QuillAudits

Audit Report
June, 2021

Asva Labs

Contents

Scope of Audit

The scope of this audit was to analyze and document ASVA smart contract codebase for quality, security, and correctness.

Check Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

	High	Medium	Low	Informational
Open	0	1	1	4
Closed	0	0	0	0

Introduction

During the period of **MAY 25th, 2021 to May 27th, 2021** - QuillAudits Team performed a security audit for **ASVA** smart contracts.

The code for the audit was taken from following the official link: https://bscscan.com/address/0x83d787203610b0737fec1cbb1aa581014009214d#contracts

Issues Found – Code Review / Manual Testing

High severity issues

No Issues found.

Medium severity issues

1. Transfer() function is utilized

Line	Function Names
974-976	receive()
978-980	fallabck()

Description:
.transfer() and **.send()** forward exactly 2,300 gas to the recipient. The goal of this hardcoded gas stipend was to prevent reentrancy vulnerabilities, but this only makes sense under the assumption that gas costs are constant. Recently EIP 1884 was included in the Istanbul hard fork. One of the changes included in EIP 1884 is an increase in the gas cost of the SLOAD operation, causing a contract's fallback function to cost more than 2300 gas.

```
// bad
contract Vulnerable {
    function withdraw(uint256 amount) external {
        // This forwards 2300 gas, which may not be enough if the recipient
        // is a contract and gas costs change.
        msg.sender.transfer(amount);
    }
}

// good
contract Fixed {
    function withdraw(uint256 amount) external {
        // This forwards all available gas. Be sure to check the return value!
        (bool success, ) = msg.sender.call.value(amount)("");
        require(success, "Transfer failed.");
    }
}
```

Recommendation:
It's recommended to stop using .transfer() and .send() and instead use .call().

References:
External Calls

Status: Acknowledged by Auditee

Low level severity issues

2. Business Logic

Line	Function Names
982-984	getBalance
978-980	fallabck()

Description:
Business logic vulnerabilities are flaws in the design and implementation of an application that allows an attacker to elicit unintended behaviour. This potentially enables attackers to manipulate legitimate functionality to achieve a malicious goal. These flaws are generally the result of failing to anticipate unusual application states that may occur and, consequently, failing to handle them safely.

In this case, the attackers cannot exploit the flaw but the function is redundant to be implemented since the **balance** of the contract is always **Zero**.

The balance of the contract always returns **Zero** due to the **transfer()** function implemented in the **receive()** function. When other addresses send any ether to the contract, the **receive()** function will trigger the **owner.transfer()** function. Therefore, the recipient, in this case, is the **owner()** not the contract itself.

```
receive() payable external {
        payable(owner()).transfer(msg.value);
}
```

Similar to the **fallabck()** function, the **owner()** in this case will always get **Zero** fund since the balance of the contract is always Zero.

```
function fallabck() public payable {
        payable(owner()).transfer(getBalance());
}
```

Recommendation:
Remove redundant functions if they congest code but offer no value.

References:
Business Logic Vulnerability

Status: Acknowledged by Auditee

Informational

3. Different pragma directives are used

Version used:

0.7.6
>=0.6.0<0.8.0
>=0.6.2<0.8.0

Description:
Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:
Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case of contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile it locally.

References:
Different pragma versions are used.
Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version

Status: Acknowledged by Auditee

4. Incorrect versions of Solidity

Version used:

0.7.6
>=0.6.0<0.8.0
>=0.6.2<0.8.0

Description:
solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statements.

Remediation:
Deploy with any of the following Solidity versions:

0.5.16 - 0.5.17
0.6.11 - 0.6.12
0.7.5 - 0.7.6

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

References:
Correct versions of solidity.

