Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 204-215
p-ISSN: 2962-3693 | e-ISSN: 2962-7788

Perbandingan Static Analysis dan Dynamic Analysis Deteksi Kerentanan Smart Contract Solidity

Andreas Perdana[1], Febri Sugandi[1], Ridwan Yusuf[1]*, Untoro Apsiswanto[1]

[1]Universitas Dharmawacana Metro, Metro, Indonesia

E-mail: andreas.perdana@dharmawacana.ac.id; fsugandi87@gmail.com; ryusuf87@gmail.com; untorolampung@gmail.com

**Penulis Korespondensi (Corresponding Author)*

© 2025. Andreas Perdana dkk. Diterbitkan oleh JTDS. Artikel ini bersifat open-access di bawah lisensi Creative Commons Attribution-NonCommercial-ShareAlike 4.0 (CC BY-NC-SA 4.0).

Info Artikel

Diterima:
20 September 2025
Direvisi:
8 November 2025
Diterbitkan:
20 Desember 2025

Kata Kunci:
smart contract; static analysis; dynamic analysis; deteksi kerentanan; ethereum

ABSTRAK

Smart contract pada platform Ethereum mengelola aset finansial bernilai besar, namun sifat immutable membuat kerentanan kode berdampak permanen sebagaimana ditunjukkan kasus The DAO, Parity Wallet, dan Ronin Bridge. Dua pendekatan pendeteksian kerentanan telah berkembang luas: static analysis yang memeriksa kode sumber atau bytecode tanpa eksekusi, dan dynamic analysis yang menjalankan kontrak pada lingkungan simulasi dengan masukan terstruktur. Keduanya memiliki trade-off kecepatan, cakupan, dan tingkat false positive yang berbeda, namun perbandingan sistematis pada lingkungan pengembangan lokal Indonesia menggunakan alat versi terbaru masih jarang dilakukan. Artikel ini memaparkan rancangan penelitian eksperimental kuantitatif yang akan membandingkan Slither (static) dengan Foundry dan Echidna (dynamic) pada dataset 25 smart contract Solidity yang mencakup lima kategori kerentanan utama: reentrancy, integer overflow, access control, unchecked return values, dan unbounded loop. Metrik perbandingan meliputi precision, recall, F1-score, dan waktu komputasi. Kontribusi yang diharapkan adalah kerangka komparatif yang dapat menjadi rujukan praktis bagi pengembang Web3 Indonesia dalam memilih alat keamanan dan strategi pengujian hibrid yang sesuai dengan tingkat risiko proyek serta sumber daya tim.

Article Info

Received:
September 20, 2025
Revised:
November 8, 2025
Published:
December 20, 2025

Keywords:
smart contract; static analysis; dynamic analysis; vulnerability detection; ethereum

ABSTRACT

Smart contracts on the Ethereum platform manage financial assets of significant value, yet their immutable nature renders any code vulnerability permanently exploitable as demonstrated by The DAO, Parity Wallet, and Ronin Bridge incidents. Two vulnerability detection approaches have developed in parallel: static analysis that inspects source code or bytecode without execution, and dynamic analysis that runs contracts in simulated environments with structured inputs. Each carries different trade-offs in speed, coverage, and false-positive rate, yet systematic comparisons within Indonesian local development settings using the latest tool versions remain scarce. This article presents the design of a quantitative experimental study that will compare Slither (static) against Foundry and Echidna (dynamic) on

Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 204-215
p-ISSN: 2962-3693 | e-ISSN: 2962-7788

a dataset of 25 Solidity smart contracts covering five major vulnerability categories: reentrancy, integer overflow, access control, unchecked return values, and unbounded loop. Comparison metrics include precision, recall, F1-score, and computation time. The expected contribution is a comparative framework that can serve as a practical reference for Indonesian Web3 developers in choosing security tools and hybrid testing strategies appropriate to project risk level and team resources.

1. PENDAHULUAN

Smart contract adalah program yang berjalan di atas blockchain dan mengeksekusi logika bisnis secara otomatis ketika kondisi yang disepakati terpenuhi. Pada platform Ethereum, smart contract umumnya ditulis dalam bahasa Solidity dan dikompilasi menjadi bytecode yang dijalankan oleh Ethereum Virtual Machine. Sifat immutable, deterministik, dan akses terbuka pada blockchain publik menjadikan smart contract sangat menarik untuk aplikasi keuangan terdesentralisasi, namun ketiga sifat itu pula yang menyebabkan kerentanan kode bersifat permanen begitu kontrak ter-deploy. Insiden The DAO 2016 yang mengakibatkan kerugian sekitar 60 juta dolar AS dan eksploitasi Parity Wallet 2017 yang membekukan lebih dari 150 juta dolar AS menjadi rujukan klasik mengenai dampak permanen tersebut.

Penelitian deteksi kerentanan smart contract berkembang dalam dua jalur paralel. Static analysis menelaah kode atau bytecode tanpa menjalankannya, mengandalkan abstract syntax tree, control flow graph, dan kadang symbolic execution. Liu dkk. [1] menggabungkan Graph Neural Network dengan pengetahuan pakar untuk mendeteksi kerentanan smart contract dan melaporkan akurasi yang melebihi pendekatan rule-based murni; Kushwaha dkk. [2] menyusun systematic review terhadap kerentanan utama Ethereum smart contract dengan taksonomi yang banyak dirujuk publikasi berikutnya; Tolmach dkk. [3] mensurvei pendekatan formal specification dan verification yang menjadi varian kuat dari static analysis. Dynamic analysis mengeksekusi kontrak pada testnet lokal dengan input terstruktur (fuzzing) atau asersi properti (invariant testing). Boi dan Esposito [4] membandingkan prompt engineering dan fine-tuning untuk deteksi kerentanan pada Solana dan Algorand, memperluas perspektif komparatif lintas platform. Kado dan Tsuchiya [5] dan Mane dkk. [6] memanfaatkan Large Language Model untuk perbaikan otomatis kontrak teobfuskasi maupun deteksi kerentanan firmware IoT.

