Reverse Engineering Protokol DVR *Legacy* Berbasis LLM

Putu Gede Semara Putra[1a)], Roy Rudolf Huizen [1b)], Dandy Pramana Hostiadi [1c)]
[1)] Magister Sistem Informasi, Institut Teknologi dan Bisnis STIKOM Bali, Bali, Indonesia
e-mail: [a)]252011003@stikom-bali.ac.id, [b)]Roy@stikom-bali.ac.id, [c)]dandy@stikom-bali.ac.id

Abstrak

Pertumbuhan penggunaan perangkat Digital Video Recorder (DVR) legacy berbasis Internet of Things tidak diimbangi dengan ketersediaan dokumentasi teknis dan dukungan interoperabilitas jangka panjang. Sebagian besar perangkat tersebut masih menggunakan protokol komunikasi proprietary yang hanya dapat diakses melalui aplikasi klien resmi. Kondisi ini menyebabkan ketergantungan vendor, keterbatasan integrasi sistem, serta meningkatnya risiko technological obsolescence. Penelitian ini bertujuan merekonstruksi protokol komunikasi DVR proprietary legacy menggunakan pendekatan non-invasif berbasis analisis statis aplikasi Android klien resmi yang diperkuat dengan asistensi Large Language Model. Analisis dilakukan terhadap berkas Android Package Kit melalui proses dekompilasi untuk mengidentifikasi struktur paket, mekanisme autentikasi, manajemen sesi, serta format streaming video tanpa melibatkan analisis firmware maupun sniffing jaringan. Hasil penelitian menunjukkan bahwa protokol menggunakan komunikasi berbasis TCP dengan struktur paket biner tetap dan payload modular Type–Length–Value, serta mentransmisikan data video dalam format raw H.264 tanpa protokol streaming standar. Spesifikasi protokol yang direkonstruksi kemudian divalidasi melalui implementasi klien independen berbasis Python yang berhasil melakukan autentikasi dan menerima streaming video secara berkelanjutan. Temuan ini membuktikan bahwa pendekatan non-invasif berbasis analisis statis dan penalaran berbantuan Large Language Model mampu menghasilkan rekonstruksi protokol yang fungsional dan dapat direproduksi untuk mendukung interoperabilitas lintas platform.

Kata kunci: *Reverse engineering, Non invasif, DVR legacy, statistic analysis, Large Language Model.*

1. Pendahuluan

Perkembangan ekosistem *Internet of Things* (IoT) telah mendorong adopsi perangkat pintar secara luas pada berbagai sektor, termasuk keamanan rumah tangga, pengawasan publik, industri, hingga infrastruktur kritis. Salah satu perangkat yang paling banyak digunakan adalah *Digital Video Recorder* (DVR) dan *Network Video Recorder* (NVR), khususnya produk kelas konsumen dan semi-industri yang diproduksi secara masif oleh vendor asal Tiongkok [1]. Popularitas perangkat ini didorong oleh biaya produksi yang rendah, kemudahan instalasi, serta dukungan akses jarak jauh melalui aplikasi seluler berbasis Android dan iOS [2]. Namun, pertumbuhan tersebut tidak diimbangi dengan pengelolaan keamanan dan interoperabilitas yang memadai. Sebagian besar DVR *legacy* masih menggunakan protokol komunikasi *proprietary* yang tidak terdokumentasi secara publik dan hanya dapat diakses melalui aplikasi resmi vendor. Ketergantungan pada protokol tertutup menciptakan kondisi vendor lock-in, yang semakin bermasalah ketika perangkat mencapai status *end-of-life*. Banyak perangkat tetap beroperasi tanpa pembaruan firmware maupun keamanan jangka panjang, sehingga menimbulkan akumulasi risiko keamanan berkelanjutan [3], [4]. Di sisi lain, aplikasi klien Android sering kali tetap tersedia meskipun dukungan resminya telah dihentikan. Kondisi ini menjadikan aplikasi klien sebagai satu-satunya representasi implementasi formal protokol yang masih dapat dianalisis. Tanpa dokumentasi terbuka, perangkat DVR *legacy* sulit diintegrasikan dengan sistem manajemen video modern, platform *open-source*, maupun solusi analitik berbasis kecerdasan buatan, sehingga berpotensi mempercepat terjadinya *technological obsolescence* dan *e-waste*. Dalam konteks tersebut, reverse engineering protokol komunikasi menjadi pendekatan penting untuk merekonstruksi struktur pesan, mekanisme autentikasi, serta alur komunikasi sistem yang tidak terdokumentasi. Pendekatan ini telah lama digunakan dalam analisis sistem *embedded* dan IoT, baik untuk tujuan keamanan maupun interoperabilitas [5]. Namun, *reverse engineering* pada perangkat *legacy* menghadapi keterbatasan akses firmware, risiko operasional, serta kendala analisis lalu lintas jaringan secara dinamis. Pendekatan non-invasif berbasis analisis statis aplikasi klien menjadi semakin relevan. Pada platform Android, file APK menyimpan logika komunikasi protokol secara relatif lengkap, sehingga memungkinkan ekstraksi struktur paket dan alur komunikasi melalui proses dekompilasi. Analisis statis APK telah diakui sebagai metode efektif dalam memahami perilaku aplikasi dan protokol

