Armors Labs

XCOM

Smart Contract Audit

XCOM Audit Summary

Project name : XCOM Contract

Project address: None

Code URL : None

Commit : None

Projct target : XCOM Contract Audit

Blockchain : Huobi ECO Chain（Heco）

Test result : PASSED

Audit Info

Audit NO : 0X202104150008

Audit Team : Armors Labs

Audit Proofreading: https://armors.io/#project-cases

XCOM Audit

The XCOM team asked us to review and audit their XCOM contract. We looked at the code and now publish our results.

Here is our assessment and recommendations, in order of importance.

Document information

Name	Auditor	Version	Date
XCOM Audit	Rock, Hosea, Rushairer, Rico, David, Alice	1.0.0	2021-04-15

Audit results

Note that this project is very similar to the aave/protocol-v2 project.

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the XCOM contract. The above should not be construed as investment advice.

Based on the widely recognized security status of the current underlying blockchain and smart contract, this audit report is valid for 18 months from the date of output.

(Statement: Armors Labs reports only on facts that have occurred or existed before this report is issued and assumes corresponding responsibilities. Armors Labs is not able to determine the security of its smart contracts and is not responsible for any subsequent or existing facts after this report is issued. The security audit analysis and other content of this report are only based on the documents and information provided by the information provider to Armors Labs at the time of issuance of this report (" information provided " for short). Armors Labs postulates that the information provided is not missing, tampered, deleted or hidden. If the information provided is missing, tampered, deleted, hidden or reflected in a way that is not consistent with the actual situation, Armors Labs shall not be responsible for the losses and adverse effects caused.)

Audited target file

file	md5
./mocks/oracle/GenericOracleI.sol	5a53cc0b4d73f6587b7151710be72fc9
./mocks/oracle/LendingRateOracle.sol	d43c7781f902fb9aff858d7a1d5cfde3
./mocks/oracle/IExtendedPriceAggregator.sol	03959a5a1fa940eacb23f234b5706a92
./mocks/oracle/ChainlinkUSDETHOracleI.sol	2c81bceee227fa7898f696f3cffd0abf
./mocks/oracle/PriceOracle.sol	2593d848f2b4fd8909c77f41eccc907f
./mocks/oracle/CLAggregators/MockAggregator.sol	14719f5b7dc1aa0718c6d8b1fa8c66ec
./mocks/swap/MockUniswapV2Router02.sol	99a2dca6d1d755059e31da5e751b0ab7
./mocks/flashloan/MockFlashLoanReceiver.sol	6078d580dd7157b6831bc0215b87978e
./mocks/dependencies/weth/WETH9.sol	5175625bc241ea512297dba1b5711d06
./mocks/tokens/MintableERC20.sol	79dd5fd193bd7e8ee1dfb805489712fd
./mocks/tokens/WETH9Mocked.sol	11ba6b97c698a623974d2b9b4c420629
./mocks/tokens/MintableDelegationERC20.sol	3ac76949d15be4c32fb63f4341fc5fa8
./mocks/attacks/SefldestructTransfer.sol	ed16f0edec9773b8bc25fc80e145c3c0
./mocks/upgradeability/MockAToken.sol	6295cd444714e8621a695d0d8a6c5d3b
./mocks/upgradeability/MockStableDebtToken.sol	4ab5a951ea6ba42e9aab3f3a33bb04fd
./mocks/upgradeability/MockVariableDebtToken.sol	9d0b10d9a3cae8bc3ace26265574aa55
./misc/WalletBalanceProvider.sol	3f8f7c5fe582fc84f541ca363efb5a3e
./misc/WETHGateway.sol	cb2cfb673f837bf5a8a3c74bf73365fa
./misc/AaveOracle.sol	b3426d7969d83741b47664f79f9b919b
./misc/AaveProtocolDataProvider.sol	8b8de07327a400bba8b27b197762f8fb
./misc/UiPoolDataProvider.sol	ffa70b16406144cff4ac72fb7b038493
./misc/interfaces/IWETHGateway.sol	d56a0efb3776bfcbb0e887dcf33d5016
./misc/interfaces/IUiPoolDataProvider.sol	a8366845fd50b1a4bd870fb5b3fbc98d
./misc/interfaces/IWETH.sol	ab2af8b1175a4e9dfdf3f7754dbee9a5
./misc/interfaces/IUniswapV2Router01.sol	4e9a370698f8a3b06afe7e5f89d85186
./misc/interfaces/IUniswapV2Router02.sol	d75fbd3aaa9e4f38730e4faf2b8fbe56
./misc/interfaces/IERC20DetailedBytes.sol	ca7eceb3285ab76efb59fd93d468ad41
./flashloan/base/FlashLoanReceiverBase.sol	342099eb47e5ca9af28b333264a5a149