Sintesis terhadap dua puluh publikasi terkait pada kurun 2020–2025 [1]–[20] memperlihatkan tiga celah substantif. Pertama, perbandingan langsung antara static dan dynamic analysis dengan dataset yang sama, alat versi terbaru, dan metrik yang konsisten masih sangat terbatas; sebagian besar studi mengevaluasi satu kelas alat atau membandingkan beberapa alat di dalam satu kelas. Kedua, studi yang ada mayoritas dilakukan pada dataset Inggris berbahasa peneliti asing dan jarang menyertakan pertimbangan ketersediaan sumber daya komputasi serta preferensi alur kerja pengembang lokal. Ketiga, panduan praktis kapan menggunakan static, dynamic, atau kombinasi keduanya untuk kategori kerentanan tertentu belum tersedia dalam bentuk yang langsung dapat diadopsi tim pengembang Web3 skala kecil.

Rancangan penelitian ini menjawab tiga pertanyaan: (RQ1) bagaimana perbandingan akurasi (precision, recall, F1-score) antara static analysis dengan Slither dan dynamic analysis dengan Foundry serta Echidna pada dataset smart contract berlabel? (RQ2) bagaimana perbandingan efisiensi waktu komputasi dan effort konfigurasi antar pendekatan? dan (RQ3) jenis kerentanan apa yang lebih efektif ditangani masing-masing pendekatan, dan kapan kombinasi keduanya direkomendasikan? Kontribusi spesifik artikel ini adalah: (1) protokol eksperimen yang dapat direplikasi dengan dataset, alat, dan

metrik yang dispesifikasikan secara rinci; (2) taksonomi lima kategori kerentanan utama beserta indikator efektivitas deteksi yang dapat dijadikan dasar pemilihan alat; dan (3) kerangka rekomendasi strategi hibrid bagi pengembang Web3 Indonesia. Sisa naskah disusun sebagai berikut: Bagian 2 mengulas literatur tentang smart contract, kerentanan utama, dan kedua pendekatan deteksi; Bagian 3 memerinci rancangan eksperimen; Bagian 4 menyajikan hasil yang diharapkan beserta argumen pendukungnya; dan Bagian 5 menutup dengan kesimpulan dan saran tindak lanjut.

2. KAJIAN PUSTAKA

Bagian ini menyusun landasan teoritis dalam lima tema: smart contract pada Ethereum, taksonomi kerentanan, prinsip static analysis, prinsip dynamic analysis, dan studi komparatif terdahulu. Tabel I merangkum perbandingan empat studi terdekat yang menjadi rujukan posisi penelitian.

2.1 Smart Contract pada Ekosistem Ethereum

Smart contract pada Ethereum adalah program yang dijalankan oleh Ethereum Virtual Machine (EVM) ketika sebuah transaksi memanggil alamat kontrak. Taherdoost [7] menyusun tinjauan kritis terhadap karakteristik teknis dan implikasi smart contract pada blockchain, sedangkan Lin dkk. [8] mensurvei spektrum aplikasi smart contract lintas domain. Smart contract menjadi tulang punggung aplikasi terdesentralisasi seperti pertukaran token, pinjaman peer-to-peer, dan token non-fungible (NFT); Ante [9] dan Karandikar dkk. [10] memperlihatkan bagaimana ekosistem token Ethereum tumbuh seiring meningkatnya jumlah pengguna aktif. Sifat immutable mengharuskan pengembang menyelesaikan seluruh validasi keamanan sebelum deployment, karena perubahan kode pada kontrak yang sudah ter-deploy umumnya tidak dimungkinkan tanpa pola proxy atau migrasi yang kompleks. Konteks ini menjadikan deteksi kerentanan pra-deployment sebagai langkah yang tidak dapat ditawar.

2.2 Taksonomi Kerentanan Smart Contract

Kushwaha dkk. [2] mensistematisasi kerentanan smart contract Ethereum ke dalam beberapa kategori; SWC Registry yang menjadi rujukan industri mendaftar lebih dari tiga puluh kelas kerentanan terstandar. Penelitian ini memilih lima kategori dengan dampak finansial paling sering dilaporkan: (1) Reentrancy pemanggilan rekursif fungsi sebelum state diperbarui yang menjadi akar kasus The DAO; (2) Integer overflow/underflow luapan aritmetika yang dapat dikurangi sejak Solidity 0.8 dengan pemeriksaan otomatis namun tetap relevan untuk kontrak warisan; (3) Access control fungsi privileged yang tidak terlindung modifier seperti onlyOwner sehingga dapat dipanggil siapa pun; (4) Unchecked return values pengabaian status keberhasilan panggilan eksternal yang dapat menyembunyikan kegagalan transfer; dan (5) Unbounded loop perulangan tanpa batas yang dapat memicu kegagalan eksekusi akibat melampaui gas limit blok. Wu dkk. [11] memperlihatkan bahwa kerentanan tipe access control juga muncul dalam pola phishing pada level jaringan transaksi Ethereum; Arcenegui dkk. [12] menambahkan perspektif keamanan lintas IoT dan blockchain yang relevan untuk skenario integrasi.

