Scout Bug Fighter on Substrate

Short description: Scout is a bug detection tool, a static analyzer currently available for ink! and Soroban (Stellar Network) smart contracts. After over one year of development, Scout is a stable tool providing assistance to both developers and auditors. We are ready for the next challenge: to provide support for the Substrate pallets, runtime code and node code.
Project Category / Type: Security Tool
Proponent: CoinFabrik
DOT address:
Requested allocation: U$152,000.-

Context of the proposal

At CoinFabrik, we believe that making a blockchain secure requires integrating vulnerability detection and best practices into the development lifecycle. As proud members of the Polkadot Assurance Legion (as auditors) and the Polkadot Alpha Program (as developers), we contend that raising the quality standards of projects before their first audit benefits the entire ecosystem.

This is why we developed Scout: Bug Fighter, a static-analyzer for detecting issues in ink! code. Simply put, Scout is an easy-to-install open source tool that integrates into the development process and pinpoints vulnerabilities in the ink! code under analysis.

In 2023, we focused on integrating Scout into the development cycle of ink! smart contracts. In collaboration with the LAFHiS Laboratory at the University of Buenos Aires, we conducted research on different detection technologies (static and dynamic analysis) applicable for detecting bugs in Rust-based smart contracts. With the support of Web3 Foundation Grants and Aleph Zero Ecosystem Grants, we built Scout, perfecting the tool up to its current state where it detects up to 33 security-issue classes in ink! code.

Realizing our vision of embedding secure best practices directly into developers' workflows, our solution empowers developers to identify vulnerabilities as they write their smart contracts. Scout can be used via a Command Line Interface (CLI), a VSCode Extension, and a GitHub Action, improving the developer experience by integrating into the secure development lifecycle of their projects.

However, the journey does not stop here: ink! smart contracts are not the only Rust code being developed to run in the ecosystem. We aim to extend Scout from smart contract development to Substrate.

This proposal aims to apply static analysis techniques to build issue detectors for Substrate pallets, runtime code and node code, while also constructing an open dataset of vulnerable test cases and an analysis framework to structure precision and recall efforts for the tool. This will pave the way for future work on detection techniques beyond the scope of static analysis.

Problem statement

Developing secure smart contracts and pallets is crucial as vulnerabilities can lead to significant losses or compromises within a blockchain network. Developers often inadvertently introduce vulnerabilities into their code, which, once deployed, can be exploited by attackers at any moment. For instance, a flaw in minting functions could potentially allow an attacker to mint an unlimited amount of tokens, leading to severe consequences for the entire parachain.

Detecting these vulnerabilities is not straightforward. The challenge of identifying security flaws in code has been an open problem for decades, as evidenced by the continuous introduction of bugs in C code despite 40 years of advancements in analysis tools. The goal of a good analysis tool is to trigger alerts for real vulnerabilities while minimizing false positives and not missing any critical flaws. Various computer science techniques have been applied to tackle this issue across different programming languages, including linting. Linting involves analyzing the syntax of a program to identify errors or bad practices.

Upon reviewing Polkadot's open-source stack, we found that while some tools offer basic bug detection for Substrate pallets, runtime code and node code, they often lack proper maintenance, user-friendly features, and clear documentation on the identified issues and their remediation process. This presents an opportunity for Scout to make a difference.

Team

CoinFabrik is a research and development company specialized in Web3, with a strong background in cybersecurity. Founded in 2014, we have worked on over 350 blockchain-related projects, EVM based and also for Solana, Algorand, and Polkadot. Beyond development, we

offer security audits through a dedicated in-house team of senior cybersecurity professionals, currently working on code in Substrate, Solidity, Clarity, Rust, Soroban and TEAL.

Our team has an academic background in computer science and mathematics, with work experience focused on cybersecurity and software development, including academic publications, patents turned into products, and conference presentations. Furthermore, we have an ongoing collaboration on knowledge transfer and open-source projects with the University of Buenos Aires.