Status: Acknowledged by Auditee

5. Conformance to Solidity naming conventions

Line	Function Names

925-950	uint256 private constant initialSupply = 90e6 * 10 ** 18; //90 million
	//$0.1 = 10 cents
	uint256 public constant seedPrice = 10;
	//$0.3 = 20 cents
	uint256 public constant privatePrice = 20;
	//$0.3 = 30 cents
	uint256 public constant publicPrice = 30;
	//7.5%

```
uint256 public constant seedTokenPercentage = 750;
//20%
uint256 public constant privateSalePercentage = 2000;
//12.5%
uint256 public constant publicSalePercentage = 1250;
//15%
uint256 public constant liquidityPercentage = 1500;
//10%
uint256 public constant teamPercentage = 1000;
//5%
uint256 public constant partnersPercentage = 500;
//20%
uint256 public constant stakingPercentage = 2000;
//5%
uint256 public constant marketingPercentage = 500;
//5%
uint256 public constant reserveFundPercentage = 500;
```

Description:
Solidity defines a naming convention that should be followed.

In this case, constants should be named with all capital letters with underscores separating words. Examples: MAX_BLOCKS, TOKEN_NAME, TOKEN_TICKER, CONTRACT_VERSION.

Remediation:
Follow the Solidity naming convention.

Status: Acknowledged by Auditee

6. Incorrect Fallback function name

Line	Function Names
978	function fallabck

Description:
The crucial mistake has been made when the **fallabck** function is aimed to act as a **Fallback** function in solidity.

A misunderstanding function name could influence code readability, in some cases, which may lead to bugs in the future. The smart contract does not initialize or incorrectly initializes a resource, which might leave the resource in an unexpected state when it is accessed or used.

Remediation:
A Fallback function should be declared using **fallback () external [payable]** (without the function keyword). This function cannot have arguments, cannot return anything and must have external visibility.

It is executed on a call to the contract if none of the other functions matches the given function signature, or if no data was supplied at all and there is no receive Ether function. The fallback function always receives data, but in order to also receive Ether, it must be marked payable.

The name should clearly, without ambiguity, indicate what the function does. In this case, the function name should be distinguished between fallabck and fallback.