2.3 Static Analysis dan Slither

Static analysis memeriksa kode tanpa mengeksekusinya, dengan fondasi pada abstract syntax tree (AST), control flow graph (CFG), data flow analysis, dan pada kasus tertentu symbolic execution. Liu dkk. [1] menggabungkan Graph Neural Network dengan pengetahuan pakar untuk meningkatkan akurasi deteksi, sedangkan Tolmach dkk. [3] mengarsip pendekatan formal verification yang menjadi varian kuat namun mahal komputasi. Slither yang dikembangkan Trail of Bits adalah salah satu alat static analysis paling banyak diadopsi karena menyajikan lebih dari delapan puluh detector bawaan, integrasi dengan continuous integration, dan keluaran yang dapat dipindai oleh tooling tambahan. Kelebihan static analysis mencakup waktu eksekusi singkat, cakupan kode yang luas secara teoritis, dan kemampuan mendeteksi bug pada jalur yang jarang dieksekusi; kelemahannya adalah tingkat false

positive yang relatif tinggi dan kesulitan menangani perilaku yang bergantung pada state runtime atau kombinasi pemanggilan lintas-fungsi yang kompleks.

2.4 Dynamic Analysis dan Fuzzing

Dynamic analysis menjalankan smart contract pada lingkungan simulasi dengan input terstruktur untuk mengamati pelanggaran properti yang ditetapkan pengembang. Foundry yang dikembangkan Paradigm menyediakan kerangka unit testing dan fuzzing berbasis Rust dengan dukungan invariant testing; Echidna menggunakan property-based fuzzing dengan strategi mutasi yang lebih agresif. Kado dan Tsuchiya [5] dan Mane dkk. [6] memperlihatkan bagaimana pendekatan berbasis Large Language Model mulai melengkapi dynamic analysis konvensional, khususnya untuk kontrak teobfuskasi atau firmware tanpa kode sumber. Kelebihan dynamic analysis adalah tingkat false positive yang rendah karena yang dilaporkan adalah pelanggaran nyata pada eksekusi konkret, serta kemampuan menangkap bug runtime yang luput dari analisis statis. Kekurangannya adalah ketergantungan cakupan pada input yang dihasilkan, kebutuhan menulis property atau invariant secara eksplisit, dan waktu eksekusi yang umumnya lebih panjang.

2.5 Posisi Penelitian Terhadap Studi Komparatif Terdahulu

Studi komparatif sebelumnya umumnya memfokuskan pada satu pendekatan saja atau membandingkan beberapa alat dalam pendekatan yang sama. Boi dan Esposito [4] membandingkan prompt engineering dan fine-tuning untuk LLM-based detection lintas Solana dan Algorand, sedangkan Moayyed dkk. [13] menganalisis sistematis penggunaan LLM pada transformasi dokumen ke smart contract. Kirli dkk. [14] dan Hamledari dkk. [15] memperluas konteks aplikasi smart contract ke sistem energi dan otomasi konstruksi, namun tidak fokus pada perbandingan alat pengujian. Tabel I memetakan posisi penelitian ini terhadap empat rujukan terdekat dengan dimensi alat, dataset, kategori kerentanan, dan metrik. Penelitian ini memilih kombinasi Slither, Foundry, dan Echidna karena ketiganya mewakili tools open-source paling banyak diadopsi pada 2024–2025 dan memiliki dokumentasi lengkap untuk replikasi.

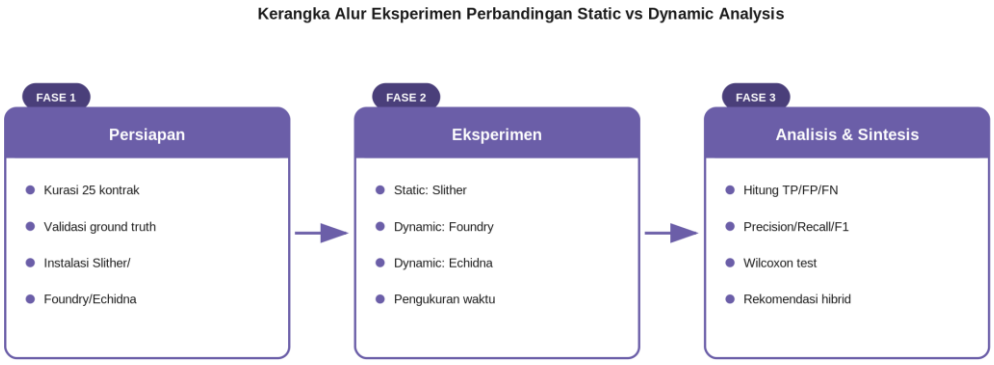
TABEL I
POSISI PENELITIAN TERHADAP STUDI KOMPARATIF TERDAHULU

Studi	Domain	Pendekatan	Dataset	Kategori Kerentanan	Metrik Utama
Liu dkk. [1]	Smart contract Ethereum	Static (GNN + ahli)	Skala besar	Multi-kelas	Akurasi
Boi & Esposito [4]	Solana, Algorand	LLM (prompt vs fine-tune)	Multi-platform	Multi-kelas	Akurasi, F1
Tolmach dkk. [3]	Smart contract umum	Formal verification	Studi pustaka	Multi-kelas	Survei kualitatif
Mane dkk. [6]	IoT firmware	LLM + EMBA + Ghidra	Firmware IoT	Multi-kelas	Deteksi
Penelitian ini	Smart contract Ethereum (Solidity)	Static (Slither) vs Dynamic (Foundry, Echidna)	25 kontrak berlabel	5 kategori (Reentrancy, Integer Overflow, Access Control, Unchecked Return, Unbounded Loop)	Precision, Recall, F1, Waktu

Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 204-215
p-ISSN: 2962-3693 | e-ISSN: 2962-7788

3. METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan eksperimental kuantitatif. Variabel bebas adalah metode deteksi (static vs dynamic), variabel terikat adalah metrik akurasi (precision, recall, F1-score) dan waktu komputasi. Kerangka alur eksperimen ditunjukkan pada Gbr. 1.



