Sri Lanka Institute of Information Technology

SMART CONTRACT REPORT

IE2062 - Web Security

Peiris H.R.T
IT21163340
Y2 S2 CS Weekend Group

Table of Contents

01.Smart Contract Security

What is smart contract Security?

The security of smart contracts is an essential component of blockchain technology, and its primary objective is to guarantee the reliability, authenticity, and completeness of smart contracts. The negotiation and execution of contractual obligations may both be automated with the use of smart contracts, which are self-executing digital contracts stored in the cloud. They are an essential component of the blockchain technology that makes it possible to develop decentralised apps (also known as DApps) that are trustworthy, open, and highly effective.

Yet, because to the inherent complexity of smart contracts and the possibility of human mistake during their formation, these types of agreements may be susceptible to a variety of security flaws. Vulnerabilities in smart contracts may result in severe repercussions, including the misappropriation of cash, the theft of assets, and the suspension of corporate activities.

It is imperative that while building smart contracts, safe coding techniques be adhered to, as this will help to reduce the risks that are connected with vulnerabilities in smart contracts. Input validation, secure coding standards, automated testing, formal verification, and continuous monitoring are some examples of the activities that fall under this category. By adhering to these best practises, developers may help guarantee that their smart contracts are safe and dependable, hence lowering the probability that an attack will be successful and limiting the severity of the effects of any security events that do take place.

Decentralized finance (DeFi) apps are based on blockchain technology and depend on smart contracts to automate financial transactions. The security of smart contracts is especially crucial in this context because of the importance of DeFi applications. As a result of its decentralised design, DeFi is especially susceptible to security flaws since there is no centralised authority to monitor and control the transactions that take place inside it. As a result, securing the safety and validity of smart contracts is essential to the long-term survival of both distributed ledger technology and distributed finance in general.

In general, the security of smart contracts is an essential component of blockchain technology that demands careful attention and consideration from developers, security professionals, and other players in the blockchain ecosystem. Consideration and attention must be paid to this issue.

02.**Blockchain Platforms**

Blockchain is a distributed ledger system that enables data to be stored in a decentralised fashion across a network of computers. This functionality is made possible by blockchain's use of blockchain technology. To put it another way, it is a database that is not controlled by a single entity but rather is distributed among a group of users who collaborate together to update and maintain it.

The method of storing data on a network is referred to as a "blockchain," and its name comes from the phrase. The word "blockchain" comes from the fact that individual transactions are organised into "blocks," which are subsequently connected to one another in the form of a chronological chain. Since each block includes a cryptographic hash of the preceding block, the chain as a whole can never be altered, and the integrity of each block can never be compromised.
Blockchain platforms are:

1. Ethereum:

    Ethereum is a public blockchain platform that has been around since 2013, making it one of the oldest and most well-known. It offers a blockchain that is completely independent of any central authority and is analogous to the Bitcoin blockchain network. According to Manders, its primary advantage is that it paves the way for genuine decentralisation by providing support for smart contracts. In comparison to other platforms, it has much longer processing times and more expensive transaction processing fees. These are its primary shortcomings. In addition to serving as a blockchain platform on which corporate apps are built, it also has its own own cryptocurrency, which is known as ether.

2. IBM Blockchain:

   According to Manders, the most successful users of IBM Blockchain have been business customers that are less risk adverse. IBM Blockchain is a private blockchain network that operates without a central authority. He believes that the most significant prospects lie in finding ways to use it to integrate into business clouds and legacy systems in a more seamless manner than is achievable with other decentralised networks.

3. Bitcoin:

   Bitcoin is the first platform built on blockchain technology, and it was designed to eliminate the need for third-party middlemen in financial transactions. It is the most well-known blockchain platform, and the majority of its applications involve online financial transactions.

4. Hyperledger Fabric:

   A blockchain technology designed for use in a business setting, Hyperledger Fabric is designed to facilitate the creation of permissioned blockchains. It allows the building of private blockchain networks that are safe and scalable, as well as being intended specifically for corporate use cases.

5. Stellar:

   Stellar is a blockchain platform that was developed specifically for the purpose of facilitating the creation of financial applications. It is a decentralised platform that allows for the creation of financial apps that are both safe and inexpensive.

03.**Blockchain Programming Languages**

Blockchain programming languages are programming languages used to develop decentralized applications and smart contracts on blockchain platforms. These languages are used to write code that runs on the blockchain network, allowing for the creation of decentralized applications and the execution of smart contracts.
Blockchain programming was popularized by an unidentified group or a person Satoshi Nakamoto in 2008. It is managed by person-to-person networking as a publicly distributed ledger, where nodes follow a specific set of protocols. According to fortunly, the demand for Blockchain rose in recent years and its total contribution is about to reach $20 billion in near future.
Blockchain Programming Languages are:

1. Solidity:
   Solidity is the Blockchain Programming Language that is recommended by developers all around the globe due to its stability and widespread usage. You may easily acquire the abilities necessary to adhere to this programming language, regardless of whether you are just starting out in the field or are a seasoned expert.
2. Java:
   Java is one of the most widely used programming languages, and it is also used to frame some of the most notable blockchain technologies, including Ethereum, Hyperledger Fabric, IOTA, NEO, and others.
   This programming language was developed in 1995 by James Gosling, and it offers a strong Application Programming Interface (API), which includes class-based object-oriented programming. In Blockchain programming, the three properties of Java that are put to use the most often are the API, OOP, and portability.

3. Python:

   Python is the most user-friendly and has the shortest codes of all of these programming languages, therefore if you are just starting out in blockchain development, it may be advisable to use Python as your language of choice. Due to the fact that Python is an open-source language, you have access to a variety of resources and plug-ins. It offers dynamic support for OOP and is widely used in Blockchain Development, Machine Learning, and Artificial Intelligence research and development.

4. C++:

   In 1985, Bjarne Stroustrup was the one who came up with the idea for what is now one of the top 10 programming languages: C++. C++ is recommended by Blockchain Programmer developers because it offers a good deal of run-time polymorphism, function overloading, and multi-threading capabilities. It gives developers the ability to shape the data in accordance with their requirements. It plays a significant role in the development of many Blockchain programming languages, including Stellar, Ripple, Bitcoin, and others.