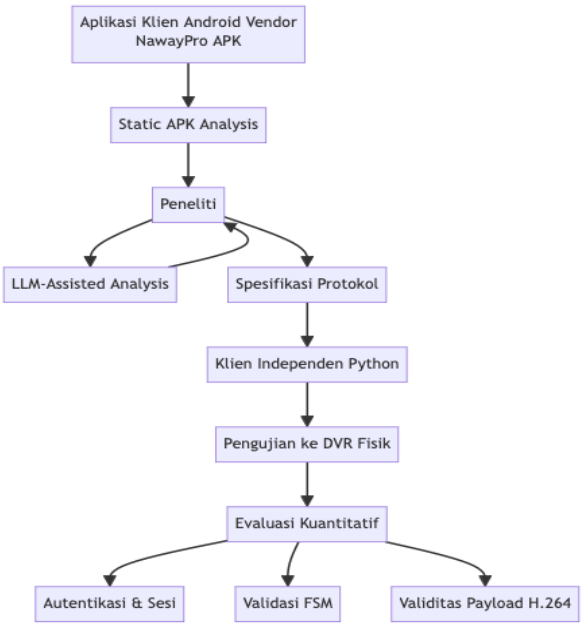
jaringan yang tidak terdokumentasi [6]. Meskipun demikian, kompleksitas aplikasi modern, termasuk obfuscation dan optimasi kode, menjadikan analisis statis murni sebagai proses yang tidak selalu efisien dan sulit diskalakan [7].

Perkembangan *Large Language Model* (LLM) membuka peluang baru dalam mendukung proses reverse engineering. LLM terbukti mampu membantu pemahaman kode, analisis semantik, dan penalaran lintas fungsi dalam sistem perangkat lunak kompleks [8], [9], [10]. Penelitian terkini menunjukkan bahwa LLM dapat dimanfaatkan sebagai alat bantu analisis statis untuk memahami aplikasi dan sistem legacy yang minim dokumentasi [11], [12]. Dalam konteks reverse engineering protokol, LLM berperan sebagai akselerator kognitif yang membantu mengidentifikasi pola struktur data, format pesan, dan hubungan antar modul kode melalui pendekatan *human-in-the-loop*.

Berdasarkan latar belakang tersebut, penelitian ini berfokus pada reverse engineering protokol komunikasi DVR proprietary legacy menggunakan pendekatan non-invasif berbasis analisis statis aplikasi Android klien resmi yang diperkuat dengan asistensi LLM dengan menggunakan model komersial seperti *Claude* dan *ChatGPT*. Penelitian ini tidak melibatkan analisis *firmware* maupun *sniffing* jaringan, melainkan memvalidasi hasil rekonstruksi melalui implementasi klien independen berbasis Python yang berkomunikasi langsung dengan perangkat DVR. Tujuan utama penelitian adalah merekonstruksi spesifikasi protokol secara fungsional guna mendukung interoperabilitas lintas platform dan pengembangan ekosistem terbuka yang berkelanjutan.