References:
Fallback function

Status: Acknowledged by Auditee

Functional test

Function Names	Testing results
fallabck	Failed
getBalance	Failed

Automated Testing

Slither

```
INFO:Detectors:
ASVA.fallabck() (asva.sol#978-988) sends eth to arbitrary user
        Dangerous calls:
        - address(owner()).transfer(getBalance()) (asva.sol#979)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Address.isContract(address) (asva.sol#684-693) uses assembly
        - INLINE ASM (asva.sol#691)
Address._verifyCallResult(bool,bytes,string) (asva.sol#829-846) uses assembly
        - INLINE ASM (asva.sol#838-841)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.7.6', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
        - >=0.6.0<0.8.0 (asva.sol#5)
        - >=0.6.0<0.8.0 (asva.sol#28)
        - >=0.6.0<0.8.0 (asva.sol#104)
        - >=0.6.0<0.8.0 (asva.sol#317)
        - >=0.6.0<0.8.0 (asva.sol#621)
        - >=0.6.2<0.8.0 (asva.sol#661)
        - >=0.6.0<0.8.0 (asva.sol#849)
        - 0.7.6 (asva.sol#917)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (asva.sol#829-846) is never used and should be removed
Address.functionCall(address,bytes) (asva.sol#737-739) is never used and should be removed
Address.functionCall(address,bytes,string) (asva.sol#747-749) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (asva.sol#762-764) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (asva.sol#772-779) is never used and should be removed
Address.functionDelegateCall(address,bytes) (asva.sol#811-813) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (asva.sol#821-827) is never used and should be removed
Address.functionStaticCall(address,bytes) (asva.sol#787-789) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (asva.sol#797-803) is never used and should be removed
Address.isContract(address) (asva.sol#684-693) is never used and should be removed
Address.sendValue(address,uint256) (asva.sol#711-717) is never used and should be removed
BEP20._setupDecimals(uint8) (asva.sol#600-602) is never used and should be removed
Context._msgData() (asva.sol#22-25) is never used and should be removed
SafeMath.div(uint256,uint256,string) (asva.sol#291-294) is never used and should be removed
SafeMath.mod(uint256,uint256) (asva.sol#253-256) is never used and should be removed
```

```
Address.sendValue(address,uint256) (asva.sol#711-717) is never used and should be removed
BEP20._setupDecimals(uint8) (asva.sol#600-602) is never used and should be removed
Context._msgData() (asva.sol#22-25) is never used and should be removed
SafeMath.div(uint256,uint256,string) (asva.sol#291-294) is never used and should be removed
SafeMath.mod(uint256,uint256) (asva.sol#253-256) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (asva.sol#311-314) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (asva.sol#125-129) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (asva.sol#161-164) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (asva.sol#171-174) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (asva.sol#146-154) is never used and should be removed
SafeMath.trySub(uint256,uint256) (asva.sol#136-139) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (asva.sol#5) is too complex
Pragma version>=0.6.0<0.8.0 (asva.sol#28) is too complex
Pragma version>=0.6.0<0.8.0 (asva.sol#104) is too complex