5. Go:

   A compiled and statically typed computer language, Go (sometimes spelt Golang) is known for its use in the Go programming language. Because of its notable characteristics, Go is the programming language of choice for Blockchain development among a significant number of developers across the globe. Golang's user-friendliness, simplicity, speed, and lack of rigidity make it an ideal programming language for beginners as well as experienced programmers.

04.Smart Contract vulnerabilities

Smart contract vulnerabilities are any faults or defects in the design or implementation of a smart contract that may be exploited by attackers. Attackers can take advantage of these vulnerabilities in order to gain control of a smart contract. Contracts that are referred to as smart contracts are contracts that are programmed to execute themselves automatically when specific circumstances are satisfied. In blockchain technology, they are a common component used to allow transactions to take place in a safe and transparent manner without the need for middlemen.

However, just like any other software program, smart contracts might include flaws that other parties can use for their own malicious purposes. Reentrancy attacks, integer overflow and underflow, time manipulation, access control concerns, malicious libraries, front-running, and logic flaws are a few examples of the frequent vulnerabilities that might exist in smart contracts.

Some common smart contract vulnerabilities are:

1. Reentrancy Attack:

   This is one of the most typical examples of a vulnerability that may occur in smart contracts, and it's a significant one. An attacker will carry out this kind of attack by repeatedly calling a function included inside a contract before the prior function call has reached its conclusion. This may provide the attacker the ability to seize control of the contract or at least access to its money and let them to withdraw them.

2. Integer Overflow:

   An integer overflow or underflow happens whenever an arithmetic operation performed on integers results in a number that is either too big or too tiny to be represented respectively. This might result in the smart contract behaving in an unanticipated manner, which opens the door for adversaries to steal cash or manipulate the contract.

3. Time Manipulation:

   A significant number of smart contracts depend on timestamps to carry out various actions. It is possible for an adversary to take advantage of the contract if they are in a position to change the timestamp.

4. Access Control:

   Access control techniques are often used in smart contracts to guarantee that only authorized parties are able to see, access, or make changes to the contract. Attackers will have the ability to acquire unauthorized access to the contract if these procedures are badly designed or executed.

5. Malicious Libraries:

   In order to carry out certain tasks, smart contracts sometimes depend on third-party libraries. Attackers may use these libraries as a means of attacking the smart contract if the libraries themselves are malicious or have flaws.

6. Front-running:

   The term "front-running" refers to a kind of attack in which an adversary monitors a transaction on a blockchain and then submits their own transaction with greater fees so that it will be processed ahead of the original transaction. This might provide the attacker the ability to alter the contract and, as a result, perhaps steal cash.

7. Logic Errors:

   The logic faults that are included in the coding of smart contracts may sometimes be exploited by malicious actors. These flaws may lead to the contract behaving in a way that was not intended, which gives attackers an opportunity to exploit the contract.

05.Smart Contract tools

Developers and users are able to build, implement, test, and interact with smart contracts with the use of tools known as smart contract tools. Smart contract tools may take the shape of software programs or platforms. Contracts that execute themselves are known as smart contracts, and they include the conditions of the agreement between the buyer and the seller being directly encoded into lines of code, which are then recorded on a blockchain and automatically carried out.
Programming languages, development frameworks, Integrated Development Environments (IDEs), personal blockchains, testing and debugging tools are all examples of the kind of features that may be found in smart contract tools. These tools are used to enable the creation and implementation of smart contracts.

Some common smart contract tools are:

1. Ethereum:

   Ethereum is a platform for blockchain technology that gives developers the ability to construct and deploy decentralized applications, such as smart contracts. It is compatible with the Solidity programming language, which is one of the most popular options for composing smart contracts

2. Remix IDE:

   On the blockchain, smart contracts may be written, tested, and deployed with the help of Remix, which is an Integrated Development Environment (IDE) that runs in the web browser. It boasts an intuitive graphical user interface and offers a wide range of capabilities, including code highlighting, auto-completion, and other tools for debugging.

3. Truffle Suite:

    The development framework known as Truffle offers users the ability to construct, test, and deploy smart contracts. It is compatible with a variety of programming languages, one of which being Solidity, and it offers functions such as the compilation, deployment, and testing of contracts.

4. Ganache:

    Developers have the ability to test and deploy smart contracts in a local environment while using Ganache, which is a personal blockchain for development that was created by Ganache. It offers functionality such as contract deployment, transaction simulation, and debugging tools, among other things.

5. Metamask:

    Users are given the ability to engage with decentralized apps and smart contracts that are stored on the blockchain with the use of a browser plugin called Metamask. It offers a straightforward user experience for making and receiving cryptocurrency transactions as well as working with smart contracts.

6. OpenZeppelin:

    OpenZeppelin is a free and open-source collection of reusable smart contract components that may be used into audited and protected smart contracts by software developers. It offers a number of different components, including access control, payment splitting, and token issuance, among others.

7. DappHub:

    DappHub is a set of tools for the creation of smart contracts that includes frameworks for the construction of decentralized apps and other types of smart contracts. In addition to that, it offers tools for the implementation, testing, and auditing of contracts.

06.Smart Contract CTF

Caviar Contest

A new kind of smart contract, to be called the Caviar smart contract, is now under development on the Ethereum distributed ledger. The term "smart contract" will eventually be used to describe this emerging legal concept. Caviar will be the name given to this novel kind of smart contract in recognition of the coin. Caviar will henceforth be used to refer to this new kind of smart contract in honour of the cryptocurrency. As a direct consequence, this change will streamline the procedures required to establish a decentralised investment fund and administer its holdings. The additional benefit of this simplification will be increased efficiency. Additionally, this streamlining will occur as an indirect result of the forthcoming growth. It is feasible to set up Caviar smart contracts at this area if the user is currently located here. To reduce the likelihood of a negative return and maximise the potential for a positive one, the Caviar platform offers investors the ability to participate in a diversified portfolio that incorporates cryptocurrency and real estate. This is achieved by giving people more opportunities to put their money to work.