2. Metode Penelitian

Penelitian ini menggunakan pendekatan non-invasif berbasis analisis statis APK yang didekompilasi menggunakan *Java-based decompiler (jadx)*. Analisis dilakukan terhadap aplikasi klien Android vendor untuk mengidentifikasi struktur paket dan alur komunikasi tanpa melibatkan modifikasi firmware maupun inspeksi lalu lintas jaringan *sniffing* [13], [14], [15].



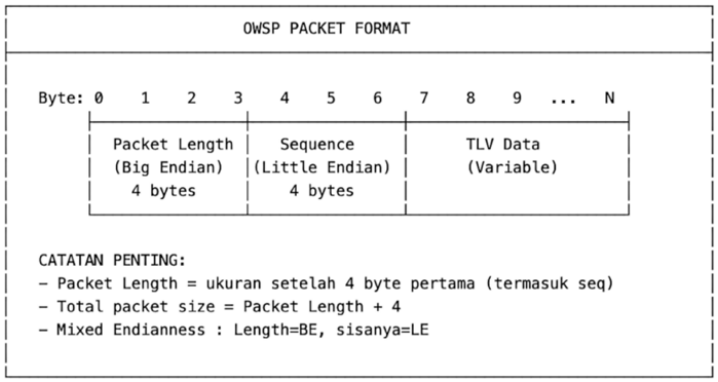
Gambar 1. Diagram Alur Penelitian

Proses interpretasi kode hasil dekompilasi dilakukan dalam kerangka *human-in-the-loop*, dengan *Large Language Model* (LLM) digunakan sebagai alat bantu untuk mengenali pola struktur data dan hubungan antar modul komunikasi, sementara validasi akhir tetap dilakukan secara manual oleh peneliti. Hasil analisis dikonsolidasikan menjadi spesifikasi protokol konseptual yang kemudian diimplementasikan dalam klien independen berbasis Python. Validasi dilakukan melalui pengujian langsung ke perangkat DVR fisik menggunakan evaluasi kuantitatif yang mencakup metrik autentikasi dan sesi, validasi perilaku berbasis *finite-state machine* (FSM), serta evaluasi validitas *payload* video H.264. Keberhasilan seluruh pengujian menunjukkan bahwa protokol hasil rekonstruksi mampu merepresentasikan perilaku aktual sistem DVR secara fungsional dan dapat direproduksi.

3. Hasil dan Pembahasan

3.1 Hasil Reverse Engineering Protokol Komunikasi DVR

DVR yang diteliti menggunakan protokol komunikasi proprietary pada lapisan aplikasi di atas TCP untuk menangani autentikasi, manajemen sesi, dan distribusi streaming video melalui satu koneksi persisten. Protokol ini tidak memiliki dokumentasi publik dan hanya dapat diakses melalui aplikasi klien resmi vendor. Analisis statis menunjukkan bahwa komunikasi menggunakan paket biner berstruktur tetap yang diawali *magic bytes*, diikuti *field* panjang paket, *command identifier*, serta *payload* dinamis. Field panjang paket sepanjang empat *byte big-endian* digunakan untuk framing data, sedangkan sequence number empat byte little-endian berfungsi menjaga urutan paket dan konsistensi sesi selama transmisi berkelanjutan. Payload utama protokol disusun menggunakan struktur *Type–Length–Value* (TLV) yang bersifat modular.



Gambar 2. OWSP Packet Format

Setiap TLV terdiri atas *field* tipe, panjang, dan nilai aktual yang disimpan dalam format *little-endian*. Nilai Type membedakan fungsi pesan seperti login, autentikasi, kontrol *channel*/kanal, permintaan *streaming*, dan pengiriman data video, sementara struktur TLV memungkinkan perluasan protokol tanpa perubahan format dasar paket. Pada pesan streaming, TLV diterapkan secara bersarang/*nested*, di mana TLV tingkat luar memuat metadata frame seperti *channel*, indeks frame, *checksum*, dan timestamp, sedangkan TLV tingkat dalam berisi data video aktual. Payload video dikirim dalam bentuk raw H.264 *Network Abstraction Layer Unit* (NALU) tanpa enkapsulasi protokol standar seperti RTP atau RTSP. Meskipun tidak menggunakan protokol terbuka seperti ONVIF atau RTSP, penggunaan format H.264 standar memungkinkan data video tetap dapat didekode menggunakan pipeline konvensional setelah proses ekstraksi dilakukan.