Gbr. 1. Kerangka alur eksperimen perbandingan static dan dynamic analysis.

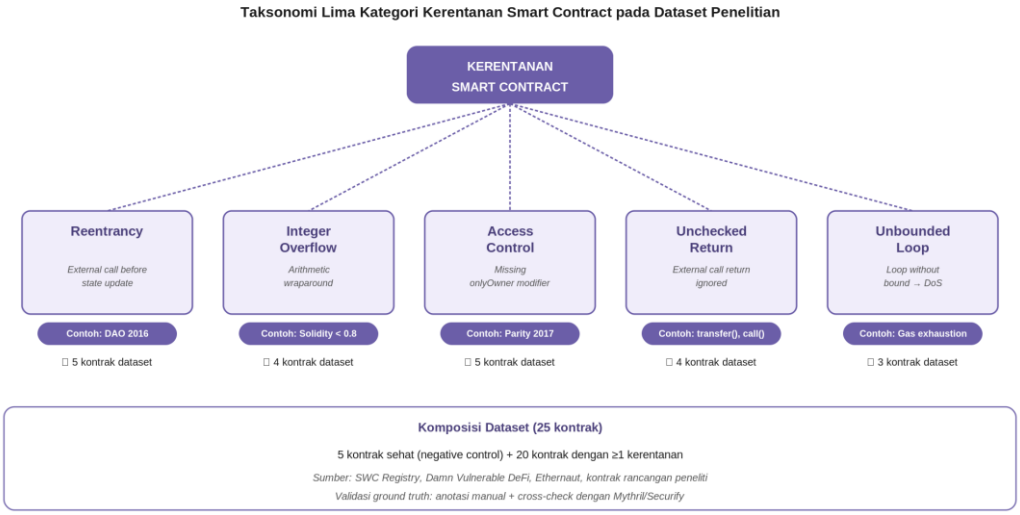
3.1 Desain Penelitian dan Variabel

Desain penelitian adalah eksperimen terkontrol dengan dataset bersama (paired design): setiap smart contract dijalankan dengan ketiga alat (Slither untuk static, Foundry dan Echidna untuk dynamic). Hal ini memungkinkan analisis berpasangan untuk mengurangi variansi yang berasal dari karakteristik kontrak. Variabel kontrol mencakup spesifikasi perangkat keras, versi alat, lama eksekusi fuzzing maksimal, dan jumlah runs untuk fuzzer.

3.2 Dataset dan Ground Truth

Dataset terdiri dari 25 smart contract Solidity yang disusun dari tiga sumber: (a) SWC Registry referensi standar Smart Contract Weakness Classification; (b) repositori publik seperti Damn Vulnerable DeFi dan Ethernaut yang menyediakan kontrak dengan kerentanan terlabel; dan (c) kontrak yang dibuat peneliti untuk mensimulasikan pola spesifik yang sulit ditemukan pada repo publik. Komposisi: 5 kontrak sehat (negative control) dan 20 kontrak dengan satu atau lebih kerentanan dari lima kategori target. Ground truth divalidasi melalui dua jalur paralel: anotasi manual berdasarkan literatur dan dokumen audit publik, serta cross-check dengan hasil tools rujukan industri (Mythril, Securify) untuk mengurangi bias anotator. Distribusi taksonomi kerentanan ditampilkan pada Gbr. 2.

Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 204-215
p-ISSN: 2962-3693 | e-ISSN: 2962-7788



Gbr. 2. Taksonomi lima kategori kerentanan smart contract pada dataset penelitian.

3.3 Instrumen dan Konfigurasi Alat

Static analysis menggunakan Slither v0.10 atau lebih baru dengan konfigurasi default (slither contract.sol) yang menjalankan seluruh detector bawaan. Dynamic analysis menggunakan dua alat secara bergantian. Foundry (forge test --fuzz-runs 5000) digunakan dengan property test dan invariant test yang ditulis spesifik untuk lima kategori kerentanan. Echidna dijalankan selama satu menit per kontrak atau hingga ditemukan pelanggaran property, mengikuti praktik baku pada literatur fuzzing smart contract. Lingkungan eksekusi adalah local testnet menggunakan Anvil dari Foundry. Spesifikasi perangkat: Intel Core i7 generasi terkini, 16 GB RAM, sistem operasi Linux Ubuntu 22.04 LTS. Arsitektur eksperimen ditunjukkan pada Gbr. 3.



Gbr. 3. Arsitektur eksperimen input, alat, dan keluaran pengukuran.

3.4 Metrik Evaluasi

Metrik mengikuti standar information retrieval dengan adaptasi pada konteks deteksi kerentanan. True Positive (TP) adalah kerentanan terdeteksi dan memang ada pada ground truth; False Positive (FP) adalah laporan kerentanan yang tidak ada pada ground truth; False Negative (FN) adalah kerentanan yang ada namun tidak terdeteksi. Precision dihitung sebagai TP/(TP+FP), Recall sebagai TP/(TP+FN), dan F1-score sebagai harmonic mean keduanya. Selain metrik akurasi, waktu rata-rata per kontrak

diukur dalam detik, mencakup waktu konfigurasi awal (untuk dynamic analysis: penulisan property dan invariant). Visualisasi metrik perbandingan ditunjukkan pada Gbr. 4 dan perbandingan rinci per kategori pada Gbr. 5.