Smart contracts are computer programmes that may be used to do a variety of tasks automatically, such as receiving and dispersing investment funds or purchasing and selling assets in a portfolio based on predetermined rules and parameters. Investment management include buying and selling assets in a portfolio, as well as receiving and allocating investment funds. Among these activities is the accumulation and distribution of capital for investment. The term "activities" as used here refers to the acquisition, allocation, and sale of investment capital and portfolio assets. Among these tasks is the collection of investment money that will be used to make new purchases for the portfolio and the sale of existing portfolio assets. Furthermore, we will be investing the money we earn from these endeavours. The goal of these actions is to increase the portfolio's holdings, which may be done by selling assets already held in the portfolio or by raising investment capital. In addition, these measures are taken to boost the portfolio's current asset worth. Because every transaction is recorded and the transaction history can be examined at any time, investing is now a more transparent and secure process. Therefore, investment is now a more transparent and secure process thanks to blockchain technology. The result of this shift is an investment process that is more open and safe for investors. As a consequence, the investment process is now not only more secure, but also more public and straightforward. In addition, it improves the process's overall transparency.

GitHub Link: https://github.com/code-423n4/2023-04-caviar

In Scope

- **caviar/Pair.sol**
  - src/EthRouter.sol
- **openzeppelin/interfaces/IERC2981.sol**
  - src/EthRouter.sol
  - src/PrivatePool.sol
- **openzeppelin/interfaces/IERC3156FlashLender.sol**
  - src/PrivatePool.sol
- **openzeppelin/utils/Base64.sol**
  - src/PrivatePoolMetadata.sol
- **openzeppelin/utils/Strings.sol**
  - src/PrivatePoolMetadata.sol
- **royalty-registry-solidity/IRoyaltyRegistry.sol**
  - src/EthRouter.sol
  - src/PrivatePool.sol
- **solady/utils/LibClone.sol**
  - src/Factory.sol
- **solady/utils/MerkleProofLib.sol**
  - src/PrivatePool.sol
- **solmate/auth/Owned.sol**
  - src/Factory.sol
- **solmate/tokens/ERC20.sol**
  - src/Factory.sol
  - src/PrivatePool.sol
  - src/PrivatePoolMetadata.sol
- **solmate/tokens/ERC721.sol**
  - src/EthRouter.sol
  - src/Factory.sol
  - src/PrivatePool.sol
  - src/PrivatePoolMetadata.sol
- **solmate/utils/FixedPointMathLib.sol**
  - src/PrivatePool.sol
- **solmate/utils/SafeTransferLib.sol**
  - src/EthRouter.sol
  - src/Factory.sol
  - src/PrivatePool.sol

Scoping Details

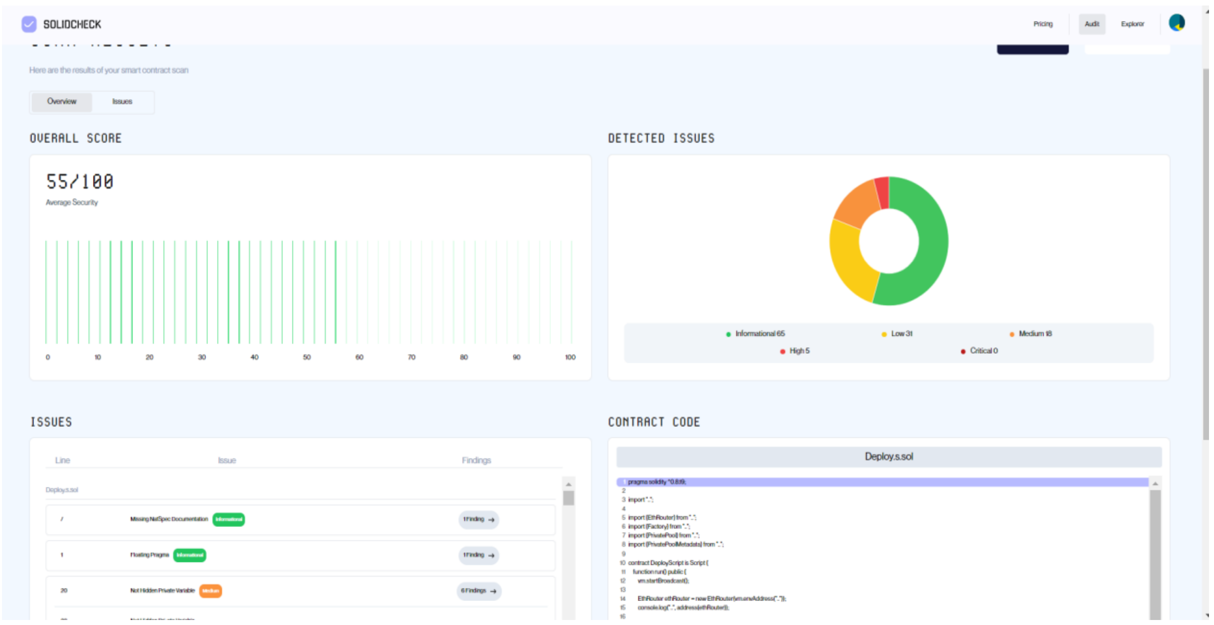
- If you have a public code repo, please share it here:
- How many contracts are in scope?:   4
- Total SLoC for these contracts?: 725
- How many external imports are there?: 12
- How many separate interfaces and struct definitions are there for the contracts within scope?: 3
- Does most of your code generally use composition or inheritance?:   inheritance
- How many external calls?:   10
- What is the overall line coverage percentage provided by your tests?: N/a
- Is there a need to understand a separate part of the codebase / get context in order to audit this part of the protocol?:   Caviar public pools: https://github.com/outdoteth/caviar
- Please describe required context: The EthRouter contract routes trades to caviar public pools (in addition to private pools)
- Does it use an oracle?: no
- Does the token conform to the ERC20 standard?: N/a
- Are there any novel or unique curve logic or mathematical models?: no
- Does it use a timelock function?: no
- Is it an NFT?: Yes
- Does it have an AMM?:   Yes
- Is it a fork of a popular project?:   No
- Does it use rollups?:   No
- Is it multi-chain?: No
- Does it use a side-chain?: No