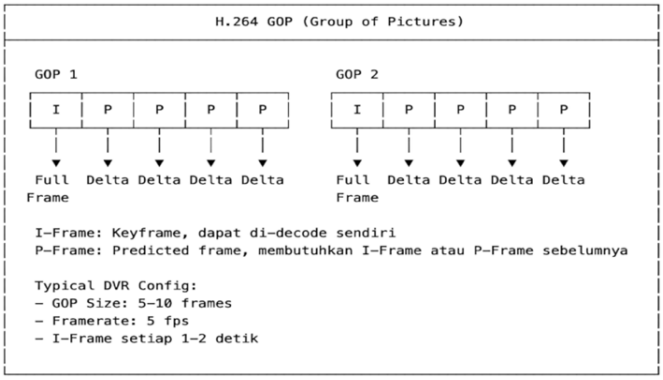
3.2 Mekanisme Autentikasi, Inisialisasi, dan Manajemen Sesi

Setiap paket login diawali dengan packet header sepanjang delapan byte, yang terdiri atas field panjang paket empat byte berformat *big-endian* dan *sequence number* empat *byte* berformat *little-endian*. *Field* panjang paket digunakan untuk menentukan batas framing data pada aliran TCP, sedangkan *sequence number* berfungsi menjaga urutan paket serta konsistensi sesi autentikasi antara klien dan DVR. Setelah header, paket login memuat sejumlah struktur *Type–Length–Value* (TLV). TLV pertama merupakan *Version* TLV dengan tipe 40 (0x28) dan panjang empat *byte*, yang berisi informasi versi protokol dalam format *major–minor little-endian*, misalnya nilai 0x03 00 08 00 yang merepresentasikan versi 3.8. Informasi ini digunakan perangkat untuk memverifikasi kompatibilitas klien sebelum melanjutkan proses autentikasi. TLV berikutnya adalah Login TLV dengan tipe 41 (0x29) dan panjang data 56 *byte*, yang memiliki struktur tetap berbasis *offset byte* absolut tanpa *delimiter* teks.

Nilai TLV login diawali dengan *username* sepanjang 32 *byte* dan *password* sepanjang 16 *byte* dalam format ASCII yang dipadatkan menggunakan *null-padding*. Tidak ditemukan mekanisme *hashing* maupun enkripsi pada level aplikasi, sehingga kredensial dikirimkan secara langsung. Struktur ini diikuti oleh device identifier empat byte little-endian, satu byte flag mode koneksi, satu byte channel identifier (0–255, dengan nilai 0xFF merepresentasikan mode multi-kanal), serta dua *byte reserved field* yang umumnya bernilai nol. Setelah paket login dikirimkan, DVR memvalidasi kredensial, versi protokol, dan parameter kanal. Jika autentikasi berhasil, perangkat mengirimkan respons berupa TLV status dengan kode keberhasilan 42 (0x2A); sebaliknya, kegagalan autentikasi atau ketidaksesuaian versi akan menghasilkan kode kesalahan dan terminasi sesi.

3.3 Struktur Data Streaming dan Alur Transmisi Video

Setelah autentikasi berhasil, klien dapat mengirimkan perintah untuk memulai streaming video dari channel tertentu. DVR kemudian mulai mengirimkan data streaming dalam bentuk paket TLV berurutan yang berisi metadata frame dan data *raw* video. Setiap unit data video diawali oleh informasi kontrol yang mencakup identitas channel, tipe frame, ukuran payload, dan penanda waktu. Data video yang dikirimkan merupakan bitstream H.264 standar yang dienkapsulasi langsung ke dalam payload TLV tanpa lapisan proteksi tambahan. Alur streaming mengikuti pola *Group of Pictures* (GOP), di mana DVR secara periodik mengirimkan frame referensi (I-frame) diikuti oleh sejumlah frame prediktif (P-frame).



