BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi *blockchain* telah berkembang sejak Bitcoin dan terus berlanjut hingga munculnya Ethereum dengan konsep *smart contract*. *Smart contract* adalah program berjalan di atas jaringan *blockchain* yang memungkinkan melakukan perjanjian otomatis tanpa perantara pihak ketiga (Buterin, 2014). Kemampuan ini membuat *smart contract* banyak digunakan dalam aplikasi terdesentralisasi (*decentralized applications*/DApps), terutama di bidang keuangan terdesentralisasi (*Decentralized Finance*/DeFi).

Seiring berjalannya waktu, *smart contract* telah mengelola aset benilai miliaran dolar yang dapat menarik perhatian penyerang. *Smart contract* bersifat *immutable* yang artinya tidak dapat diubah setelah di-*deploy* ke *blockchain*, sehingga jika ada kerentanan pada kodenya, maka kerentanan itu tidak bisa langsung diperbaiki (Zheng et al., 2019). Akibatnya, platform *blockchain* seperti Ethereum, EOS, dan TRON dilaporkan mengalami kerugian lebih dari 3,3 miliar dolar akibat celah keamanan pada *smart contract* (Ghiyami Pour et al., 2025).

Salah satu kerentanan paling berbahaya bagi *smart contract* adalah *reentrancy*. *Reentrancy* muncul Ketika kontrak penyerang memanggil Kembali fungsi pada kontrak korban sebelum selesai dijalankan. Penyerang memanfaatkan celah waktu itu untuk menguras saldo kontrak sebelum system memperbarui state (Atzei et al., 2017). The DAO Attack pada 2016 membawa *reentrancy* ke perhatian publik. Penyerang mencuri sekitar 3,6 juta Ether, senilai 60 juta dolar saat itu, lewat satu celah *reentrancy*(Durieux et al., 2020). Kapitalisasi pasar Ethereum kemudian terbelah menjadi Ethereum dan Ethereum Classic.

Meski sudah hampir 1 dekade berlalu, *reentrancy* terus mengeksploitasi kontrak baru hingga sekarang. Studi sistematis mengenai literatur dari tahun 2016 hingga 2023 menunjukkan kerugian akibat *reentrancy* mencapai sekitar 350 juta dolar AS. (Ghiyami Pour et al., 2025). Grim Finance kehilangan 30 juta dolar pada 2021. Fei Protokol dan Rari Capital kehilangan 80 juta dolar pada 2022. Curve Finance kehilangan 50 juta dolar pada 2023.

Komunitas riset dan industri mengembangkan berbagai alat analisis keamanan untuk mendeteksi kerentanan *reentrancy* sebelum *smart contract* di-*deploy*. Alat audit dibagi menjadi 3 pendekatan yaitu *static analysis*, *dynamic analysis* dan *fuzzing*. *Static analysis* unggul dari ketiganya karena menganalisis kode sumber tanpa menjalankan programnya. Slither, Slohint dan SmartCheck merupakan alat berbasis *static analysis*.

Slither dan Aderyn merupakan 2 alat *static analysis* yang dapat diuji lebih dalam. Trail of Bits mengembangkan Slither menggunakan Python dan memanfaatkan representasi intermediate SlithIR berbasis *Single* Static Assignment (SSA)(Feist et al., 2019). Cyrin mengembangkan Aderyn menggunakan Rust sebagai alat audit yang lebih baru. Aderyn bekerja berdasarkan analisis Abstract Syntax Tree (AST) dari kode solidity dan mendukung ekosistem pengembangan *smart contract* modern. Kedua alat tersebut memiliki pendekatan internal yang berbeda secara teknis meski dikategorikan sebagai *static analysis*. Perbedaan itu yang membuat perbandingan keduanya relevan secara ilmiah.