<u>Online tools</u>

Solid Check

Solidcheck.io is a platform for cybersecurity that offers a number of services to help businesses and organizations in assessing the areas in which their online presence puts them at risk and the areas in which they have the ability to take actions to mitigate those risks. These services can be found on the Solidcheck.io website. The system provides automated security scans and evaluations, which are able to identify potential threats such as malware, phishing, and other types of attacks.

https://solidcheck.io/



Here are the results of your smart contract scan

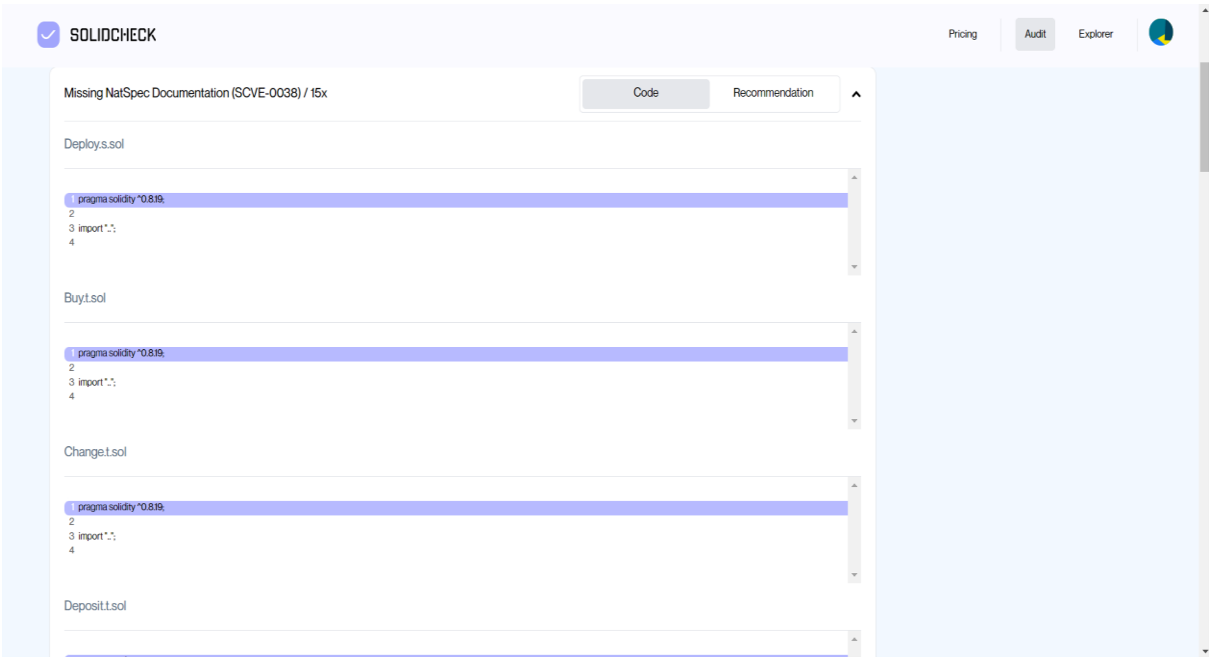
55/100
Average Security

● Informational 65 ● Low 31 ● Medium 18 ● High 5 ● Critical 0

Line	Issue	Findings
Deploy.s.sol		
7	Missing NatSpec Documentation Informational	1 Finding →
1	Floating Pragma Informational	1 Finding →
20	Not Hidden Private Variable Medium	6 Findings →

Deploy.s.sol

```
pragma solidity ^0.8.19;

import "..";

import {EthRouter} from "..";
import {Factory} from "..";
import {PrivatePool} from "..";
import {PrivatePoolMetadata} from "..";

contract DeployScript is Script {
  function run() public {
    vm.startBroadcast();

    EthRouter ethRouter = new EthRouter(vm.envAddress(".."));
    console.log("..", address(ethRouter));

```



Missing NatSpec Documentation (SCVE-0038) / 15x

Deploy.s.sol

```
pragma solidity ^0.8.19;

import "..";

```

Buy.t.sol

```
pragma solidity ^0.8.19;

import "..";

```

Change.t.sol

```
pragma solidity ^0.8.19;

import "..";

```

Deposit.t.sol

Sell.t.sol

```
pragma solidity ^0.8.19;

import "..";

```

Constructor.t.sol

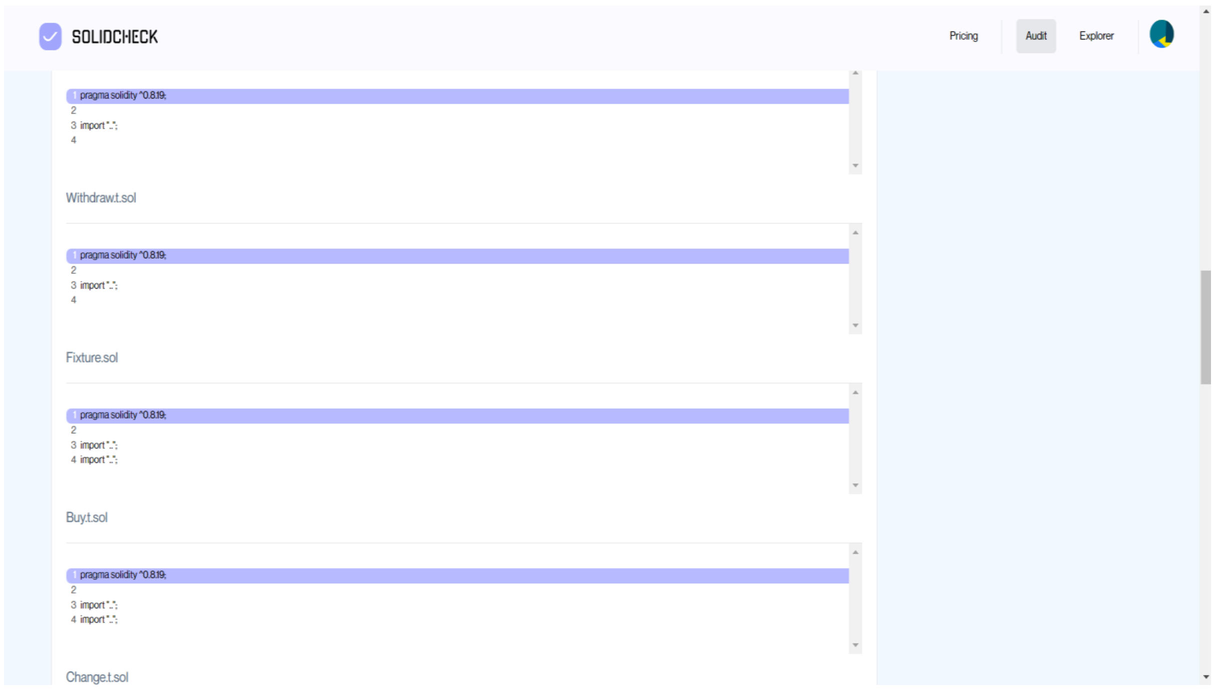
```
pragma solidity ^0.8.19;

import "..";