Gambar 3. Pola *Group of Pictures*

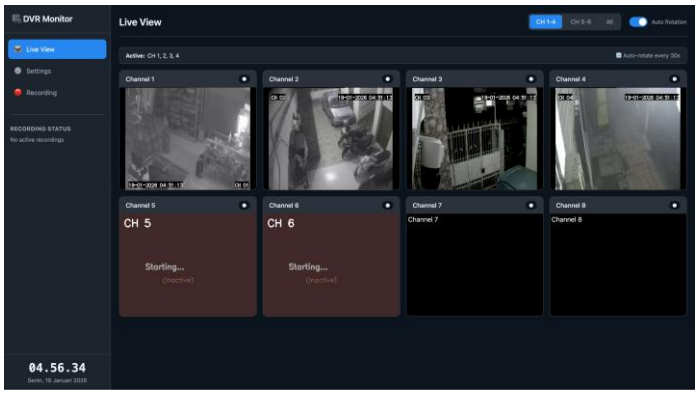
Urutan ini harus dipertahankan agar decoding dapat dilakukan dengan benar pada sisi klien. Klien yang tidak menerima I-*frame* awal tidak dapat langsung melakukan decoding dan harus menunggu siklus GOP berikutnya. Analisis menunjukkan bahwa DVR tidak menyediakan mekanisme retransmisi atau koreksi kesalahan. Kehilangan paket pada lapisan transport dapat mengakibatkan artefak visual atau kegagalan decoding hingga I-*frame* berikutnya diterima. Hal ini menunjukkan bahwa desain protokol lebih mengutamakan kesederhanaan implementasi dibandingkan ketahanan terhadap kondisi jaringan yang tidak stabil.

3.4 Kontribusi Large Language Model (LLM) dalam Analisis Protokol

Large Language Model (LLM) digunakan sebagai alat bantu analisis untuk mempercepat dan memperdalam proses reverse engineering protokol. Dalam konteks penelitian ini, LLM berperan sebagai sistem pendukung penalaran yang membantu menginterpretasikan struktur logika aplikasi klien hasil dekompilasi. LLM dimanfaatkan untuk mengidentifikasi pola-pola struktural dalam kode, seperti fungsi yang berulang dalam pembentukan paket, penggunaan konstanta numerik tertentu sebagai penanda perintah, serta hubungan antara fungsi autentikasi dan streaming. Selain itu, LLM membantu merumuskan hipotesis awal mengenai makna semantik dari elemen data yang tidak diberi nama deskriptif dalam kode. Penting untuk dicatat bahwa LLM tidak digunakan untuk menggantikan analisis teknis manual. Setiap interpretasi yang dihasilkan oleh LLM diperlakukan sebagai hipotesis sementara yang harus diverifikasi melalui penelusuran kode lanjutan dan pengujian langsung terhadap perangkat DVR. Pendekatan ini memastikan bahwa hasil akhir tetap berbasis bukti empiris. Integrasi LLM dalam proses reverse engineering terbukti efektif dalam mengurangi beban kognitif peneliti, khususnya ketika menghadapi basis kode legacy yang kompleks dan minim dokumentasi. Dengan demikian, LLM berfungsi sebagai akselerator analisis, bukan sebagai sumber kebenaran tunggal.

3.5 Implementasi Klien Independen dan Validasi Empiris

Untuk memverifikasi keakuratan hasil rekonstruksi protokol, penelitian ini mengembangkan sebuah klien independen berbasis Python yang mengimplementasikan seluruh tahapan komunikasi OWSP sebagaimana yang telah direkonstruksi. Klien ini dirancang untuk membangun koneksi TCP, mengirimkan paket inisialisasi dan autentikasi, mem-parsing respons DVR, serta menangani aliran data streaming video. Implementasi parsing TLV dilakukan secara generik agar mampu menangani variasi panjang dan urutan elemen data.



Gambar 4. Implementasi Webapp Protokol DVR

Pengujian mencakup keberhasilan autentikasi, kesetaraan perilaku protokol (*Behavioral Equivalence Metrics*), serta validitas semantik *payload* video. Sebanyak 20 sesi login independen dijalankan secara berurutan. Hasil pengujian autentikasi dan pembentukan sesi ditunjukkan pada Tabel 1.