file	md5
./flashloan/interfaces/IFlashLoanReceiver.sol	5e569987e7faa6a858ede0e47939966b
./dep.../ope../contracts/SafeERC20.sol	7274eaf096edf0d7dde9b3282e4fcb42
./dep.../ope../contracts/Context.sol	35c596d58cea32ae629aeb27ad2ead6c
./dep.../ope../contracts/SafeMath.sol	cf6dbbc25d7163c47e5260b755361955
./dep.../ope../contracts/Ownable.sol	42b1116808efc7ee5bd735beefb0c3a0
./dep.../ope../contracts/Address.sol	bdb6e9218c297368a6ccd434a6ceb0ff
./dep.../ope../contracts/ERC20.sol	1c9ef76d716ea1e56f8e93d99a49bb10
./dep.../ope../contracts/IERC20.sol	91c0afdd4eb122474c0d0c040dab794a
./dep.../ope../contracts/IERC20Detailed.sol	0448c76ea7803fca57646a5beb17e8a2
./dep.../ope../upgradeability/UpgradeabilityProxy.sol	0f693cbb0e05cb04aa30f79b105385b1
./dep.../ope../upgradeability/AdminUpgradeabilityProxy.sol	1895075e0d13e7bfddc4deee8a19a368
./dep.../ope../upgradeability/Initializable.sol	13ee7092ae365253018384630bc40f37
./dep.../ope../upgradeability/InitializableUpgradeabilityProxy.sol	4578675ddb0759973506da311540ec82
./dep.../ope../upgradeability/Proxy.sol	ebf21fde26904e77797d7ada834c29e5
./dep.../ope../upgradeability/BaseUpgradeabilityProxy.sol	843dbecc88a254c6e02012b1efb48bcc
./dep.../ope../upgradeability/InitializableAdminUpgradeabilityProxy.sol	02d15f5e79d44c710cb65d24f96e8a61
./dep.../ope../upgradeability/BaseAdminUpgradeabilityProxy.sol	2bd0cc4dc7392dc2c414ca903c3e2ada
./protocol/configuration/LendingPoolAddressesProviderRegistry.sol	15a1e88b6cd45ae8e6d6bac72f9e87be
./protocol/configuration/LendingPoolAddressesProvider.sol	c7a933bb7c54957268e6889c75ba6158
./protocol/libraries/configuration/ReserveConfiguration.sol	c0f46e3d5df412e9da76cd291344f026
./protocol/libraries/configuration/UserConfiguration.sol	14cd911a7c407bdc1b0d23cb7a258a54
./protocol/libraries/types/DataTypes.sol	2127b8d36674faba96bc703f72164a1a
./protocol/libraries/logic/ReserveLogic.sol	d15ddc108407134c8293813a2bf3d2bb
./protocol/libraries/logic/ValidationLogic.sol	956cfd1dde7c76c40951271fc7ab70aa
./protocol/libraries/logic/GenericLogic.sol	63c0b6b808431eddcdd4f72f15116e1c
./protocol/libraries/math/MathUtils.sol	11581a64630b7297869b359ac31927e7
./protocol/libraries/math/WadRayMath.sol	bf062018f7a0f55c2c54909629b3028d
./protocol/libraries/math/PercentageMath.sol	ac3f1f6dfed160907baca58211bb6d37
./pro.../lib.../aav.../BaseImmutableAdminUpgradeabilityProxy.sol	c2f279451bae93b05a05b4dfe73965cc
./pro.../lib.../aav.../InitializableImmutableAdminUpgradeabilityProxy.sol	b6341ef33dcb8301a510795e68fdfb47
./pro.../lib.../aav.../VersionedInitializable.sol	e46d1675e7ec7446ef0536ab7c3db963
./protocol/libraries/helpers/Helpers.sol	2c6f19fbad822bd9fc72c2012188eedc