```

Create.t.sol

```
pragma solidity ^0.8.19;

import "..";

```

Nft.t.sol

```
pragma solidity ^0.8.19;

```



```
pragma solidity ^0.8.19;

import "..";

```

Withdraw.t.sol

```
pragma solidity ^0.8.19;

import "..";

```

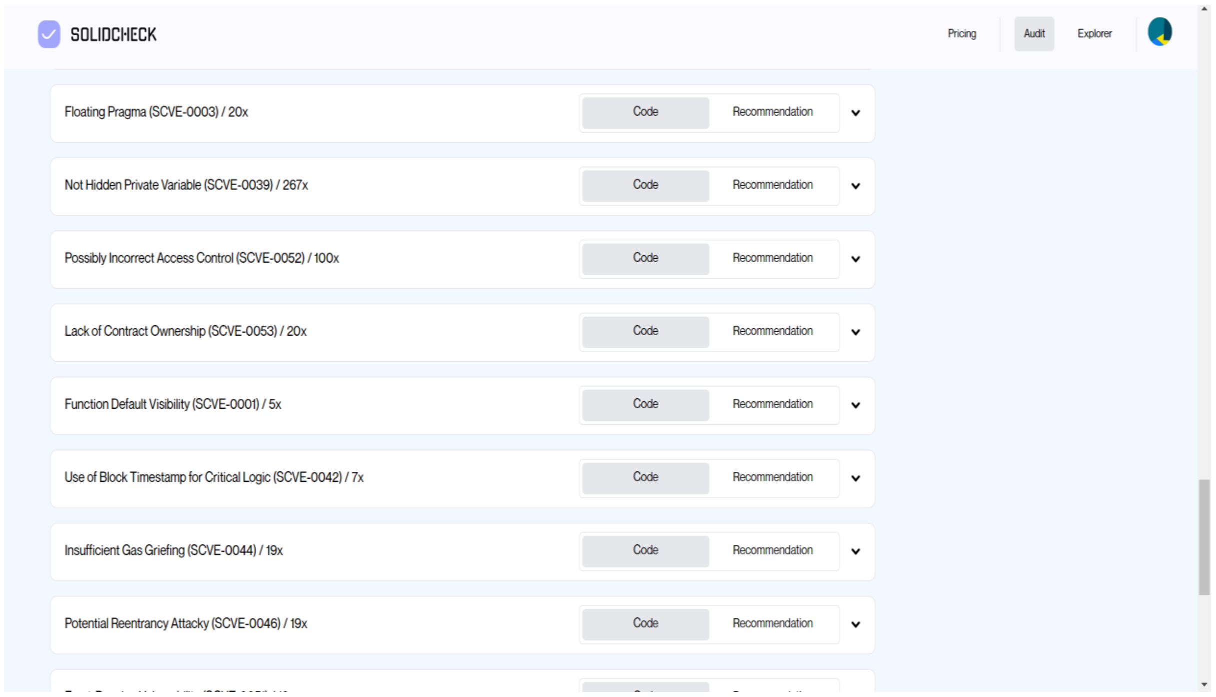
Fixture.sol

```
pragma solidity ^0.8.19;

import "..";
import "..";
```

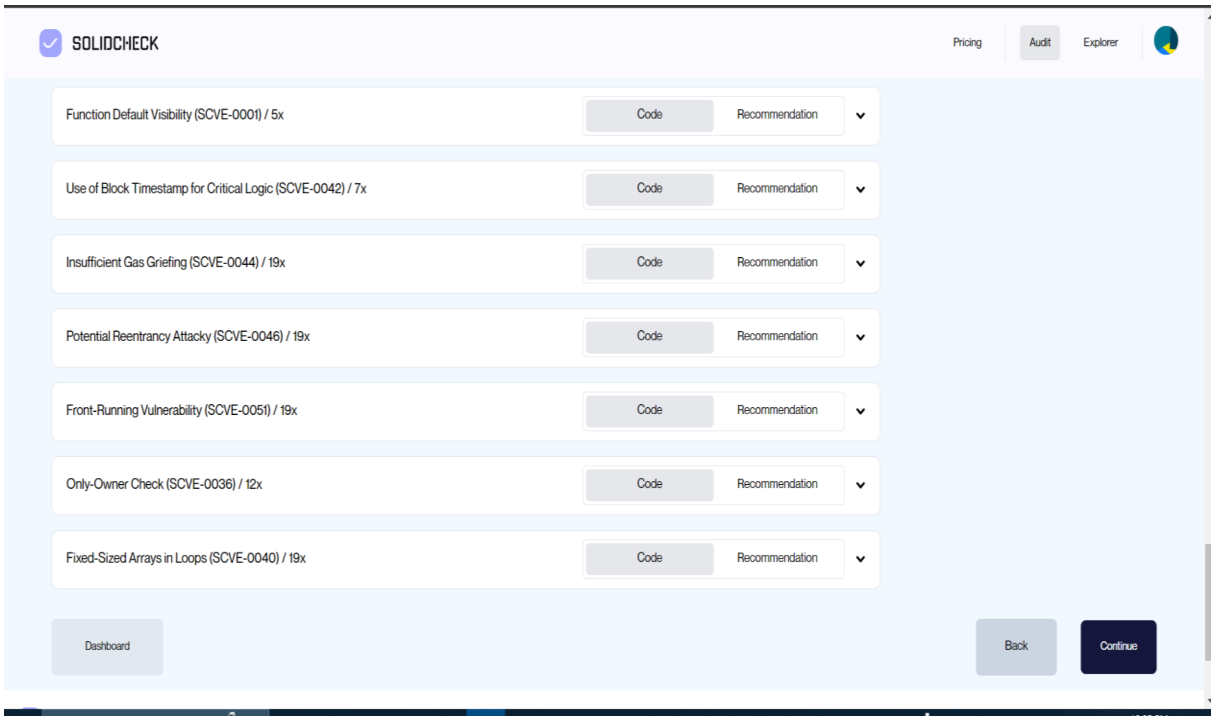
Buy.t.sol

```
pragma solidity ^0.8.19;

import "..";
import "..";
```