3.5 Prosedur Eksperimen

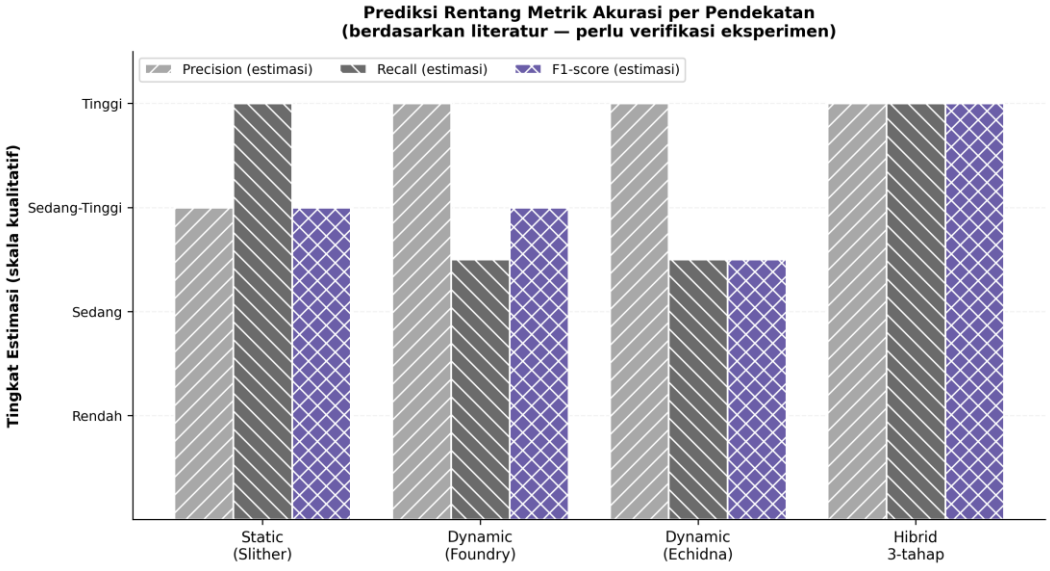
Prosedur eksperimen mengikuti enam langkah berurutan. (1) Persiapan dataset: pengumpulan dan verifikasi 25 kontrak dengan ground truth. (2) Eksperimen static: setiap kontrak dijalankan dengan Slither, hasilnya diklasifikasikan ke dalam lima kategori target dan dibandingkan dengan ground truth. (3) Eksperimen dynamic: penulisan property dan invariant untuk masing-masing kategori, lalu eksekusi Foundry dan Echidna dengan parameter yang ditetapkan. (4) Pengukuran waktu: dicatat sebagai rerata tiga ulangan untuk mengurangi variansi. (5) Analisis perbandingan: tabel kontingensi dibuat untuk masing-masing pendekatan dan diuji signifikansi perbedaan menggunakan Wilcoxon signed-rank test untuk data berpasangan. (6) Sintesis rekomendasi: pemetaan kategori kerentanan ke pendekatan yang lebih efektif dan rumusan strategi hibrid.

4. HASIL YANG DIHARAPKAN

Bagian ini memaparkan hasil yang diharapkan berdasarkan tinjauan literatur dan karakteristik kedua pendekatan. Angka spesifik akan dikonfirmasi melalui eksperimen aktual; pemaparan di bawah ini berfungsi sebagai prediksi berlandaskan teori dan studi sejenis pada dataset terkait.

4.1 Prediksi Pola Akurasi per Pendekatan

Berdasarkan karakter algoritmik kedua pendekatan, static analysis dengan Slither diprediksi mencapai recall yang lebih tinggi pada kategori yang memiliki pola sintaksis kuat terutama integer overflow, access control missing modifier, dan unchecked return values namun dengan precision yang lebih rendah karena kecenderungan melaporkan false positive pada konstruksi yang secara semantik aman. Dynamic analysis dengan Foundry dan Echidna diprediksi mencapai precision yang lebih tinggi karena pelanggaran property hanya tercatat ketika benar-benar terjadi pada eksekusi konkret, namun dengan recall yang lebih rendah pada kategori yang memerlukan urutan transaksi spesifik (misalnya cross-function reentrancy) jika property yang ditulis tidak menangkap pola tersebut. Liu dkk. [1] dan Tolmach dkk. [3] telah memperlihatkan pola serupa pada dataset yang lebih luas, sehingga prediksi ini berdiri di atas dasar empirik yang konsisten.



Gbr. 4. Prediksi rentang metrik akurasi (precision, recall, F1) per pendekatan.

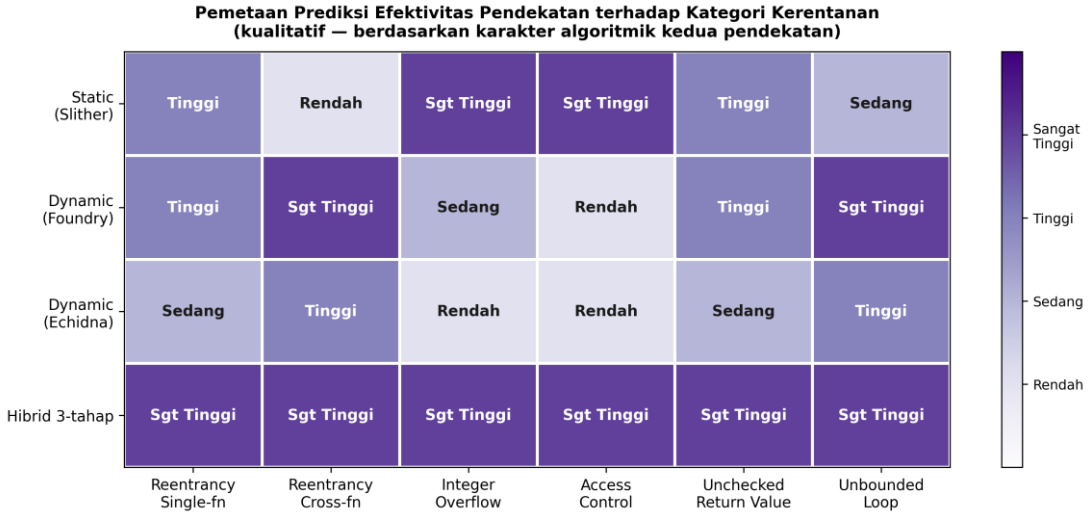
Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 204-215
p-ISSN: 2962-3693 | e-ISSN: 2962-7788

