Advanced Topics in Software Engineering
CSE - 6324 - Section 001

TEAM 4

INCEPTION
(Written Deliverable)

Team Members

Suvarna, Arjun - 1002024437
Sinari, Navina - 1002072310
Palnati, Netra - 1002030626
Waje, Sanjana - 1002069940

I. Project Objective

Slither is a static analysis framework that provides rich information about Ethereum smart contracts [1]. Slither's plugin architecture lets you integrate new detectors from the command line [2].

We propose to add 3 new detectors using the Python API [3]. The first detector will be targeted toward the improper usage of the ecrecover() function in solidity smart contracts [4]. The ecrecover function can be used to process meta transactions in smart contracts and verifies signed data coming from someone other than the transaction signer [5], but due to the nature of public key private key signatures, this function is vulnerable to replay attacks [11]. The proposed detector will detect usage of the ecrecover function without nonce and flag it as a security vulnerability.

The second detector will detect API Keys stored as plain text within the solidity code (SWC 136) [6]. Hardcoded credentials present a grave security risk and compromise the security setup associated with the keys [6].

The third detector will detect passwords hardcoded into the solidity code (SWC 136) [6]. Hardcoded passwords in solidity smart contracts are not private and can be seen by third parties due to the public nature of smart contracts [7].

II. Features

i. Increased Security:- The improper use of an ecrecover detector will provide smart contract developers who have explicit key verification with increased security in their smart contracts by ensuring adequate security measures are in place.

ii. Increased Data Protection:- The hard-coded API key and hard-coded Password detectors will provide greater data security to smart contract developers who have sensitive information in their solidity code by ensuring no API keys or passwords are stored in plain text.

III. Risks

i. Incomplete rule definition:- The detector may overlook a few issues because there are incomplete rules and checks.
There is a 30% probability that this issue might occur, and we would need 10 hours to fix it. Hence, risk exposure would be 3 extra hours.

ii. False Positives:- The detector might detect that there is a data leak that does not exist.
We need 4 hours to resolve this problem, which has a 20% chance of happening. Thus, 0.8 hours of risk exposure.

iii. False Negatives:- The detector may not detect that there is a data leak that exists.
There is a 10% chance that this problem would arise, and it would take 4 hours to fix. Hence, risk exposure would be 0.4 extra hours.

IV. Planning

i. Iteration 1:
Define rules for ecrecover improper usage without nonces. The exact conditions that constitute a security vulnerability using ecrecover which will be built into the new slither detectors.

ii. Iteration 2:
Define rules for hardcoded API keys and passwords that will be built into the new slither detectors.

iii. Final Deliverable:
Build the detectors based on the rules identified and defined in the previous steps. Improve the detector through iterative development and feedback.

V. Competitors

i. Slither:- The current stable build of slither (0.9.6) [9], has a robust set of public detectors like Dangerous usage of tx. origin and Unchecked low-level calls [10]. But Slither currently does not have a public detector for API Keys, Passwords, or a public detector for improper usage of the ecrecover function as described in the GitHub Issue #1950 [4].

ii. Mythril:- Mythril is a security analysis tool for Ethereum smart contracts [12]. It has a set of modules to detect vulnerabilities and issues in the Ethereum solidity code [13] like External Calls and Unprotected Ether Withdrawal [13]. But there are currently no modules to detect unprotected use of the ecrecover function or hardcoded credentials.

iii. MythX:- MythX is a tool that scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts[16]. However, it does not have a detector available for improper use of ecrecover or hardcoded API keys and passwords [17]. MythX is a closed-source tool.

VI. Customers and Users

i. Smart contract developers with explicit signature checks: Ethereum smart contracts generally use the ecrecover() function to verify the authenticity of the messages [14]. But users can elect to explicitly verify the signatures to save on gas. These users will benefit from the ecrecover() vulnerability check to ensure secure implementation.

ii. Wallet Providers: Applications for wallets like MetaMask and Toshi assist with signing transactions that are verified using ecrecover() [15].

iii. Smart contract developers with authenticated external calls: Developers who use external calls through an oracle with an authentication mechanism for the external calls [18].

VII. GitHub Repository

https://github.com/arjunsuvarna1/CSE6324_Team4_Fall23

VIII. References

[1] Feist, Josselin & Grieco, Gustavo & Groce, Alex. (2019). Slither: A Static Analysis Framework For Smart Contracts.

[2] Adding a new detector - https://github.com/crytic/slither/wiki/Adding-a-new-detector

[3] Python API - https://github.com/crytic/slither/wiki/Python-API

[4] Improper usage of ecrecover - https://github.com/crytic/slither/issues/1950

[5] What is ecrecover in Solidity? - https://soliditydeveloper.com/ecrecover

[6] Unencrypted Private Data On-Chain - https://swcregistry.io/docs/SWC-136/

[7] CWE-798: Use of Hard-coded Credentials - https://cwe.mitre.org/data/definitions/798.html

[8] Sensitive Information Disclosure in Smart Contracts - https://knowledge-base.secureflag.com/vulnerabilities/sensitive_information_exposure/sensitive_information_disclosure_smart_contracts.html

[9] Slither - https://github.com/crytic/slither

[10] Slither Detector Documentation - https://github.com/crytic/slither/wiki/Detector-Documentation

[11] SWC-121: Missing Protection against Signature Replay Attacks - https://swcregistry.io/docs/SWC-121/

[12] Mythril - https://github.com/Consensys/mythril

[13] Mythril Analysis Modules - https://mythril-classic.readthedocs.io/en/develop/module-list.html

[14] Ecrecover function in Solidity - https://www.educative.io/answers/what-is-an-ecrecover-function-in-solidity

[15] Multi-signatures for Ethereum -

[16] https://medium.com/dsys/now-open-source-friendly-multi-signatures-for-ethereum-d75ca5a0dc5c

[17] MythX - https://mythx.io/about/

[18] MythX Detectors - https://mythx.io/detectors/

[19] Make HTTP Request Using Your Solidity Smart Contract - https://medium.com/coinmonks/make-http-request-using-your-solidity-smart-contract-4f7173bd391c