Assigned Team and Roles

The time allocation for each team member is detailed within each milestone. Below, we outline the specific roles assigned to each team member.

Project Manager (PM): Responsible for the planning, execution, and completion of the project, ensuring it meets goals, stays on schedule, and adheres to budget while coordinating the team and managing stakeholder communication. **Matias**, our Project Manager for this project, has extensive experience managing similar initiatives, including the successful Scout version for ink! and Soroban.

Technical Leader (TL): Responsible for the technical direction of the project, code quality and scalability, guiding the development team, and making key technical decisions. The TL also contributes as a full-time developer. **Victor**, our Technical Leader for this project, has prior experience in managing similar projects and has a strong background in developing blockchain-based security solutions, as well as Substrate-based ones.

Security Advisor, Senior Substrate Auditor: Responsible for providing expert security guidance on the project's direction, while analyzing development results (detectors and outputs) and assessing overall project outcomes to ensure robust security standards are met. **Aureliano**, our Security Advisor for this project, is a senior member of our Auditing Team, bringing extensive experience in cybersecurity, particularly within Rust-based blockchains and Substrate auditing, including our recent work with the LAOS Network.

Developers, Rust/Substrate: Responsible for implementing and maintaining the core detection functionalities of Scout for Substrate runtimes, pallets and node code. The Developers will be tasked with identifying and building issue test cases, building new static analysis detectors, refining existing ones, and integrating them into the tool's different features (CLI, VSCode, CI/CD). **José and Facundo**, our Developers for this project, have successfully built static

analysis tools in Rust, contributing to enhanced detection techniques for other Web3 environments, including ink! smart contracts.

Developer Relations (DevRel): Responsible for creating and maintaining project documentation, including usage guides, issue descriptions, and educational content, while gathering user feedback and providing technical support. The DevRel bridges the gap between the development team and end-users, ensuring effective communication, facilitating adoption, and showcasing new features through tutorials, webinars, and workshops to enhance the user experience and tool engagement. **Arturo**, our DevRel for this project and a graduate alumnus of the Polkadot Blockchain Academy at Berkeley, brings valuable community feedback and a proven track record in similar roles and tasks for security tooling.

Proposal objectives

Our objectives are to construct a Data Set of vulnerabilities and best practices deviations in Substrate pallets, runtime code and node code, design and implement static analysis detectors, and build a working tool that scales to meet the demands of the ecosystem.

Scout will enable developers to seamlessly integrate security into the development lifecycle of their Substrate projects. By utilizing our detection features—such as hover-over warnings in the VSCode extension, vulnerability reports generated through our CLI, and CI/CD integrations via the Scout GitHub Action—developers can effectively identify and resolve security issues, leading to higher-quality code.

Milestones

We have broken down our proposal into a series of milestones. For each milestone, we describe the tasks and deliverables.

Milestone 1: Initial Vulnerability Data Set and PoC Detectors (4 weeks)

Tasks:

- Analyze 10+ audited Substrate pallets, runtime code and node code, preparing a repository with annotated versions of both their vulnerable and fixed states. These annotations will highlight the vulnerabilities, their locations within the code, and their classification.

- Compile a list of top/relevant vulnerabilities specific to Substrate pallets, runtime and node code, using audit reports and other publicly available references in the ecosystem (e.g., this blogpost).
- Identify security issues that can be effectively addressed with static analysis detectors.
- Adapt existing static analyzer detectors for Rust and ink! (as outlined in Context of the Proposal), currently integrated into Scout, to detect the identified security issues within Substrate pallets, runtime code and node code.

Deliverables:

- Data Set. An open-source GitHub repository containing annotated Substrate pallets, runtime code and node code, accompanied by a detailed document listing vulnerability classes and their locations within the code. The dataset will also be uploaded to Hugging Face data sets.
- Proof-of-concept version of the detectors. Building on our experience with Scout for ink! and Soroban, we aim to detect 4 vulnerability classes. The tool will be delivered as source code in the repository.

