Advanced Topics in Software Engineering
CSE 6324 - Section 001

Final Iteration

Team 4

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

First Detector

- The ecrecover detector was built to detect improper use of ecrecover() in smart contracts. (Github Issue #1950) [1][5][4] and enhanced based on the signature validity check in open zeppelin implementation. [8]

Second Detector

- **API Key Detector** was Built to detect hard coded API Keys [2][3] based on sonar secrets. [7]
- Enhanced with additional capability for API Key Pattern detection for popular API Keys as described in “Detecting and Mitigating Secret-Key Leaks in Detecting and Mitigating Secret-Key Leaks in Source Code Repositories” [9]

Third Detector

- Detector to detect hard coded passwords[2][3] which can be leaked due to nature of smart contracts.
- **Password detector** uses variable name regular expressions similar to the API Key variable name detector and also based on passwords check implemented in sonar secrets [7].

Hardcoded password example on Github

```
1   pragma solidity ^0.4.8;
2
3   contract password {
4       address public owner;
5       bytes32 key = keccak256("Coolio Englasias");
6
7
8   // This modifier will be included in the function. The modifier essentially tells the function below
9   // "Hey. Do your thing, but only if the input is the same as what we called out to be the correct key."
10
11      modifier onlyKey(string _key) {
12          var hashed = keccak256(_key);
13          require(key == hashed);
14          _;
15      }
16
17  // The function will take the input from the user, and check with the modifier, then let the contract execute if you're correct
18
19      function sausage (string _key) onlyKey(_key) {
20          var hashed = keccak256(_key);
21          key = hashed;
22      }
23  }
```

[https://github.com/MikeD123/Password/blob/master/Password.sol]

Enhancement to API Key Detector

- Previous implementation only checked the variable name to detect hard coded API Keys which was based on sonar secrets API Key Check. [7]
- Added API Key pattern detector based on the patterns for AWS Client ID and Secret Key to detect API keys based on actual variable value.

Enhancement to API Key Detector

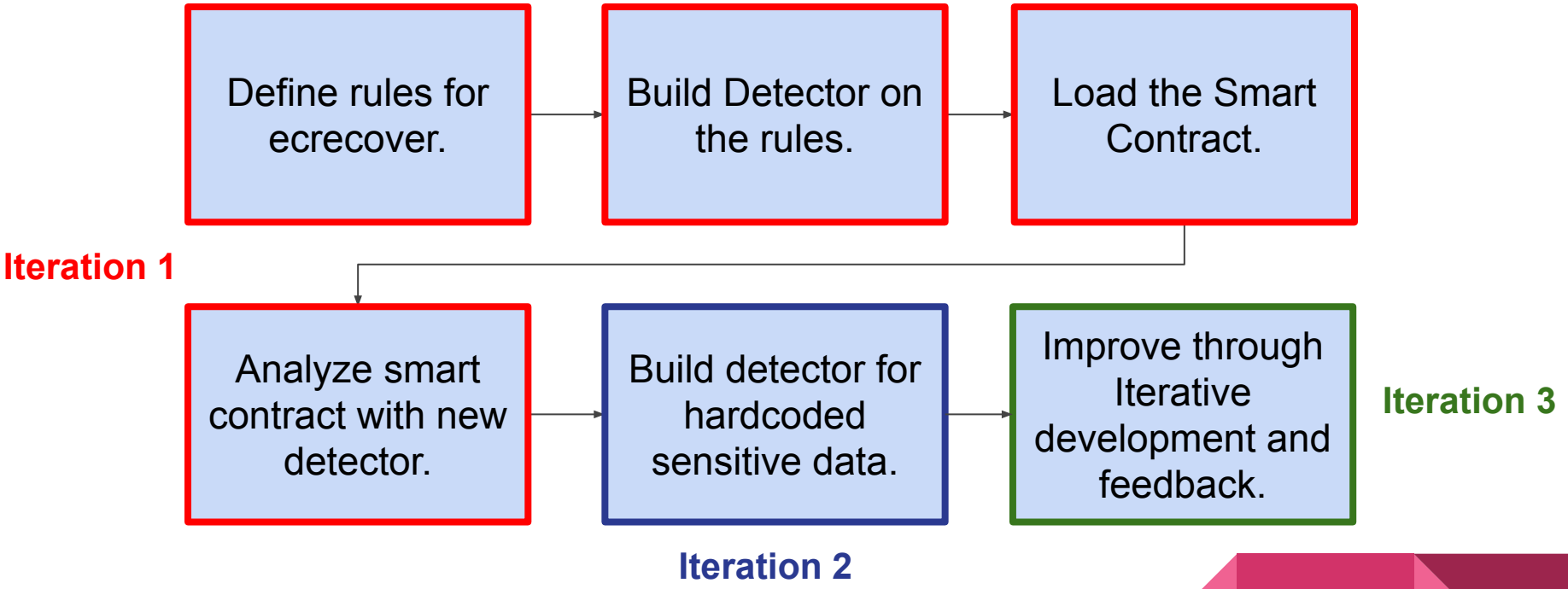
```
def _detect_apikey(function: Function,) -> List[Tuple[Function, DefaultDict[LocalVariable, List[Node]]]]:
    results = []

    for node in function.nodes:
        for ir in node.irs:
            expression = str(ir.expression)
            # Regular expressions used:
            # 1. Sonar qube
            # 2. AWS Client ID format
            # 3. AWS Secret key format
            if re.compile('(api|gitlab|github|slack|google|aws|jenkins)_?(key|token|secret|auth)?'
                    ).search(expression) or re.compile(
                    'AKIA[0-9A-Z]{16}').search(expression) or re.compile(
                    '[0-9a-zA-Z/+]{40}').search(expression):
                results.append(node)

    return (results)
```