Tabel 1. Hasil Pengujian Autentikasi dan Sesi

Metrik	Nilai
Authentication Success Rate	100%
Session Establishment Rate	100%
Session Failure Rate	0%

Seluruh sesi berhasil dibangun secara konsisten, menunjukkan bahwa struktur paket autentikasi dan mekanisme negosiasi hasil rekonstruksi telah sesuai dengan perilaku aktual perangkat. Pengujian selanjutnya adalah kesetaraan perilaku protokol yang divalidasi menggunakan model *finite-state machine* (FSM). Setiap respons perangkat dianalisis untuk memastikan kesesuaian urutan transisi state. Setiap respons DVR selama pengujian dipetakan terhadap transisi FSM. Pengujian hanya dianggap berhasil apabila seluruh transisi terjadi secara berurutan tanpa penyimpangan. Hasil pengujian ditunjukkan pada Tabel 2.

Tabel 2. Hasil Pengujian Kesataraan Perilaku Protocol

Metrik	Nilai
Command Acceptance Rate	100%
Valid State Transitions	6
Protocol Errors	0

Seluruh *transisi state* yang diamati sesuai dengan model FSM yang direkonstruksi. Tidak ditemukan transisi ilegal, kondisi *undefined state*, maupun protocol *desynchronization*. Hasil ini menunjukkan bahwa spesifikasi protokol yang direkonstruksi telah mencerminkan perilaku kontrol sistem DVR yang sama. Pengujian berikutnya yaitu validasi payload yang dilakukan pada tingkat *codec* menggunakan *pipeline* dekode H.264 standar. Hasilnya dirangkum pada Tabel 3.

Tabel 3. Hasil Pengujian Validitas Semantik *Payload* Video

Metrik	Nilai
Total NAL Units	40
Valid NALU Ratio	100%
Decode Success Rate	100%
Average Interval GOP	3,0 NAL

Nilai total NAL unit menunjukkan jumlah unit bitstream H.264 yang berhasil diidentifikasi selama sesi streaming. Interval GOP dihitung berdasarkan jarak antar NAL unit bertipe IDR (*Instantaneous Decoding Refresh*), sehingga dinyatakan dalam satuan NAL (*Network Abstraction Layer*). Pendekatan ini dipilih karena struktur GOP ditentukan secara intrinsik oleh urutan frame I dan P dalam bitstream H.264, bukan oleh nilai temporal eksternal. Interval GOP dihitung berdasarkan jarak antar NAL unit bertipe IDR dan dinyatakan dalam satuan NAL untuk merefleksikan kontinuitas struktural stream.

4. Kesimpulan

Penelitian ini berhasil merekonstruksi protokol komunikasi proprietary pada sistem DVR legacy secara non-invasif melalui analisis statis aplikasi klien Android dengan asistensi *Large Language Model*. Hasil analisis mengungkap struktur paket berbasis *Type–Length–Value*, mekanisme autentikasi dan manajemen sesi, serta transmisi bitstream video H.264 *raw* di atas koneksi TCP tanpa dukungan protokol standar seperti RTSP atau ONVIF. Validasi empiris menggunakan klien independen berbasis Python menunjukkan tingkat keberhasilan autentikasi dan pembentukan sesi sebesar 100%, kesetaraan perilaku protokol yang konsisten dengan model *finite-state machine*, serta validitas semantik *payload* video dengan rasio NALU dan keberhasilan dekode mencapai 100%. Temuan ini menegaskan bahwa pendekatan analisis statis berbantuan LLM mampu menghasilkan spesifikasi protokol yang akurat, fungsional, dan dapat direproduksi tanpa intervensi firmware maupun analisis jaringan dinamis, sehingga berpotensi mendukung peningkatan interoperabilitas sistem video *legacy*.

Daftar Pustaka

[1] L. Rzayeva, M. Shayakhmetov, Y. Atanbayev, R. Budenov, dan H. Mutaher, "Automated Forensic Recovery Methodology for Video Evidence from Hikvision and Dahua DVR/NVR Systems," *Information*, vol. 16, no. 11, hlm. 983, Nov 2025, doi: 10.3390/info16110983.