Team:

- 1 Project Manager (PM) 80h
- 1 Technical Leader (TL) 160h
- 1 Security Consultant (Senior Substrate Auditor) 40h
- 2 Developers (Rust / Substrate) 320h
- 1 Developer Relations (DevRel) 80h

FTEs: 4.25 / 680 hours

Total Budget for Milestone 1: U$38,000.-

Milestone 2: Extended Vulnerability Data Set and Prototype Detectors with Precision and Recall (4 weeks)

Tasks:

- Extend the Data Set of vulnerable-projects repository with more annotated pallets, runtimes, and node code (a second set analyzing at least 10 additional pallets/runtimes/nodes).
- Extend the list of security issues in Substrate and associated test cases by analyzing the second set of audited pallets/runtimes/nodes.
- Develop new static analysis detectors for issues identified in the first and current milestone. Given our experience developing Scout for ink! and Soroban we aim to deliver at least 5 new detectors.

- Create a framework for running analysis tools, similar to SmartBugs but for Rust, to structure the subsequent precision and recall efforts on all detectors built. This framework should run a process that receives a bug detection tool (our static analyzer), a set of audited substrate projects annotated with vulnerabilities, and produces a report. This report shall include: what are the vulnerabilities that were detected, what detections were false positives, and what are the undetected vulnerabilities.
- Analyze precision and recall of static analysis detectors, using the framework developed against the Data Set of audited Substrate projects with annotated vulnerabilities, to find what (problematic) vulnerability classes need better detection.

Deliverables:

- Prototype version of the detectors. Given our experience developing Scout for ink! and Soroban we aim to include detection of at least 5 new issue classes. Delivered as source code in the repository.
- Extended Data Set. Extended repository of vulnerable pallets, also extending list of vulnerabilities. Updated Hugging Face data set.
- Open Source Framework, for running analysis tools on Rust code, publicly available on CoinFabrik's GitHub repository.
- Detector's evaluation report on benchmark Data Set. List of suggested vulnerability classes that appear as false negatives in the report, or have a high rate of false positives. (See an example of the evaluation report for Scout for Soroban)

Team:

- 1 Project Manager (PM) 80h
- 1 Technical Leader (TL) 160h
- 1 Security Consultant (Senior Substrate Auditor) 40h
- 2 Developers (Rust / Substrate) 320h
- 1 Developer Relations (DevRel) 80h

FTEs: 4.25 / 680 hours

Total Budget for Milestone 2: U$38,000.-

Milestone 3: Prototype Tool Integration with CLI, VSCode, and CI/CD, Documentation (4 weeks)

Tasks:

- Integrate and Test the Prototype Tool: Implement the following features:
    - Command Line Interface (CLI): Enable various output report formats (HTML, Markdown, SARIF, JSON) and provide detector filter options.

    - VSCode Extension: Display hover error indicators and allow users to enable/disable detectors using inline macros.
    - CI/CD Scout GitHub Action: Make Scout action available on the GitHub Marketplace to seamlessly integrate the tool into development pipelines, providing markdown reports in GitHub Issues and Pull Requests.
- Develop and improve detectors. Address problematic issues identified in Milestone 2.
- Document issue classes and associated detectors.

Deliverables:

- A prototype tool that integrates built detectors with a CLI, a VSCode extension, and a CI/CD GitHub Action. (See existing VSCode extension and GitHub Action for ink! and Soroban)
- Additional or improved detectors for problematic issues identified in Milestone 2. Given our experience developing and improving Scout for ink! and Soroban, we aim to improve or further develop 3 detectors.
- Comprehensive integration tests for all detectors and features.
- A Documentation Site (using Docusaurus or GitBook) detailing tool usage and an initial set of detectors, including nine documented detectors developed in Milestones 1 and 2. (See the documentation pages for Scout on ink! and Soroban)
- A public project GitHub repository and website, along with an alpha tool release for selected projects and users.