Pragma version>=0.6.0<0.8.0 (asva.sol#317) is too complex
Pragma version>=0.6.0<0.8.0 (asva.sol#621) is too complex
Pragma version>=0.6.2<0.8.0 (asva.sol#661) is too complex
Pragma version>=0.6.0<0.8.0 (asva.sol#849) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (asva.sol#711-717):
        - (success) = recipient.call{value: amount}() (asva.sol#715)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (asva.sol#772-779):
        - (success,returndata) = target.call{value: value}(data) (asva.sol#777)
Low level call in Address.functionStaticCall(address,bytes,string) (asva.sol#797-803):
        - (success,returndata) = target.staticcall(data) (asva.sol#801)
Low level call in Address.functionDelegateCall(address,bytes,string) (asva.sol#821-827):
        - (success,returndata) = target.delegatecall(data) (asva.sol#825)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Constant ASVA.initialSupply (asva.sol#925) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ASVA.seedPrice (asva.sol#927) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ASVA.privatePrice (asva.sol#929) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ASVA.publicPrice (asva.sol#931) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ASVA.seedTokenPercentage (asva.sol#933) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ASVA.privateSalePercentage (asva.sol#935) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ASVA.publicSalePercentage (asva.sol#937) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ASVA.liquidityPercentage (asva.sol#939) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ASVA.marketingPercentage (asva.sol#947) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ASVA.reserveFundPercentage (asva.sol#949) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ASVA.percentageDivider (asva.sol#959) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (asva.sol#23)" inContext (asva.sol#17-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
name() should be declared external:
        - BEP20.name() (asva.sol#377-379)
symbol() should be declared external:
        - BEP20.symbol() (asva.sol#385-387)
decimals() should be declared external:
        - BEP20.decimals() (asva.sol#402-404)
totalSupply() should be declared external:
        - BEP20.totalSupply() (asva.sol#409-411)
balanceOf(address) should be declared external:
        - BEP20.balanceOf(address) (asva.sol#416-418)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (asva.sol#428-431)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (asva.sol#447-450)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (asva.sol#465-469)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (asva.sol#483-486)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (asva.sol#502-505)
burn(uint256) should be declared external:
        - BEP20Burnable.burn(uint256) (asva.sol#638-640)
burnFrom(address,uint256) should be declared external:
        - BEP20Burnable.burnFrom(address,uint256) (asva.sol#653-658)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (asva.sol#899-902)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (asva.sol#908-912)
fallabck() should be declared external:
        - ASVA.fallabck() (asva.sol#978-980)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

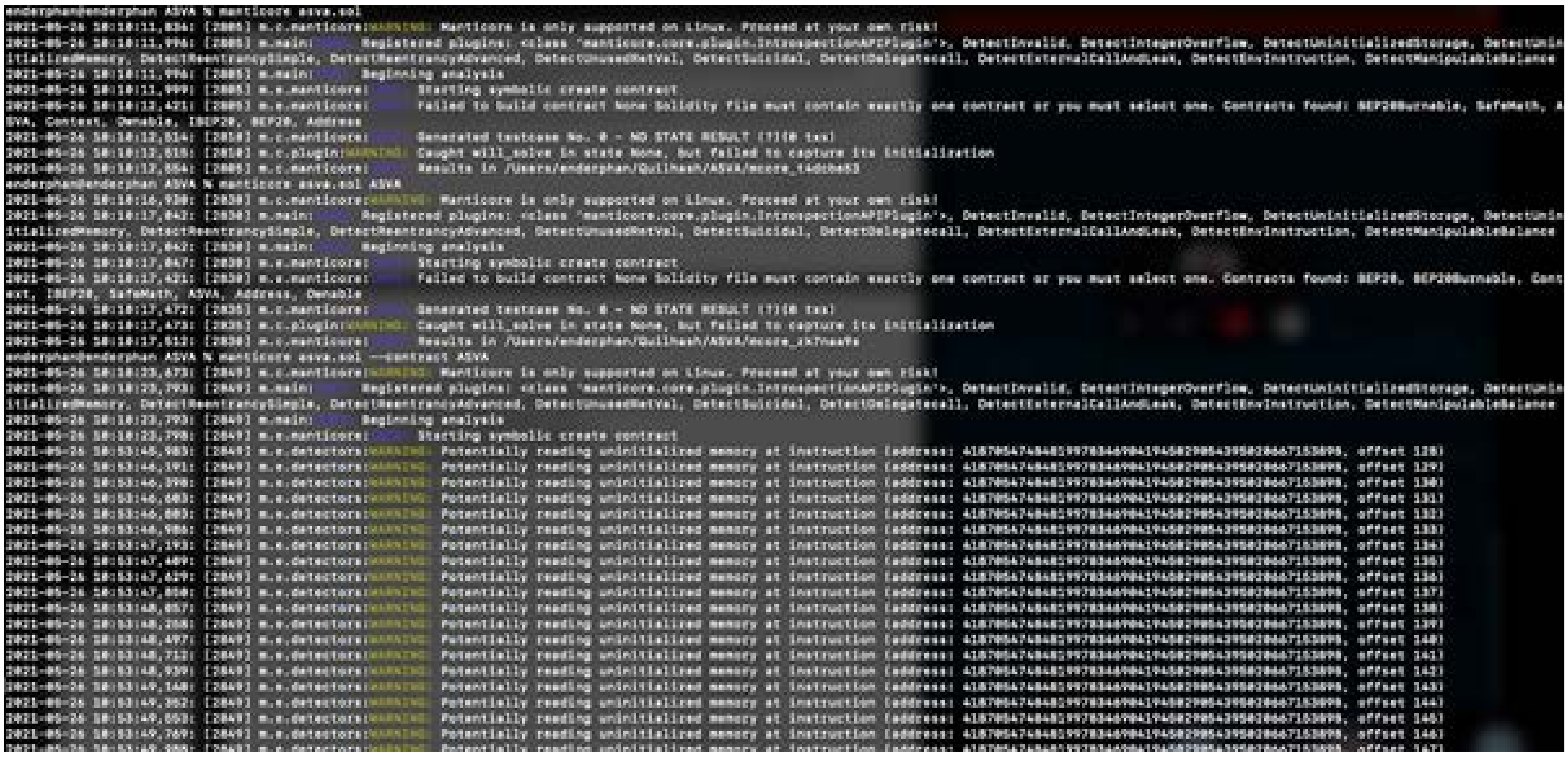
Theo

