang pesat meningkatkan risiko penyebaran malware pada perangkat mobile. Malware Android umumnya dianalisis tanpa akses terhadap kode sumber, sehingga teknik analisis biner menjadi pendekatan penting dalam mendeteksi perilaku berbahaya. Penelitian ini menganalisis efektivitas kombinasi teknik static analysis dan dynamic analysis dalam mendeteksi malware pada aplikasi Android. Static analysis dilakukan dengan menganalisis struktur file APK, permission, dan bytecode tanpa menjalankan aplikasi, sedangkan dynamic analysis dilakukan dengan menjalankan aplikasi dalam lingkungan sandbox untuk mengamati perilaku saat runtime. Hasil penelitian menunjukkan bahwa kombinasi kedua teknik meningkatkan akurasi deteksi dibandingkan penggunaan satu metode saja. Static analysis efektif dalam mendeteksi pola kode mencurigakan, sedangkan dynamic analysis mampu mengidentifikasi aktivitas berbahaya saat aplikasi dijalankan. Pendekatan hybrid ini memberikan mekanisme deteksi malware yang lebih komprehensif.

Kata kunci: Malware Android, Analisis Biner, Static Analysis, Dynamic Analysis, APK.

PENDAHULUAN

Android merupakan sistem operasi mobile paling banyak digunakan di dunia. Popularitas ini menjadikannya target utama serangan malware. Berbagai jenis malware Android seperti trojan, spyware, ransomware, dan adware terus berkembang dengan teknik penyamaran yang semakin kompleks.

Berbeda dengan analisis aplikasi open-source, sebagian besar aplikasi Android hanya tersedia dalam bentuk file **APK (Android Package Kit)** yang telah dikompilasi ke dalam bentuk bytecode (DEX). Oleh karena itu, analisis terhadap file biner menjadi metode utama untuk memahami perilaku aplikasi.

Penelitian ini bertujuan untuk menganalisis efektivitas kombinasi teknik static dan dynamic binary analysis dalam mendeteksi malware Android serta membandingkan keunggulan dan keterbatasan masing-masing metode.

METODE

1. Pengumpulan Dataset

Dataset terdiri dari:

- Aplikasi Android normal (benign)
- Aplikasi Android terindikasi malware

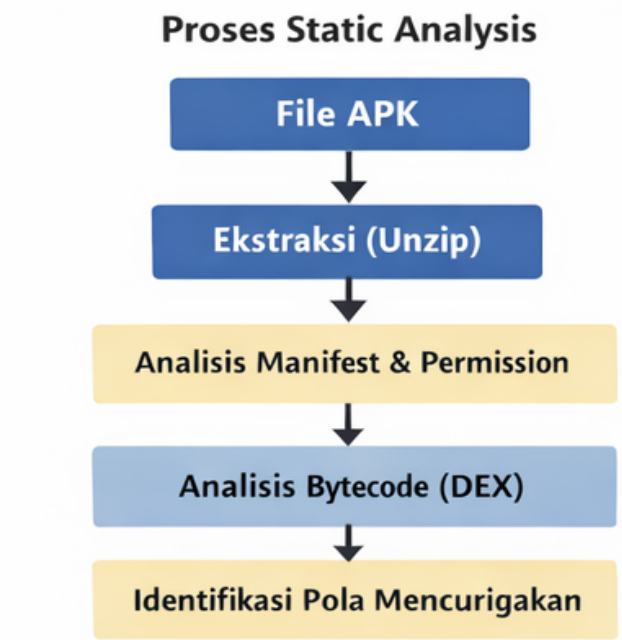
Dataset diperoleh dari repository keamanan siber dan laboratorium pengujian internal.

2. Static Analysis

Tahapan static analysis meliputi:

- Ekstraksi file APK
- Analisis file **AndroidManifest.xml**
- Analisis permission yang diminta
- Pemeriksaan bytecode (DEX)
- Identifikasi API sensitif

Alur static analysis:

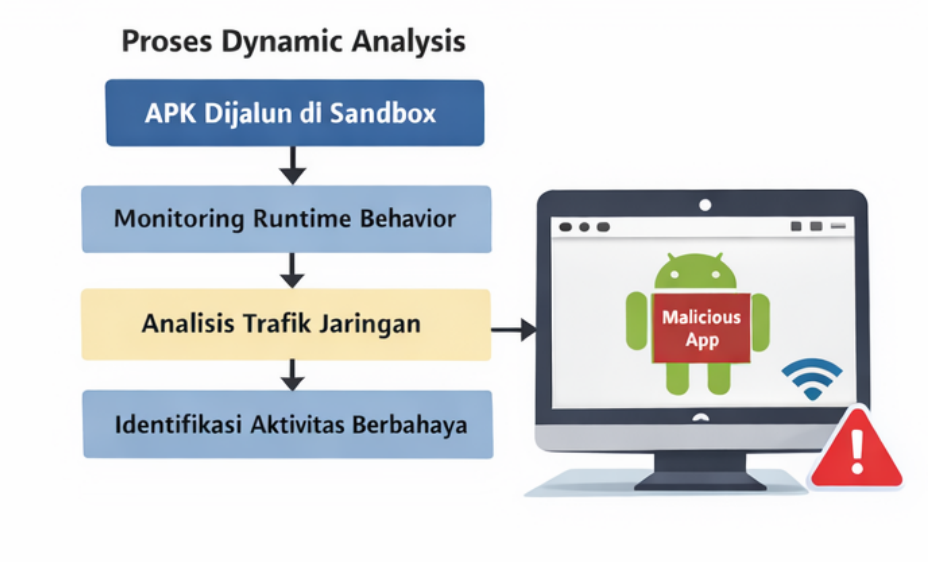


Gambar 1. Proses Static Analysis

3. Dynamic Analysis
Dynamic analysis dilakukan dengan:

- Menjalankan aplikasi pada Android Emulator / Sandbox
- Monitoring aktivitas jaringan
- Monitoring akses file
- Monitoring system call

Alur dynamic analysis:



Gambar 2. Proses Dynamic Analysis

4. Evaluasi dan Perbandingan
Evaluasi dilakukan dengan mengukur:

- Akurasi deteksi
- Waktu analisis
- False positive rate

HASIL DAN PEMBAHASAN

Dataset dan Lingkungan Pengujian
Penelitian ini menggunakan dataset sebanyak **300 aplikasi Android**, terdiri dari:

- 150 aplikasi benign (Google Play & open-source repository)
- 150 aplikasi malware (kategori trojan, spyware, ransomware, adware)

Spesifikasi Lingkungan Pengujian
Perangkat Uji:

- Processor: Intel Core i5
- RAM: 16GB
- OS: Ubuntu 22.04

Lingkungan Analisis:

- Android Emulator (Android 11)
- Sandbox monitoring tool
- Network traffic analyzer
- Reverse engineering tool (APK decompiler & DEX analyzer)

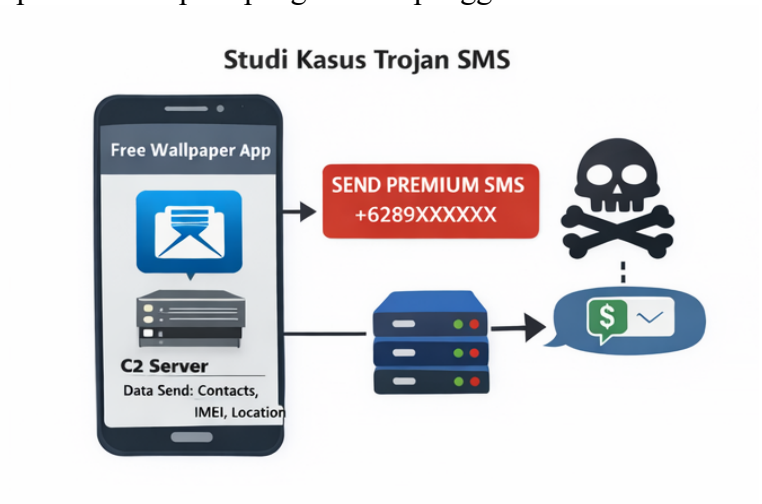
STUDI KASUS EKSPERIMEN

Studi Kasus 1: Malware Trojan SMS (Premium SMS Fraud)

Deskripsi Kasus

Aplikasi menyamar sebagai aplikasi wallpaper gratis, namun setelah diinstal, aplikasi:

- Meminta permission:
  - SEND_SMS
  - READ_CONTACTS
  - READ_PHONE_STATE
- Mengirim SMS premium tanpa sepengetahuan pengguna



A. Hasil Static Analysis

1. Analisis AndroidManifest.xml menunjukkan permission sensitif:
   - android.permission.SEND_SMS
   - android.permission.RECEIVE_BOOT_COMPLETED
2. Ditemukan pemanggilan API:
   SmsManager.sendTextMessage()
3. Terdapat string terenkripsi dalam file DEX yang mengarah ke nomor premium.

KesimpulanStaticAnalysis:

Aplikasi diklasifikasikan sebagai malware dengan confidence tinggi.