file	md5
./protocol/libraries/helpers/Errors.sol	f9f147c0810b2cf78866930180648be5
./protocol/tokenization/VariableDebtToken.sol	46ae51546a44c7d66e49bb017eeb4985
./protocol/tokenization/AToken.sol	618e6f62742873950a54f2d6cbe4950c
./protocol/tokenization/StableDebtToken.sol	e4103247d379ff0070b87ce4bf9fa4d4
./protocol/tokenization/IncentivizedERC20.sol	2eaf57d54f36c3017a3596f77be8532d
./protocol/tokenization/DelegationAwareAToken.sol	04f040cdfc28acce298999828702fa64
./protocol/tokenization/base/DebtTokenBase.sol	c7a46dce60aa28172ccb12e1eee3143d
./protocol/lendingpool/LendingPoolCollateralManager.sol	bd0641b34d2bdf3a679abf9e1f34acea
./protocol/lendingpool/LendingPoolConfigurator.sol	a450e1be0037ab48cc05717afb1e6889
./protocol/lendingpool/DefaultReserveInterestRateStrategy.sol	18d3168edac10e76ae1214192ac4483d
./protocol/lendingpool/LendingPoolStorage.sol	71ed7efdbf03ef30cf22a32e94aee84e
./protocol/lendingpool/LendingPool.sol	b9d1d1e3f8c43d0a15e7d4474b1e411b
./adapters/BaseUniswapAdapter.sol	865cf347d6d906fb50a90f811e6a42fc
./adapters/FlashLiquidationAdapter.sol	32701d733ab5ba66408cb8939382aa2f
./adapters/UniswapRepayAdapter.sol	6fd39f9dbf395553165724c6e0654eea
./adapters/UniswapLiquiditySwapAdapter.sol	11dbb42a3bae79b9f0ca43fcca69625c
./adapters/interfaces/IBaseUniswapAdapter.sol	a62c64a0a9901e9f9806fb32b0366a01
./interfaces/IERC20WithPermit.sol	cd97ed0a4faa085377a953815750d216
./interfaces/IScaledBalanceToken.sol	05f5c83b51e4b04e17785f92ca7840de
./interfaces/IChainlinkAggregator.sol	7a25050278be7e9f3ac7424c0d69a049
./interfaces/IInitializableDebtToken.sol	ae4d2dc34379e54d05226248c41da496
./interfaces/IVariableDebtToken.sol	297a4e07a70791964b5e5b96fee1572a
./interfaces/ILendingRateOracle.sol	dc30a43ed82b7bfe230593226324c891
./interfaces/ILendingPoolCollateralManager.sol	7b3bff7810d10b985d9e2fca06efb374
./interfaces/IUniswapExchange.sol	02272f4b5f947c3607168fc3eaaaa29d
./interfaces/IReserveInterestRateStrategy.sol	b11d9c14920ab9302b232bd3f7226e36
./interfaces/IDelegationToken.sol	082e2de4fca3cfa794b548eb64959089
./interfaces/IAaveIncentivesController.sol	1e9878616988e4923ea5563cc7d7e921
./interfaces/ILendingPoolAddressesProvider.sol	a487bfd067ce6e71ddc7e9cc91302bae
./interfaces/IUniswapV2Router02.sol	e6a3fa068b9ccc4f9d95180fbc8536f4
./interfaces/ILendingPool.sol	c9d32b9924ef08162c2453bcb64c569a
./interfaces/IPriceOracle.sol	2f743812a3f5003b3a8aad85caa6e335

file	md5
./interfaces/ICreditDelegationToken.sol	9d4c60aae9eb8093c9475ab517bdcd99
./interfaces/IInitializableAToken.sol	b408cb493b8651eca053fc831bea1cd8
./interfaces/IPriceOracleGetter.sol	56181b85c44591f931b25fa0b92ed581
./interfaces/IExchangeAdapter.sol	0b4971bce9c94dba30de0d4f92394ebe
./interfaces/ILendingPoolAddressesProviderRegistry.sol	249239470e9a5ff454bf3c1aa2f5bfd3
./interfaces/ILendingPoolConfigurator.sol	26bc9369a4456e17626fe2c3a8a6d6db
./interfaces/IStableDebtToken.sol	ce31ee6194c17b0b68078bcade919040
./interfaces/IAToken.sol	325b239ba3bede374aaa1931b90550d5
./deployments/ATokensAndRatesHelper.sol	cd58ce56df80332597f200f2c512d9a1
./deployments/StringLib.sol	062b80046c48b7a0945936544e868b62
./deployments/StableAndVariableTokensHelper.sol	54833bad78c142b27ac0ec19df58faf0

Vulnerability analysis

Vulnerability distribution

vulnerability level	number
Critical severity	0
High severity	0
Medium severity	0
Low severity	0