Team:

- 1 Project Manager (PM) 80h
- 1 Technical Leader (TL) 160h
- 1 Security Consultant (Senior Substrate Auditor) 40h
- 2 Developers (Rust / Substrate) 320h
- 1 Developer Relations (DevRel) 80h

FTEs: 4.25 / 680 hours

Total Budget for Milestone 3: U$38,000.-

Milestone 4: Final Precision and Recall Evaluation & Full Tool Release (4 weeks)

Tasks:

- Conduct final precision and recall evaluation for detectors using the framework developed in Milestone 2 against Data Set of audited Substrate projects with annotated

vulnerabilities, to find what (problematic) remaining vulnerability classes need better detection.
- Final detector improvements based on precision and recall evaluation.
- Document remaining detectors and issue classes.
- Create video tutorials on tool usage and issue classes.
- Create Release webinar and social media communications.

Deliverables:

- Final precision and recall evaluation report. Responsible disclosure of any sensible findings to their corresponding projects.
- Improved detectors based on evaluation results. Given our experience developing Scout for ink! and Soroban, we aim to improve or develop 2 detectors after this final precision and recall.
- Fully integrated tool with CLI, VSCode Extension, and/or CI/CD GitHub Action.
- Public release of the tool with full documentation, publicly available on documentation sites (Docusaurus or GitBook)(See documentation examples here and here).
- Video tutorials on how to use the tool, along with one video tutorial for each issue detected by the tool. Given our experience developing Scout for ink! and Soroban, we aim to publish between 10 and 15 video tutorials on CoinFabrik's YouTube channel. (See Scout video tutorials for other blockchain here).
- Release Webinar.
- Posts on CoinFabrik's social media.

Team:

- 1 Project Manager (PM) 80h
- 1 Technical Leader (TL) 160h
- 1 Security Consultant (Senior Substrate Auditor) 40h
- 2 Developers (Rust / Substrate) 320h
- 1 Developer Relations (DevRel) 80h

FTEs: 4.25 / 680 hours

Total Budget for Milestone 4: U$38,000.-

Find a condensed version of milestones, deliverables, resources, average hour cost and cost breakdown per milestones over here

Commercial terms

CoinFabrik accepts PAL commercial agreement, of 50% upfront payment for each proposed milestone, with the remaining 50% due upon completion of such milestone.

References

1. Scouting Vulnerabilities and Detection Techniques in Substrate https://forum.polkadot.network/t/scouting-vulnerabilities-and-detection-techniques-in-substrate/6609#scouting-vulnerabilities-and-detection-techniques-in-substrate-1
2. Common Vulnerabilities in Substrate Polkadot Development https://forum.polkadot.network/t/common-vulnerabilities-in-substrate-polkadot-development/3938
3. Verifying Rust - Exploring Verification Options for Substrate https://forum.polkadot.network/t/verifying-rust-exploring-verification-options-for-substrate/2109
4. Research on Analysis Techniques and Tools for Identifying Issues in ink! Smart Contracts https://github.com/CoinFabrik/web3-grant/blob/f37be295455ce4a86ce2d0b6fb94c55128fc71c8/detectors/README.md
5. Scout https://github.com/CoinFabrik/scout
6. Scout Actions https://github.com/CoinFabrik/scout-actions
7. Solodit - Finding Aggregation Platform https://solodit.xyz/
8. Smart Contract and DeFi Security Tools: Do They Meet the Needs of Practitioners? https://arxiv.org/abs/2304.02981
9. Static Application Security Testing (SAST) Tools for Smart Contracts: How Far Are We? https://arxiv.org/abs/2404.18186
10. Securify: Practical Security Analysis of Smart Contracts https://arxiv.org/pdf/1806.01143
11. SmartBugs 2.0: An Execution Framework for Weakness Detection in Ethereum Smart Contracts https://arxiv.org/abs/2306.05057