Change.t.sol


SOLIDCHECK

Floating Pragma (SCVE-0003) / 20x	Code	Recommendation
Not Hidden Private Variable (SCVE-0039) / 267x	Code	Recommendation
Possibly Incorrect Access Control (SCVE-0052) / 100x	Code	Recommendation
Lack of Contract Ownership (SCVE-0053) / 20x	Code	Recommendation
Function Default Visibility (SCVE-0001) / 5x	Code	Recommendation
Use of Block Timestamp for Critical Logic (SCVE-0042) / 7x	Code	Recommendation
Insufficient Gas Griefing (SCVE-0044) / 19x	Code	Recommendation
Potential Reentrancy Attacky (SCVE-0046) / 19x	Code	Recommendation

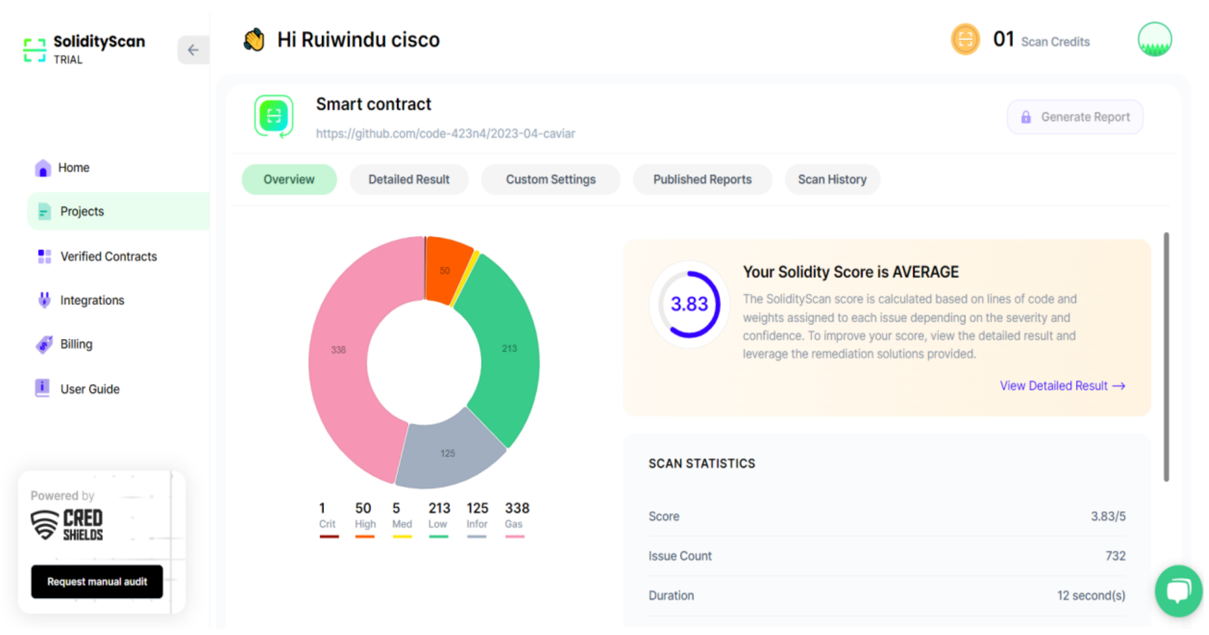

SOLIDCHECK

Function Default Visibility (SCVE-0001) / 5x	Code	Recommendation
Use of Block Timestamp for Critical Logic (SCVE-0042) / 7x	Code	Recommendation
Insufficient Gas Griefing (SCVE-0044) / 19x	Code	Recommendation
Potential Reentrancy Attacky (SCVE-0046) / 19x	Code	Recommendation
Front-Running Vulnerability (SCVE-0051) / 19x	Code	Recommendation
Only-Owner Check (SCVE-0036) / 12x	Code	Recommendation
Fixed-Sized Arrays in Loops (SCVE-0040) / 19x	Code	Recommendation

Solidity Scan

SolidityScan.com is a website that offers a broad variety of tools and services for auditing and evaluating smart contracts that were written in the Solidity programming language and designed for the Ethereum blockchain. These smart contracts were created by users. It is possible to conduct tests and inspections on these smart contracts in order to guarantee that they are operating properly.