Demo

Planning


Define rules for ecrecover.
Build Detector on the rules.
Load the Smart Contract.
Iteration 1
Analyze smart contract with new detector.
Build detector for hardcoded sensitive data.
Improve through Iterative development and feedback.
Iteration 3
Iteration 2

Risks & Mitigations

Risk	Mitigation Plan
Incomplete rule definition	Iteratively build and test the detectors with rules defined
Changing Solidity Standards	Fix Solidity version 0.8.18 as the target version
Missing Slither User's needs from the detector's	Thoroughly research the issue and interact with slither users
False Positive	Test with multiple negative scenarios
False Negatives	Test with multiple positive scenarios

Test cases

Test Case	Expected Output
Implementing API key detector for API key values from popular API's [9].	Analyzing slither deter with AWS [10] Client ID and secrets.
Implementing a hardcoded password detector with password regular expressions based on sonar secrets [9].	Analyzing the slither detector with hardcoded password check.

Customers and Users

User	Role	Feedback
Mehul Hivlekar	New User to Smart Contracts and Solidity	All features implemented in the project were useful and interesting.
Shruthaja Patali Rao	CSE-6324-001 Team 5	All 3 detectors will help increase security in smart contracts.
Devyani Singh	CSE-6324-001 Team 8	Enhancement to API key detectors will make it more effective but more popular API key patterns can be added.

Github Repository

https://github.com/arjunsuvarna1/CSE6324_Team4_Fall23

- Changes made for ecrecover detector have been opened as a pull request to the original slither repository (Pending maintainer review as of 11/27) (Github pull request #2249). [11]

References

[1] Improper usage of ecrecover - https://github.com/crytic/slither/issues/1950
[2] SWC-136 - https://swcregistry.io/docs/SWC-136/
[3] CWE-798 - https://cwe.mitre.org/data/definitions/798.html
[4] SWC-121- https://swcregistry.io/docs/SWC-121/
[5] Detector for ecrecover return value validation and use of nonces - https://github.com/crytic/slither/pull/2015
[6] Ethereum: A Secure Decentralized Generalized Transaction Ledger - https://ethereum.github.io/yellowpaper/paper.pdf
[7] Sonar Secrets Github: https://github.com/Skyscanner/sonar-secrets/tree/master
[8] Openzeppelin ECDSA solidity implementation - https://github.com/OpenZeppelin/openzeppelin-contracts
[9] Detecting and Mitigating Secret-Key Leaks in Source Code Repositories - https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7180102
[10] AWS Access Key ID formats - https://awsteele.com/blog/2020/09/26/aws-access-key-format.html
[11] Improper usage of ecrecover() detector with improvements : https://github.com/crytic/slither/pull/2249