```
enderphan@enderphan ASVA % theo --rpc-http TTP://127.0.0.1:7545
The account's private key (input hidden)
>
Contract to interact with
> 0xd46C6Dca856ee8BC2c4Dcbe078001adFe6cbFBC8
Scanning for exploits in contract: 0xd46C6Dca856ee8BC2c4Dcbe078001adFe6cbFBC8
Could not connect to RPC server. Make sure that your node is running and that RPC parameters are set correctly.
No exploits found. You're going to need to load some exploits.

Tools available in the console:
- `exploits` is an array of loaded exploits found by Mythril or read from a file
- `w3` an initialized instance of web3py for the provided HTTP RPC endpoint
- `dump()` writing a json representation of an object to a local file

Check the readme for more info:
https://github.com/cleanunicorn/theo

Theo version v0.8.2.


>>>
```

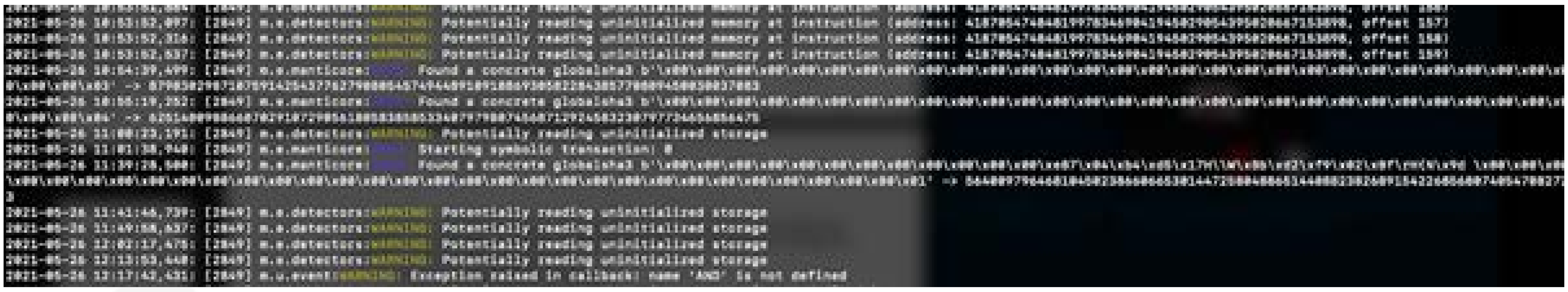
Mythril

```
enderphan@enderphan ASVA % myth a asva.sol
The analysis was completed successfully. No issues were detected.