https://solidityscan.com/



Hi Ruiwindu cisco

Smart contract

https://github.com/code-423n4/2023-04-caviar

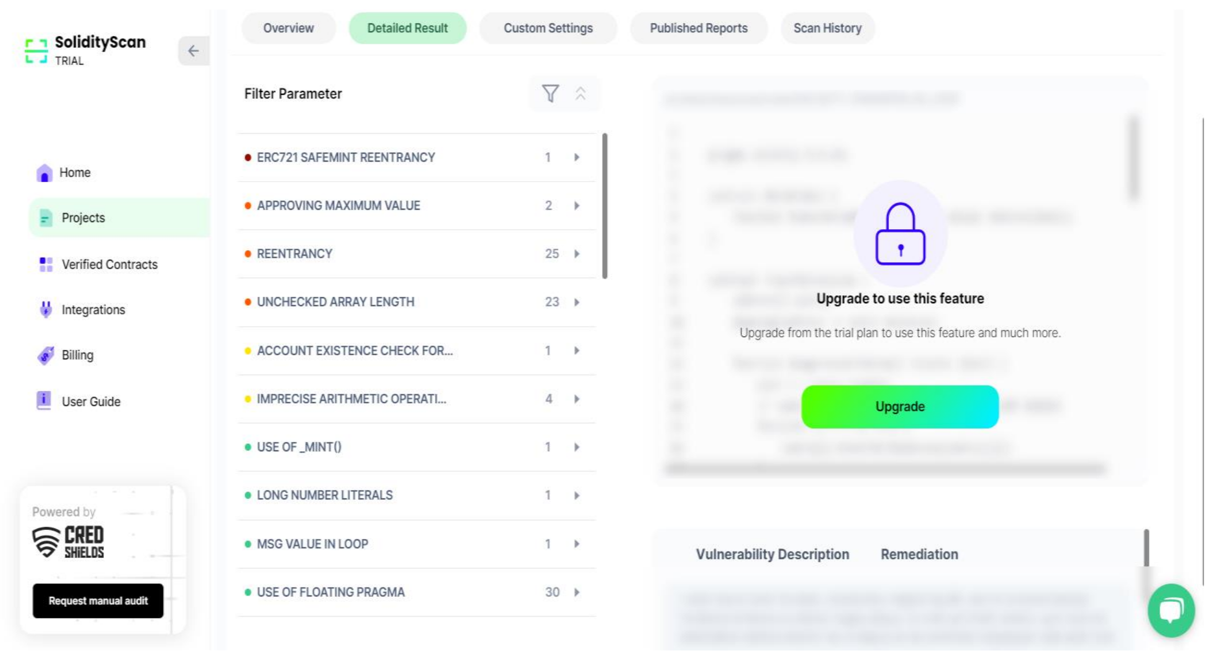
1	50	5	213	125	338
Crit	High	Med	Low	Infor	Gas

Your Solidity Score is AVERAGE

The SolidityScan score is calculated based on lines of code and weights assigned to each issue depending on the severity and confidence. To improve your score, view the detailed result and leverage the remediation solutions provided.

SCAN STATISTICS

Score	3.83/5
Issue Count	732
Duration	12 second(s)

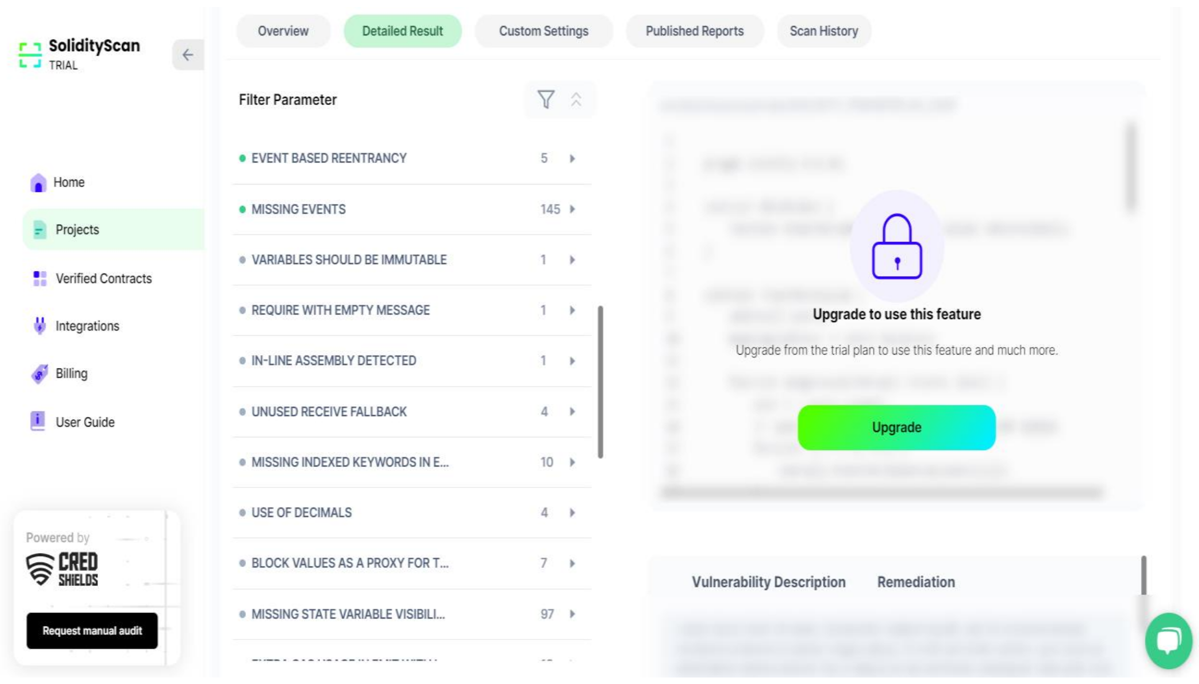


Filter Parameter

● ERC721 SAFEMINT REENTRANCY	1
● APPROVING MAXIMUM VALUE	2
● REENTRANCY	25
● UNCHECKED ARRAY LENGTH	23
● ACCOUNT EXISTENCE CHECK FOR...	1
● IMPRECISE ARITHMETIC OPERATI...	4
● USE OF _MINT()	1
● LONG NUMBER LITERALS	1
● MSG VALUE IN LOOP	1
● USE OF FLOATING PRAGMA	30