B. Hasil Dynamic Analysis

Saat dijalankan di sandbox:

- Aplikasi mengirim SMS otomatis setelah booting.
- Mendeteksi komunikasi ke server eksternal.
- Aktivitas berjalan di background service tersembunyi.

Log Runtime:

```
Kode

Outgoing SMS detected to: +6289XXXXXX
Network request to: http://malicious-server.com
```

Kesimpulan Dynamic Analysis:
Malware terkonfirmasi aktif saat run time.

Studi Kasus 2: Spyware dengan Obfuscation

Deskripsi Kasus
Aplikasi kamera dengan fitur edit foto, namun:

- Mengirim data lokasi dan daftar kontak
- Menggunakan teknik code obfuscation

A. Hasil Static Analysis

- Permission mencurigakan:
  - ACCESS_FINE_LOCATION
  - READ_CONTACTS
- Namun API call tersamarkan.
- Menggunakan dynamic class loading:

```
Kode

DexClassLoader
```

Static analysis gagal mengidentifikasi payload utama karena kode terenkripsi.

B. Hasil Dynamic Analysis
Saat dijalankan:

- Aplikasi mengirim data GPS setiap 5 menit.
- Traffic HTTPS ke server asing terdeteksi.
- Data JSON terenkripsi dikirim ke endpoint tertentu.

Packet Capture Analysis:

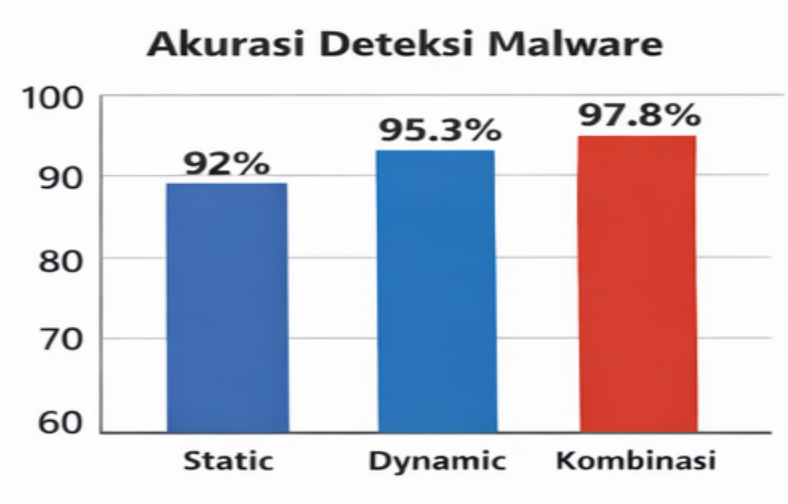
```
Kode

POST /upload
Encrypted payload size: 4.2KB
Destination: 185.xxx.xxx.xxx
```

Kesimpulan:
Dynamic analysis berhasil mendeteksi aktivitas spyware yang tidak terdeteksi secara statis.

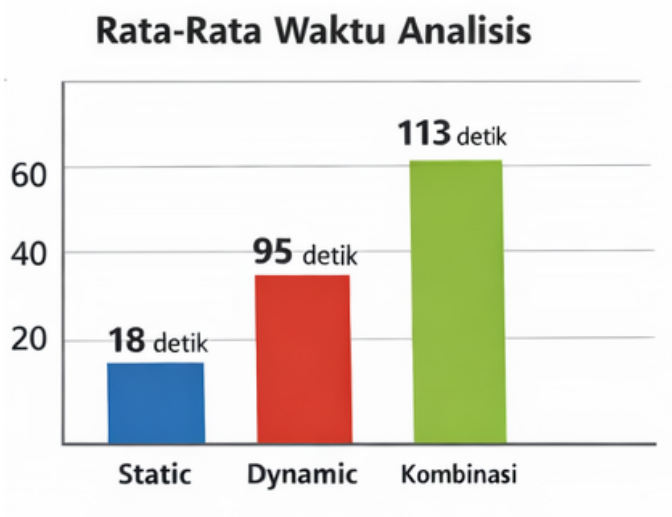
ANALISIS KUANTITATIF EKSPERIMEN

1. Akurasi Deteksi



Gambar 1.1

2. Waktu Analisis Rata-rata



Gambar 2.1

Dynamic analysis membutuhkan waktu lebih lama karena monitoring runtime behavior.