4.2 Prediksi Efisiensi Waktu Komputasi

Static analysis diprediksi memerlukan waktu paling singkat per kontrak biasanya dalam orde detik karena tidak melibatkan eksekusi. Dynamic analysis dengan Foundry pada konfigurasi 5.000 fuzz runs diperkirakan berkisar puluhan detik per kontrak, sedangkan Echidna pada anggaran satu menit per kontrak akan mendekati batas atas konfigurasi tersebut. Selain waktu eksekusi murni, dynamic analysis menambahkan effort konfigurasi berupa penulisan property dan invariant yang mungkin memerlukan jam-orang tambahan, terutama bagi tim yang baru mengadopsi alat tersebut. Trade-off ini menjadi pertimbangan penting saat memilih strategi pengujian dengan anggaran waktu terbatas.

4.3 Pemetaan Efektivitas per Kategori Kerentanan

Lima kategori kerentanan diperkirakan memiliki pola efektivitas yang berbeda. Reentrancy klasik (single-function) terdeteksi baik oleh Slither karena pola pemanggilan eksternal sebelum update state mudah dikenali secara statis; reentrancy lintas-fungsi membutuhkan dynamic analysis dengan invariant pada saldo total. Integer overflow pada Solidity warisan (< 0.8) terdeteksi oleh kedua pendekatan, dengan static lebih cepat. Access control missing modifier hampir selalu terdeteksi oleh Slither dengan presisi tinggi karena polanya sintaksis. Unchecked return values terdeteksi baik oleh Slither namun dynamic analysis lebih dapat membuktikan konsekuensi runtime. Unbounded loop sering memerlukan dynamic analysis untuk memicu kondisi gas exhaustion secara konkret. Pemetaan ini, jika terverifikasi, menjadi dasar matriks rekomendasi pemilihan alat.



Gbr. 5. Pemetaan prediksi efektivitas pendekatan terhadap kategori kerentanan.

TABEL II
PREDIKSI RENTANG METRIK BERDASARKAN LITERATUR (UNTUK DIVERIFIKASI EKSPERIMEN)

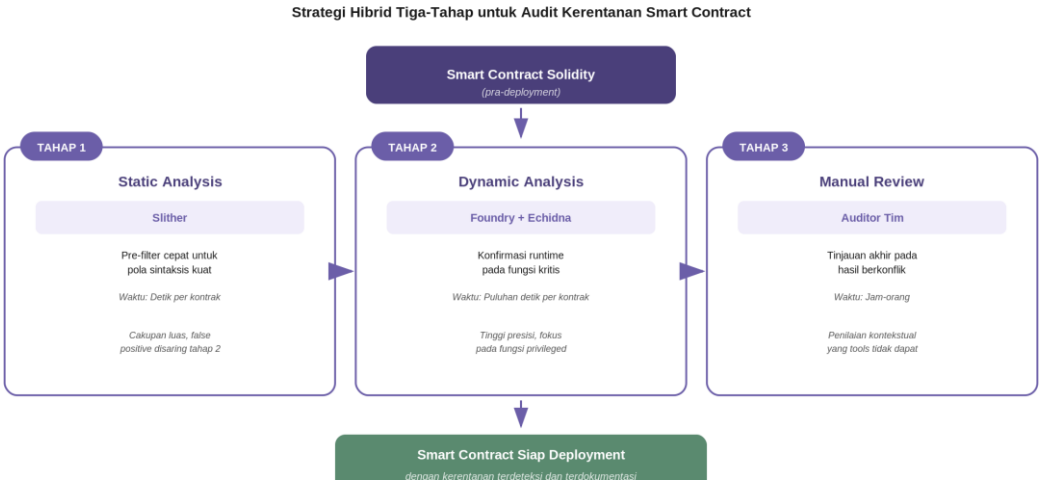
Pendekatan	Precision (estimasi)	Recall (estimasi)	F1 (estimasi)	Waktu/Kontrak	Effort Setup
Static (Slither)	Sedang Tinggi	Tinggi	Sedang Tinggi	Detik	Rendah
Dynamic (Foundry)	Tinggi	Sedang	Sedang Tinggi	Puluhan detik	Sedang Tinggi
Dynamic (Echidna)	Tinggi	Sedang	Sedang	≤ 1 menit	Sedang Tinggi
Hibrid 3-tahap	Tinggi	Tinggi	Tinggi	Variabel	Tinggi

DOI: https://doi.org/10.59025/a948780c | https://jtech.onlinelibrary.id

Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 204-215
p-ISSN: 2962-3693 | e-ISSN: 2962-7788

4.4 Strategi Hibrid yang Diusulkan

Berdasarkan pemetaan di atas, strategi hibrid yang diusulkan adalah tiga-tahap. Tahap pertama menggunakan static analysis sebagai pre-filter untuk menyaring temuan dengan pola sintaksis kuat dengan biaya komputasi minimal. Tahap kedua menggunakan dynamic analysis pada fungsi-fungsi kritis (mengelola dana, memiliki modifier privileged, atau menjalankan perulangan) untuk mengonfirmasi temuan static dan menangkap bug runtime. Tahap ketiga menyertakan tinjauan manual oleh auditor pada hasil berkonflik antara static dan dynamic. Pendekatan tiga-tahap ini sejalan dengan praktik audit profesional yang diadopsi oleh tim audit blockchain industri.



Gbr. 6. Strategi hibrid tiga-tahap untuk audit kerentanan smart contract.