enderphan@enderphan ASVA %
```

Manticore



```
enderphan@enderphan ASVA % manticore asva.sol
2021-05-26 10:10:11,834: [2805] m.c.manticore:WARNING: Manticore is only supported on Linux. Proceed at your own risk!
2021-05-26 10:10:11,996: [2805] m.main:INFO: Registered plugins: <class 'manticore.core.plugin.IntrospectionAPIPlugin'>, DetectInvalid, DetectIntegerOverflow, DetectUninitializedStorage, DetectUnin
itializedMemory, DetectReentrancySimple, DetectReentrancyAdvanced, DetectUnusedRetVal, DetectSuicidal, DetectDelegatecall, DetectExternalCallAndLeak, DetectEnvInstruction, DetectManipulableBalance
2021-05-26 10:10:11,996: [2805] m.main:INFO: Beginning analysis
2021-05-26 10:10:11,999: [2805] m.e.manticore:INFO: Starting symbolic create contract
2021-05-26 10:10:12,421: [2805] m.e.manticore:INFO: Failed to build contract None Solidity file must contain exactly one contract or you must select one. Contracts found: BEP20Burnable, SafeMath, A
SVA, Context, Ownable, IBEP20, BEP20, Address
2021-05-26 10:10:12,514: [2818] m.c.manticore:INFO: Generated testcase No. 0 - NO STATE RESULT (?)(0 txs)
2021-05-26 10:10:12,515: [2818] m.c.plugin:WARNING: Caught will_solve in state None, but failed to capture its initialization
2021-05-26 10:10:12,554: [2805] m.c.manticore:INFO: Results in /Users/enderphan/Quillhash/ASVA/mcore_t4dcba53
enderphan@enderphan ASVA % manticore asva.sol ASVA
2021-05-26 10:10:16,930: [2830] m.c.manticore:WARNING: Manticore is only supported on Linux. Proceed at your own risk!
2021-05-26 10:10:17,042: [2830] m.main:INFO: Registered plugins: <class 'manticore.core.plugin.IntrospectionAPIPlugin'>, DetectInvalid, DetectIntegerOverflow, DetectUninitializedStorage, DetectUnin
itializedMemory, DetectReentrancySimple, DetectReentrancyAdvanced, DetectUnusedRetVal, DetectSuicidal, DetectDelegatecall, DetectExternalCallAndLeak, DetectEnvInstruction, DetectManipulableBalance
2021-05-26 10:10:17,042: [2830] m.main:INFO: Beginning analysis
2021-05-26 10:10:17,047: [2830] m.e.manticore:INFO: Starting symbolic create contract
2021-05-26 10:10:17,421: [2830] m.e.manticore:INFO: Failed to build contract None Solidity file must contain exactly one contract or you must select one. Contracts found: BEP20, BEP20Burnable, Cont
ext, IBEP20, SafeMath, ASVA, Address, Ownable
2021-05-26 10:10:17,472: [2835] m.c.manticore:INFO: Generated testcase No. 0 - NO STATE RESULT (?)(0 txs)
2021-05-26 10:10:17,473: [2835] m.c.plugin:WARNING: Caught will_solve in state None, but failed to capture its initialization
2021-05-26 10:10:17,512: [2830] m.c.manticore:INFO: Results in /Users/enderphan/Quillhash/ASVA/mcore_zk7nae9s
enderphan@enderphan ASVA % manticore asva.sol --contract ASVA
2021-05-26 10:10:23,673: [2849] m.c.manticore:WARNING: Manticore is only supported on Linux. Proceed at your own risk!
2021-05-26 10:10:23,793: [2849] m.main:INFO: Registered plugins: <class 'manticore.core.plugin.IntrospectionAPIPlugin'>, DetectInvalid, DetectIntegerOverflow, DetectUninitializedStorage, DetectUnin
itializedMemory, DetectReentrancySimple, DetectReentrancyAdvanced, DetectUnusedRetVal, DetectSuicidal, DetectDelegatecall, DetectExternalCallAndLeak, DetectEnvInstruction, DetectManipulableBalance
2021-05-26 10:10:23,793: [2849] m.main:INFO: Beginning analysis
2021-05-26 10:10:23,798: [2849] m.e.manticore:INFO: Starting symbolic create contract
2021-05-26 10:53:45,985: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 128)
2021-05-26 10:53:46,191: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 129)
2021-05-26 10:53:46,398: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 130)
2021-05-26 10:53:46,603: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 131)
2021-05-26 10:53:46,803: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 132)
2021-05-26 10:53:46,986: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 133)
2021-05-26 10:53:47,193: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 134)
2021-05-26 10:53:47,409: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 135)
2021-05-26 10:53:47,629: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 136)
2021-05-26 10:53:47,850: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 137)
2021-05-26 10:53:48,057: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 138)
2021-05-26 10:53:48,268: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 139)
2021-05-26 10:53:48,497: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 140)
2021-05-26 10:53:48,712: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 141)
2021-05-26 10:53:48,939: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 142)
2021-05-26 10:53:49,148: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 143)
2021-05-26 10:53:49,352: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 144)
2021-05-26 10:53:49,553: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 145)
2021-05-26 10:53:49,769: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 146)
```