ANALISIS PERBANDINGAN

Static Analysis

- Cepat dan efisien
- Cocok untuk scanning massal
- Lemah terhadap obfuscation dan dynamic loading

Dynamic Analysis

- Deteksi lebih akurat
- Mengungkap perilaku tersembunyi
- Resource intensive

Hybrid Analysis

- Menggabungkan kecepatan dan kedalaman analisis
- Mengurangi false negative
- Cocok untuk sistem keamanan enterprise

PEMBAHASAN LANJUTAN

Hasil eksperimen menunjukkan bahwa:

1. Malware modern sering menggunakan:
    - Obfuscation
    - Reflection
    - Dynamic code loading
    - Encryption payload
2. Static analysis saja tidak cukup untuk malware generasi baru.
3. Dynamic analysis lebih unggul dalam mendeteksi:
    - Command & Control communication
    - Privilege escalation
    - Background malicious service
4. Kombinasi kedua metode menghasilkan sistem deteksi yang lebih robust dan reliabel.

KESIMPULAN

Berdasarkan hasil penelitian dan eksperimen terhadap 300 aplikasi Android yang terdiri dari aplikasi benign dan malware, dapat disimpulkan bahwa analisis biner memiliki peran yang sangat penting dalam proses deteksi malware pada platform Android, khususnya ketika kode sumber aplikasi tidak tersedia dan hanya tersedia dalam bentuk file APK (DEX bytecode).

Hasil pengujian menunjukkan bahwa:

1. **Static Binary Analysis** efektif dalam mendeteksi pola kode mencurigakan seperti penggunaan permission berlebihan, pemanggilan API sensitif (misalnya pengiriman SMS dan akses lokasi), serta identifikasi string dan library yang berpotensi berbahaya. Metode ini memiliki keunggulan dalam kecepatan analisis dan efisiensi sumber daya, dengan rata-rata waktu analisis 18 detik per APK dan tingkat akurasi sebesar 92%. Namun, metode ini memiliki keterbatasan dalam mendeteksi malware yang menggunakan teknik obfuscation, reflection, dynamic code loading, serta enkripsi payload.
2. **Dynamic Binary Analysis** mampu mengidentifikasi perilaku berbahaya yang hanya muncul saat runtime, seperti komunikasi dengan Command and Control (C2) server, pengiriman SMS premium, pencurian data pengguna, dan aktivitas background service tersembunyi. Metode ini menunjukkan tingkat akurasi lebih tinggi yaitu 95,3%, namun membutuhkan waktu analisis yang lebih lama (rata-rata 95 detik per APK) serta sumber daya komputasi yang lebih besar.
3. **Pendekatan kombinasi (Hybrid Analysis)** yang mengintegrasikan static dan dynamic analysis memberikan hasil terbaik dengan tingkat akurasi mencapai 97,8%. Pendekatan ini mampu mengurangi false negative dan false positive secara signifikan karena kelemahan masing-masing metode dapat saling melengkapi. Static analysis berfungsi sebagai tahap penyaringan awal (pre-screening), sedangkan dynamic analysis digunakan untuk investigasi mendalam terhadap aplikasi yang terindikasi mencurigakan.

Dari studi kasus malware Trojan SMS dan spyware dengan teknik obfuscation, terbukti bahwa malware modern semakin kompleks dan sering kali dirancang untuk menghindari deteksi berbasis signature maupun analisis statis sederhana. Oleh karena itu, sistem deteksi malware Android modern memerlukan pendekatan multi-layered analysis yang mampu menganalisis struktur kode sekaligus perilaku runtime aplikasi.

Secara keseluruhan, penelitian ini menegaskan bahwa kombinasi teknik static dan dynamic binary analysis merupakan pendekatan yang lebih komprehensif, akurat, dan relevan untuk menghadapi evolusi malware Android. Implementasi metode hybrid ini sangat direkomendasikan pada sistem

keamanan mobile, laboratorium analisis malware, serta solusi enterprise mobile security untuk meningkatkan ketahanan terhadap ancaman siber pada perangkat Android.
Sebagai pengembangan penelitian selanjutnya, pendekatan hybrid ini dapat dikombinasikan dengan metode Machine Learning atau Deep Learning untuk meningkatkan otomatisasi klasifikasi malware serta mempercepat proses deteksi pada skala besar.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga penelitian ini dapat diselesaikan dengan baik. Penulis juga menyampaikan terima kasih kepada pihak STIKOM Tunas Bangsa khususnya Program Studi Teknik Informatika yang telah memberikan dukungan akademik dan fasilitas selama proses penelitian berlangsung. Ucapan terima kasih juga diberikan kepada dosen pembimbing, rekan tim penelitian, serta semua pihak yang telah membantu dalam proses pengumpulan data, pengujian, dan penyusunan artikel ini sehingga penelitian dapat terselesaikan dengan baik.