4.5 Keterbatasan Rancangan dan Mitigasi

Tiga keterbatasan rancangan perlu diakui sejak awal. Pertama, dataset 25 kontrak relatif kecil dibandingkan korpus puluhan ribu kontrak yang dipakai studi seperti Liu dkk. [1]; mitigasi dilakukan melalui pemilihan kontrak yang representatif untuk lima kategori target dan validasi ground truth yang ketat. Kedua, hasil bergantung pada versi alat dan dapat berubah ketika Slither, Foundry, atau Echidna merilis versi baru laporan akan menyertakan hash commit alat yang digunakan untuk replikasi. Ketiga, dynamic analysis sangat bergantung pada kualitas property dan invariant yang ditulis; mitigasi melalui peer-review property antar peneliti dan dokumentasi rasionalitas penulisan setiap property dalam lampiran.

5. KESIMPULAN

Artikel ini telah memaparkan rancangan eksperimen perbandingan static analysis dan dynamic analysis untuk deteksi kerentanan smart contract Solidity yang menargetkan tiga pertanyaan penelitian: perbandingan akurasi, perbandingan efisiensi, dan pemetaan efektivitas per kategori kerentanan. Rancangan menggabungkan tiga alat utama (Slither, Foundry, Echidna), dataset 25 kontrak yang mencakup lima kategori kerentanan, dan metrik standar information retrieval. Kontribusi yang diharapkan adalah protokol eksperimen yang dapat direplikasi, taksonomi efektivitas alat per kategori, serta kerangka rekomendasi strategi hibrid bagi pengembang Web3 Indonesia.

Pelaksanaan eksperimen berikutnya direncanakan dalam tiga bulan dengan tahapan: bulan pertama untuk persiapan dataset dan instalasi alat, bulan kedua untuk eksekusi eksperimen static dan dynamic, serta bulan ketiga untuk analisis dan penulisan laporan akhir. Saran penelitian lanjutan mencakup empat arah konkret: (1) memperluas alat dengan menyertakan Mythril dan Manticore untuk diversifikasi pembanding pada static analysis; (2) menambahkan kerentanan tingkat lanjut seperti flash

DOI: https://doi.org/10.59025/a948780c | https://jtech.onlinelibrary.id

loan attack dan oracle manipulation yang memerlukan setup eksperimen berbeda; (3) mengukur false positive rate secara mendalam dengan klasifikasi alasan kemunculan; dan (4) mengintegrasikan pendekatan berbasis Large Language Model sebagaimana dipelajari Boi dan Esposito [4], Kado dan Tsuchiya [5], dan Mane dkk. [6] sebagai layer ketiga di samping static dan dynamic.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Universitas Dharmawacana Metro atas dukungan akademik selama penyusunan rancangan penelitian ini, serta kepada komunitas open-source Slither, Foundry, dan Echidna atas dokumentasi yang lengkap sehingga rancangan eksperimen yang reprodusibel dapat disusun.

KONTRIBUSI PENULIS

Andreas Perdana: konseptualisasi rancangan eksperimen, perumusan taksonomi kerentanan, dan penulisan draf awal. Febri Sugandi: kurasi dataset, validasi ground truth, dan instalasi lingkungan eksperimen. Ridwan Yusuf: supervisi metodologi, perumusan metrik evaluasi, korespondensi, dan revisi akhir. Untoro Apsiswanto: review naskah, analisis posisi penelitian terhadap literatur, dan penyempurnaan struktur kajian pustaka.

PERNYATAAN KONFLIK KEPENTINGAN

Penulis menyatakan tidak terdapat konflik kepentingan finansial maupun non-finansial yang dapat memengaruhi objektivitas rancangan penelitian ini. Pemilihan alat Slither, Foundry, dan Echidna berdasarkan kriteria ketersediaan open-source dan adopsi komunitas, bukan afiliasi vendor.

REFERENSI

[1] Z. Liu et al., "Combining Graph Neural Networks With Expert Knowledge for Smart Contract Vulnerability Detection," IEEE Transactions on Knowledge and Data Engineering, 2023, doi: 10.1109/TKDE.2021.3095196.

[2] S. S. Kushwaha et al., "Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract," IEEE Access, 2022, doi: 10.1109/ACCESS.2021.3140091.

[3] P. Tolmach et al., "A Survey of Smart Contract Formal Specification and Verification," ACM Computing Surveys, 2023, doi: 10.1145/3464421.

[4] B. Boi dan C. Esposito, "Prompt Engineering vs. Fine-Tuning for LLM-Based Vulnerability Detection in Solana and Algorand Smart Contracts," dalam Proc. 2025 7th International Conference on Blockchain Computing and Applications (BCCA), 2025, doi: 10.1109/bcca66705.2025.11229457.

[5] C. Kado dan T. Tsuchiya, "Initial Attempt at Automated Vulnerability Fixing of Obfuscated Smart Contracts Using a Large Language Model," dalam Proc. 2024 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), 2024, doi: 10.1109/icce-asia63397.2024.10773661.

[6] S. Mane, J. Bhortake, V. Wankhade, dan F. Kazi, "Automated IoT Firmware Vulnerability Detection Using Large Language Models," Preprints, 2025, doi: 10.20944/preprints202510.0166.v1.

[7] H. Taherdoost, "Smart Contracts in Blockchain Technology: A Critical Review," Information, vol. 14, no. 2, hal. 117, 2023, doi: 10.3390/info14020117.

[8] S. Y. Lin, L. Zhang, J. Li, L. L. Guo, dan Y. Lin, "A survey of application research based on blockchain smart contract," Wireless Networks, 2022, doi: 10.1007/s11276-021-02874-x.

[9] L. Ante, "The Non-Fungible Token (NFT) Market and Its Relationship with Bitcoin and Ethereum," FinTech, 2022, doi: 10.3390/fintech1030017.