```
2021-05-26 10:53:52,097: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 157)
2021-05-26 10:53:52,316: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 158)
2021-05-26 10:53:52,537: [2849] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 418705474848199783469841945829054395820667153898, offset 159)
2021-05-26 10:54:39,499: [2849] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x03' -> 87985029871070914254377627900004574944091091006930502284385779000945003037083
2021-05-26 10:55:19,252: [2849] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x06' -> 62014000986660702910729056100583805053348797900745607129245832307977246566475
2021-05-26 11:00:23,191: [2849] m.e.detectors:WARNING: Potentially reading uninitialized storage
2021-05-26 11:01:38,940: [2849] m.e.manticore:INFO: Starting symbolic transaction: 0
2021-05-26 11:39:28,500: [2849] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xe0?\x04\xb4\xd8\x17M\W\x8b\xd2\xf9\x02\x9f\rn(N\x9d \x00\x00\x0
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01' -> 56400979646010450238660665301447250048865144008230260915422606600740547082
3
2021-05-26 11:41:46,739: [2849] m.e.detectors:WARNING: Potentially reading uninitialized storage
2021-05-26 11:49:58,637: [2849] m.e.detectors:WARNING: Potentially reading uninitialized storage
2021-05-26 12:02:17,475: [2849] m.e.detectors:WARNING: Potentially reading uninitialized storage
2021-05-26 12:13:53,440: [2849] m.e.detectors:WARNING: Potentially reading uninitialized storage
2021-05-26 12:17:42,431: [2849] m.u.event:WARNING: Exception raised in callback: name 'AND' is not defined
```

Solhint Linter

```
asva.sol:925:30: Error: Constant name must be in capitalized SNAKE_CASE
asva.sol:927:29: Error: Constant name must be in capitalized SNAKE_CASE
asva.sol:929:29: Error: Constant name must be in capitalized SNAKE_CASE
asva.sol:931:29: Error: Constant name must be in capitalized SNAKE_CASE
asva.sol:933:29: Error: Constant name must be in capitalized SNAKE_CASE
asva.sol:935:29: Error: Constant name must be in capitalized SNAKE_CASE
asva.sol:937:29: Error: Constant name must be in capitalized SNAKE_CASE
asva.sol:939:29: Error: Constant name must be in capitalized SNAKE_CASE
asva.sol:941:29: Error: Constant name must be in capitalized SNAKE_CASE
asva.sol:943:29: Error: Constant name must be in capitalized SNAKE_CASE
asva.sol:945:29: Error: Constant name must be in capitalized SNAKE_CASE
```

```
asva.sol:5:1: Error: Compiler version >=0.6.0 <0.8.0 does not satisfy the
r semver requirement
```

```
asva.sol:28:1: Error: Compiler version >=0.6.0 <0.8.0 does not satisfy
the r semver requirement
```

```
asva.sol:104:1: Error: Compiler version >=0.6.0 <0.8.0 does not satisfy
the r semver requirement
```

```
asva.sol:317:1: Error: Compiler version >=0.6.0 <0.8.0 does not satisfy
the r semver requirement
```

```
asva.sol:618:94: Error: Code contains empty blocks
```

```
asva.sol:621:1: Error: Compiler version >=0.6.0 <0.8.0 does not satisfy
the r semver requirement
```

```
asva.sol:661:1: Error: Compiler version >=0.6.2 <0.8.0 does not satisfy
the r semver requirement
```

```
asva.sol:849:1: Error: Compiler version >=0.6.0 <0.8.0 does not satisfy
the r semver requirement
```

```
asva.sol:917:1: Error: Compiler version 0.7.6 does not satisfy the r
semver requirement
```

Solidity Static Analysis

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 772:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 691:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 838:16:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

more

Pos: 715:27:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

more

Pos: 777:50:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.

more

Pos: 825:50:

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Re-entrancy or Back-Door Entry were found in the contract.

A medium severity issue, a low severity issue, and four informational issues were discovered during the audit. It is recommended to kindly go through the above-mentioned details and fix the code accordingly.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the ASVA platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the ASVA Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Asva Labs

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com