Meski sudah ada beberapa studi yang membandingkan alat analisis *smart contract*, publikasi akademik yang membandingkan Slither dan Aderyn secara langsung khususnya untuk deteksi *reentrancy* belum ditemukan. Umumnya membandingkan Slither dengan alat lama seperti Oyente, Osiris, Mythril (Ottati et al., 2023). Kesenjangan penelitian itu mendorong untuk membandingkan performa Slither dan Aderyn dalam mendeteksi *reentrancy* pada *smart contract* Solidity, menggunakan metrik standar klasifikasi biner yaitu Precision, Re*call* dan F1-Score. Diharapkan dapat memberi panduan bagi pengembang dan auditor *smart contract* dalam memilih alat sesuai kebutuhan dan konteks penggunaannya.

1.2 Rumusan Masalah

Penelitian ini merumuskan 3 masalah yang didasari latar belakang di atas:

1. Bagaimana performa Slither dalam mendeteksi kerentanan *reentrancy* pada *smart contract* Solidity berdasarkan metrik Precision, Re*call*, dan F1-Score?

2. Bagaimana performa Aderyn dalam mendeteksi kerentanan *reentrancy* pada *smart contract* Solidity berdasarkan metrik Precision, Re*call*, dan F1-Score?
3. Bagaimana perbandingan performa Slither dan Aderyn dalam mendeteksi kerentanan *reentrancy* pada *smart contract* Solidity?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah di atas, penelitian ini menetapkan 3 tujuan:

1. Mengukur performa Slither dalam mendeteksi kerentanan *reentrancy* pada *smart contract* Solidity menggunakan metrik Precision, Re*call*, dan F1-Score.
2. Mengukur performa Aderyn dalam mendeteksi kerentanan *reentrancy* pada *smart contract* Solidity menggunakan metrik Precision, Re*call*, dan F1-Score.
3. Membandingkan performa Slither dan Aderyn dalam mendeteksi kerentanan *reentrancy* untuk mengetahui alat mana yang lebih efektif untuk keperluan audit *smart contract*.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat baik secara teoritis maupun praktis, di antaranya:

1. Manfaat Teoritis
    a. Penelitian ini memberikan data empiris perbandingan performa Slither dan Aderyn yang belum ada dalam literatur ilmiah sebelumnya.
    b. Penelitian ini memperkaya kajian keamanan *smart contract*, khususnya evaluasi alat *static analysis* berbasis metrik klasifikasi biner.
2. Manfaat Praktis

a. Pengembang *smart contract* bisa menjadikan hasil penelitian ini referensi untuk memilih alat audit yang paling efektif mendeteksi *reentrancy* sebelum kontrak di-*deploy* ke blockchain.
b. Auditor keamanan blockchain mendapat perbandingan objektif yang mendukung keputusan dalam proses audit *smart contract* secara profesional.
c. Peneliti selanjutnya bisa menjadikan hasil ini baseline untuk penelitian komparatif yang lebih luas, misalnya dengan menambahkan jenis kerentanan lain atau dataset yang lebih besar.

1.5 Batasan Penelitian

Agar penelitian ini terfokus dan terukur, ditetapkan batasan-batasan sebagai berikut:

1. Kerentanan yang dianalisis dibatasi pada jenis *reentrancy* vulnerability saja.
2. *Smart contract* yang diuji ditulis dalam bahasa pemrograman Solidity dan berjalan di atas jaringan Ethereum.
3. Alat yang dibandingkan terbatas pada Slither (versi 0.11.5) dan Aderyn (versi 0.6.8), keduanya menggunakan pendekatan *static analysis*.
4. Lingkungan pengujian menggunakan sistem operasi Windows 11 dengan WSL2 (Ubuntu) sebagai platform eksekusi alat.
5. Dataset yang digunakan adalah *Labelled Dataset* (*Reentrancy* & Safe) dari Wei dkk. (2024), yang dipublikasikan bersama paper "A *Comparative Evaluati*on of Automated Analysis for Solidity *Smart Contract*s" (IEEE, 2023).