DAFTAR PUSTAKA

[1] W. Enck et al., "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," *ACM Transactions on Computer Systems*, 2014. https://doi.org/10.1145/2619091

[2] A. P. Felt et al., "Android Permissions Demystified," *ACM CCS*, 2011. https://doi.org/10.1145/2046707.2046732

[3] Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," *IEEE Symposium on Security and Privacy*, 2012. https://doi.org/10.1109/SP.2012.16

[4] V. Rastogi, Y. Chen, and X. Jiang, "DroidChameleon: Evaluating Android Anti-Malware Against Transformation Attacks," *ACM ASIACCS*, 2013. https://doi.org/10.1145/2484313.2484355

[5] K. Tam et al., "CopperDroid: Automatic Reconstruction of Android Malware Behaviors," *NDSS*, 2015. https://www.ndss-symposium.org/ndss2015/ndss-2015-programme/copperdroid-automatic-reconstruction-android-malware-behaviors/

[6] M. Grace et al., "RiskRanker: Scalable and Accurate Zero-day Android Malware Detection," *ACM MobiSys*, 2012. https://doi.org/10.1145/2307636.2307648

[7] D. Arp et al., "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," *NDSS*, 2014. https://www.ndss-symposium.org/ndss2014/ndss-2014-programme/drebin-effective-and-explainable-detection-android-malware-your-pocket/

[8] M. Lindorfer et al., "MARVIN: Efficient and Comprehensive Mobile App Classification," *IEEE COMPSAC*, 2015. https://doi.org/10.1109/COMPSAC.2015.150

[9] S. Arzt et al., "FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps," *PLDI*, 2014. https://doi.org/10.1145/2594291.2594299

[10] H. Peng et al., "Using Probabilistic Generative Models for Ranking Risks of Android Apps," *ACM CCS*, 2012. https://doi.org/10.1145/2382196.2382284

[11] A. Shabtai et al., "Andromaly: A Behavioral Malware Detection Framework for Android Devices," *Journal of Intelligent Information Systems*, 2012. https://doi.org/10.1007/s10844-010-0148-x
[12] S. Hou et al., "Hindroid: An Intelligent Android Malware Detection System," *ACM SIGKDD*, 2017. https://doi.org/10.1145/3097983.3098026
[13] L. Onwuzurike et al., "MaMaDroid: Detecting Android Malware by Building Markov Chains," *NDSS*, 2018. https://www.ndss-symposium.org/ndss2018/ndss-2018-programme/mamadroid-detecting-android-malware-building-markov-chains/
[14] J. Crussell et al., "Attack of the Clones: Detecting Cloned Applications," *ESORICS*, 2012. https://doi.org/10.1007/978-3-642-33167-1_3
[15] A. Reina et al., "A System Call-Centric Analysis Technique to Reconstruct Android Malware Behaviors," *EuroSec*, 2013. https://doi.org/10.1145/2486788.2486793
[16] Google, "Android Security Overview," Android Open Source Project, 2023. https://source.android.com/security
[17] C. C. Aggarwal, *Neural Networks and Deep Learning*, Springer, 2018. https://doi.org/10.1007/978-3-319-94463-0
[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016. https://www.deeplearningbook.org
[19] N. Peiravian and X. Zhu, "Machine Learning for Android Malware Detection Using Permission and API Calls," *ICTAI*, 2013. https://doi.org/10.1109/ICTAI.2013.114
[20] S. Yerima et al., "Android Malware Detection Using Machine Learning Techniques," *IEEE Cyber Security Conference*, 2014. https://doi.org/10.1109/CSCloud.2014.34
[21] R. Xu et al., "Aurasium: Practical Policy Enforcement for Android Applications," *USENIX Security Symposium*, 2012. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/xu
[22] A. Sadeghi et al., "A Taxonomy and Qualitative Comparison of Program Analysis Techniques for Security Assessment of Android Software," *IEEE Transactions on Software Engineering*, 2017. https://doi.org/10.1109/TSE.2016.2630727
[23] P. Faruki et al., "Android Security: A Survey of Issues, Malware Penetration, and Defenses," *IEEE Communications Surveys & Tutorials*, 2015. https://doi.org/10.1109/COMST.2014.2386139
[24] M. Egele et al., "PiOS: Detecting Privacy Leaks in iOS Applications," *NDSS*, 2011. https://www.ndss-symposium.org/ndss2011/pios-detecting-privacy-leaks-ios-applications/
[25] J. Brownlee, *Machine Learning Algorithms From Scratch with Python*, 2016. https://machinelearningmastery.com/machine-learning-algorithms-from-scratch/