[2] A. Allen, A. Mylonas, S. Vidalis, dan D. Gritzalis, "Security Evaluation of Companion Android Applications in IoT: The Case of Smart Security Devices," *Sensors*, vol. 24, no. 17, hlm. 5465, Agu 2024, doi: 10.3390/s24175465.

[3] Y. Meidan *dkk.*, "ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis," dalam *Proceedings of the Symposium on Applied Computing*, Marrakech Morocco: ACM, Apr 2017, hlm. 506–509. doi: 10.1145/3019612.3019878.

[4] K. Liu *dkk.*, "On Manually Reverse Engineering Communication Protocols of Linux-Based IoT Systems," *IEEE Internet Things J.*, vol. 8, no. 8, hlm. 6815–6827, Apr 2021, doi: 10.1109/JIOT.2020.3036232.

[5] T. Bakhshi, B. Ghita, dan I. Kuzminykh, "A Review of IoT Firmware Vulnerabilities and Auditing Techniques," *Sensors*, vol. 24, no. 2, hlm. 708, Jan 2024, doi: 10.3390/s24020708.

[6] W. Enck, D. Octeau, P. McDaniel, dan S. Chaudhuri, "A study of android application security," dalam *Proceedings of the 20th USENIX Conference on Security*, dalam SEC'11. USA: USENIX Association, 2011, hlm. 21.

[7] B. Molina-Coronado, A. Ruggia, U. Mori, A. Merlo, A. Mendiburu, dan J. Miguel-Alonso, "Light up that Droid! On the effectiveness of static analysis features against app obfuscation for Android malware detection," *J. Netw. Comput. Appl.*, vol. 235, hlm. 104094, Mar 2025, doi: 10.1016/j.jnca.2024.104094.

[8] Adewale A. Adeniran, "Research Survey on the Use of Reverse Engineering Embedded Systems in Security Analysis of IoT Device Firmware and FCC's Voluntary IoT Labeling Program," *Int. J. Comput. Appl. Technol. Res.*, Jul 2025, doi: 10.7753/IJCATR1407.1004.

[9] X. Hu, Z. Fu, S. Xie, S. H. H. Ding, dan P. Charland, "SoK: Potentials and Challenges of Large Language Models for Reverse Engineering," 26 September 2025, *arXiv*: arXiv:2509.21821. doi: 10.48550/arXiv.2509.21821.

[10] Z. Zheng *dkk.*, "Towards an Understanding of Large Language Models in Software Engineering Tasks," 10 Desember 2024, *arXiv*: arXiv:2308.11396. doi: 10.48550/arXiv.2308.11396.

[11] H. A. Mohammed Salih dan Q. I. Sarhan, "A Systematic Survey on Large Language Models for Static Code Analysis," *ARO- Sci. J. KOYA Univ.*, vol. 13, no. 1, hlm. 251–265, Jun 2025, doi: 10.14500/aro.12082.

[12] Z. Sheng, Z. Chen, S. Gu, H. Huang, G. Gu, dan J. Huang, "LLMs in Software Security: A Survey of Vulnerability Detection Techniques and Insights," *ACM Comput. Surv.*, vol. 58, no. 5, hlm. 1–35, Apr 2026, doi: 10.1145/3769082.

[13] D. Gomes, E. Felix, F. Aires, dan M. Vieira, "Static Code Analysis for IoT Security: A Systematic Literature Review," *ACM Comput. Surv.*, vol. 58, no. 3, hlm. 1–47, Feb 2026, doi: 10.1145/3745019.

[14] Z. Li, S. Dutta, dan M. Naik, "IRIS: LLM-Assisted Static Analysis for Detecting Security Vulnerabilities," 6 April 2025, *arXiv*: arXiv:2405.17238. doi: 10.48550/arXiv.2405.17238.

[15] J. Wang, N. Tao, W.-B. Lee, dan Q. Zhao, "A Contemporary Survey of Large Language Model Assisted Program Analysis," *Trans. Artif. Intell.*, hlm. 6, Mei 2025, doi: 10.53941/tai.2025.100006.