[10] N. Karandikar, A. Chakravorty, dan C. Rong, "Blockchain based transaction system with fungible and non-fungible tokens for a community-based energy infrastructure," Sensors, vol. 21, no. 11, hal. 3822, 2021, doi: 10.3390/s21113822.

[11]J. Wu et al., "Who Are the Phishers? Phishing Scam Detection on Ethereum via Network Embedding," IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, doi: 10.1109/TSMC.2020.3016821.

[12]J. Arcenegui, R. Arjona, R. Roman, dan I. Baturone, "Secure combination of IoT and blockchain by physically binding IoT devices to smart non-fungible tokens using PUFs," Sensors, vol. 21, no. 9, hal. 3119, 2021, doi: 10.3390/s21093119.

[13]E. Moayyed, C. Anumba, dan A. Morteza, "Systematic analysis of large language models for automating document-to-smart contract transformation," Automation in Construction, 2025, doi: 10.1016/j.autcon.2025.106209.

[14]D. Kirli et al., "Smart contracts in energy systems: A systematic review of fundamental approaches and implementations," Renewable and Sustainable Energy Reviews, 2022, doi: 10.1016/j.rser.2021.112013.

[15]H. Hamledari dan M. Fischer, "Construction payment automation using blockchain-enabled smart contracts and robotic reality capture technologies," Automation in Construction, 2021, doi: 10.1016/j.autcon.2021.103926.

[16]A. P. Balcerzak, E. Nica, E. Rogalska, M. Poliak, T. Klieštik, dan O.-M. Sabie, "Blockchain Technology and Smart Contracts in Decentralized Governance Systems," Administrative Sciences, vol. 12, no. 3, hal. 96, 2022, doi: 10.3390/admsci12030096.

[17]G. D. Ritterbusch dan M. R. Teichmann, "Defining the Metaverse: A Systematic Literature Review," IEEE Access, 2023, doi: 10.1109/ACCESS.2023.3241809.

[18]K. B. Wilson, A. Karg, dan H. Ghaderi, "Prospecting non-fungible tokens in the digital economy: Stakeholders and ecosystem, risk and opportunity," Business Horizons, 2022, doi: 10.1016/j.bushor.2021.10.007.

[19]B. Hammi, S. Zeadally, R. Adja, M. El Madhoun, dan J. Nebhen, "Non-Fungible Tokens: A Review," IEEE Internet of Things Magazine, 2023, doi: 10.1109/IOTM.001.2200244.

[20]M. Turki, S. Cheikhrouhou, B. Dammak, M. Baklouti, R. Mars, dan A. Dhahbi, "NFT-IoT Pharma Chain: IoT Drug traceability system based on Blockchain and Non Fungible Tokens (NFTs)," Journal of King Saud University - Computer and Information Sciences, 2023, doi: 10.1016/j.jksuci.2022.12.016.

[21]L. Ante, "Non-fungible token (NFT) markets on the Ethereum blockchain: temporal development, cointegration and interrelations," Economics of Innovation and New Technology, 2023, doi: 10.1080/10438599.2022.2119564.

[22]L. Cao, "Decentralized AI: Edge Intelligence and Smart Blockchain, Metaverse, Web3, and DeSci," IEEE Intelligent Systems, 2022, doi: 10.1109/MIS.2022.3181504.

[23]H. Albayati, S. K. Kim, dan J. J. Rho, "Accepting financial transactions using blockchain technology and cryptocurrency: A customer perspective approach," Technology in Society, 2020, doi: 10.1016/j.techsoc.2020.101320.

[24]L. Ma, Z. Yang, Z. Bu, Q. Lao, dan W. Yang, "Statement Recognition of Access Control Policies in IoT Networks," Sensors, vol. 23, no. 18, hal. 7935, 2023, doi: 10.3390/s23187935.

[25]I. Ghosh, S. Alfaro-Cortes, M. Galindo, dan J. M. Sanchez-Garcia, "Prediction and interpretation of daily NFT and DeFi prices dynamics: Inspection through ensemble machine learning & XAI," International Review of Financial Analysis, 2023, doi: 10.1016/j.irfa.2023.102558.

[26]S. Casale-Brunet, P. Ribeca, P. Doyle, dan M. Mattavelli, "Networks of Ethereum Non-Fungible Tokens: A graph-based analysis of the ERC-721 ecosystem," dalam Proc. 2021 IEEE International Conference on Blockchain (Blockchain), 2021, doi: 10.1109/Blockchain53845.2021.00033.

[27]J. Köpke, G. Meroni, dan M. Salnitri, "Towards Automated Handling of Object State Exceptions in Smart Contracts," dalam Lecture Notes in Business Information Processing, 2025, doi: 10.1007/978-3-032-02936-2_13.

[28]Y. Liu, S. Chan, J. Chu, Y. Zhang, dan Y. Chen, "When LLM Meets Simplicial Complex: A Novel Graph Prompt Learning on Ethereum Transaction Networks," dalam Proc. 2025 IEEE International Conference on Data Mining (ICDM), 2025, doi: 10.1109/icdm65498.2025.00148.

[29]J. Truby, R. D. Brown, A. Dahdal, dan I. Ibrahim, "Blockchain, climate damage, and death: Policy interventions to reduce the carbon emissions, mortality, and net-zero implications of non-fungible tokens and Bitcoin," Energy Research & Social Science, 2022, doi: 10.1016/j.erss.2022.102499.

[30]L. Wu, K. Lu, F. Xue, dan W. Lu, "A blockchain non-fungible token-enabled 'passport' for construction waste material cross-jurisdictional trading," Automation in Construction, 2023, doi: 10.1016/j.autcon.2023.104783.