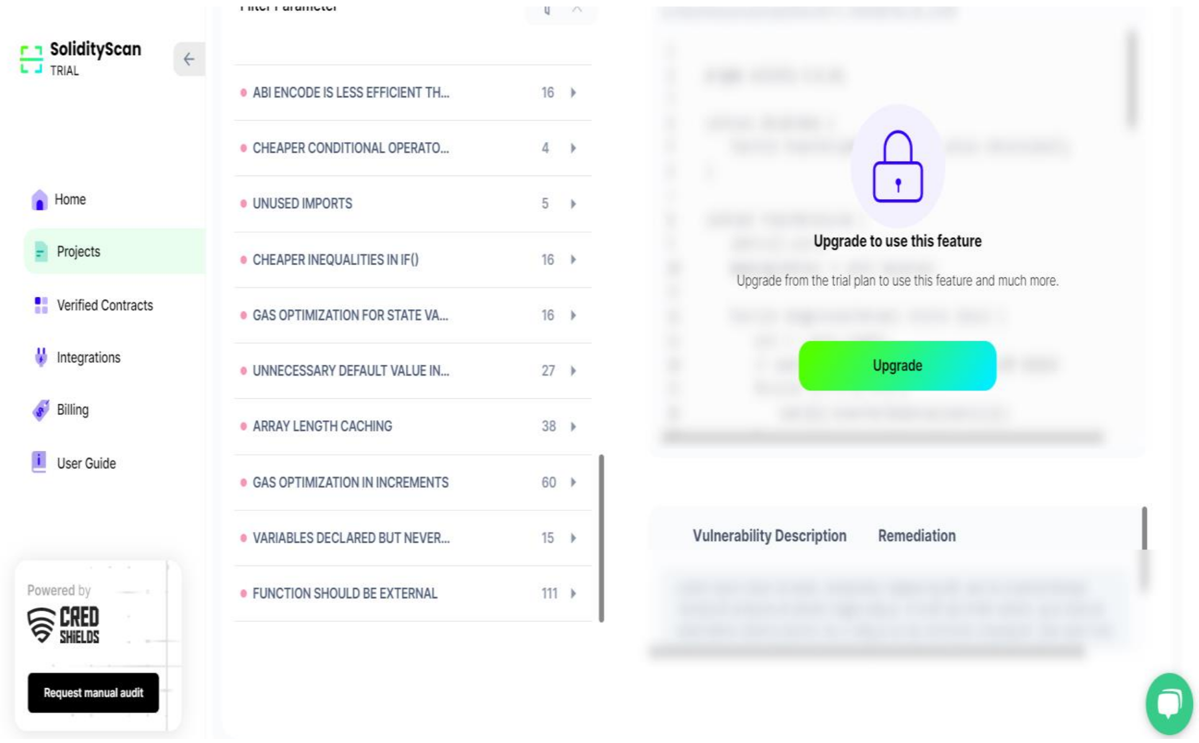
Upgrade to use this feature

Upgrade from the trial plan to use this feature and much more.



Filter Parameter

EVENT BASED REENTRANCY	5
MISSING EVENTS	145
VARIABLES SHOULD BE IMMUTABLE	1
REQUIRE WITH EMPTY MESSAGE	1
IN-LINE ASSEMBLY DETECTED	1
UNUSED RECEIVE FALLBACK	4
MISSING INDEXED KEYWORDS IN E...	10
USE OF DECIMALS	4
BLOCK VALUES AS A PROXY FOR T...	7
MISSING STATE VARIABLE VISIBILI...	97



ABI ENCODE IS LESS EFFICIENT TH...	16
CHEAPER CONDITIONAL OPERATO...	4
UNUSED IMPORTS	5
CHEAPER INEQUALITIES IN IF()	16
GAS OPTIMIZATION FOR STATE VA...	16
UNNECESSARY DEFAULT VALUE IN...	27
ARRAY LENGTH CACHING	38
GAS OPTIMIZATION IN INCREMENTS	60
VARIABLES DECLARED BUT NEVER...	15
FUNCTION SHOULD BE EXTERNAL	111

Conclusion of the Smart Contract

Code4rena is a decentralized organization that is made up of people that are experts in the field of smart contracts. Researchers in cybersecurity, auditors, programmers, and a range of other specialists make up this group of people. Members of the community, who are known as Wardens, compete in an event called a Code4rena audit contest in order to examine, audit, or study the logic of smart contracts in return for a payment that is provided by the projects that are sponsoring the contest. The projects that are supporting the competition will provide rewards for the Wardens in recognition of their accomplishments. During the audit contest that is described in this article, Code4rena conducted an analysis of the Caviar smart contract system. Solidity was used as the underlying language for the Caviar smart contract system. The deadline for entry for the auditing competition was the 19th of December 2022, and it was open beginning on the 12th of the same month.

The investigation of the system that was carried out by Code4rena uncovered a total of eight separate flaws or vulnerabilities in the software. These vulnerabilities were ranked as having a risk that was classed as belonging to the category of HIGH severity, while these vulnerabilities were ranked as having a risk that was classified as belonging to the category of MEDIUM severity for a total of five of them. In addition, the investigation of Code4rena yielded 25 reports that detailed problems with a risk assessment of LOW intensity or were deemed to be insignificant. These reports provided information about the problems. It was determined that these concerns did not pose an urgent danger. In addition, there were 31 publications that presented a range of suggestions for the improvement of gas. You can trace the very first observation of each and every one of the topics that are discussed in this essay back to the point when it all started.

References

[1] "Smart contracts defined," [Online]. Available: https://www.ibm.com/topics/smart-contracts#:~:text=Smart%20contracts%20are%20simply%20programs,intermediary's%20involvement%20or%20time%20loss..

[2] "smart-contract-tools," [Online]. Available: https://www.alchemy.com/best/smart-contract-tools.

[3] [Online]. Available: https://www.elluminatiinc.com/smart-contract-development-tools/.

[4] [Online]. Available: https://startupstash.com/smart-contract-tools/.

[5] [Online]. Available: https://github.com/code-423n4/2023-04-caviar.