Summary of audit results

Vulnerability	status
Re-Entrancy	safe
Arithmetic Over/Under Flows	safe
Unexpected Blockchain Currency	safe
Delegatecall	safe
Default Visibilities	safe
Entropy Illusion	safe
External Contract Referencing	safe
Short Address/Parameter Attack	safe

Vulnerability	status
Unchecked CALL Return Values	safe
Race Conditions / Front Running	safe
Denial Of Service (DOS)	safe
Block Timestamp Manipulation	safe
Constructors with Care	safe
Unintialised Storage Pointers	safe
Floating Points and Numerical Precision	safe
tx.origin Authentication	safe

Contract file

```
.
├── adapters
│   ├── BaseUniswapAdapter.sol
│   ├── FlashLiquidationAdapter.sol
│   ├── UniswapLiquiditySwapAdapter.sol
│   ├── UniswapRepayAdapter.sol
│   └── interfaces
│       └── IBaseUniswapAdapter.sol
├── dependencies
│   └── openzeppelin
│       ├── contracts
│       │   ├── Address.sol
│       │   ├── Context.sol
│       │   ├── ERC20.sol
│       │   ├── IERC20.sol
│       │   ├── IERC20Detailed.sol
│       │   ├── Ownable.sol
│       │   ├── SafeERC20.sol
│       │   └── SafeMath.sol
│       └── upgradeability
│           ├── AdminUpgradeabilityProxy.sol
│           ├── BaseAdminUpgradeabilityProxy.sol
│           ├── BaseUpgradeabilityProxy.sol
│           ├── Initializable.sol
│           ├── InitializableAdminUpgradeabilityProxy.sol
│           ├── InitializableUpgradeabilityProxy.sol
│           ├── Proxy.sol
│           └── UpgradeabilityProxy.sol
├── deployments
│   ├── ATokensAndRatesHelper.sol
│   ├── StableAndVariableTokensHelper.sol
│   └── StringLib.sol
├── flashloan
│   ├── base
│   │   └── FlashLoanReceiverBase.sol
│   └── interfaces
│       └── IFlashLoanReceiver.sol
├── interfaces
│   ├── IAToken.sol
│   ├── IAaveIncentivesController.sol
│   ├── IChainlinkAggregator.sol
│   ├── ICreditDelegationToken.sol
│   ├── IDelegationToken.sol
│   ├── IERC20WithPermit.sol
│   ├── IExchangeAdapter.sol
│   ├── IInitializableAToken.sol
│   ├── IInitializableDebtToken.sol
│   ├── ILendingPool.sol
│   ├── ILendingPoolAddressesProvider.sol
│   ├── ILendingPoolAddressesProviderRegistry.sol
│   ├── ILendingPoolCollateralManager.sol
│   ├── ILendingPoolConfigurator.sol
│   ├── ILendingRateOracle.sol
│   ├── IPriceOracle.sol
│   ├── IPriceOracleGetter.sol
│   ├── IReserveInterestRateStrategy.sol
│   ├── IScaledBalanceToken.sol
│   ├── IStableDebtToken.sol
│   ├── IUniswapExchange.sol
│   ├── IUniswapV2Router02.sol
│   └── IVariableDebtToken.sol
├── misc
│   ├── AaveOracle.sol
│   ├── AaveProtocolDataProvider.sol
│   ├── UiPoolDataProvider.sol
│   ├── WETHGateway.sol
│   ├── WalletBalanceProvider.sol
│   └── interfaces
│       ├── IERC20DetailedBytes.sol
│       ├── IUiPoolDataProvider.sol
│       ├── IUniswapV2Router01.sol
│       ├── IUniswapV2Router02.sol
│       ├── IWETH.sol
│       └── IWETHGateway.sol
├── mocks
│   ├── attacks
│   │   └── SefldestructTransfer.sol
│   ├── dependencies
│   │   └── weth
│   │       └── WETH9.sol
│   ├── flashloan
│   │   └── MockFlashLoanReceiver.sol
│   ├── oracle
│   │   ├── CLAggregators
│   │   │   └── MockAggregator.sol
│   │   ├── ChainlinkUSDETHOracleI.sol
│   │   ├── GenericOracleI.sol
│   │   ├── IExtendedPriceAggregator.sol
│   │   ├── LendingRateOracle.sol
│   │   └── PriceOracle.sol
│   ├── swap
│   │   └── MockUniswapV2Router02.sol
│   ├── tokens
│   │   ├── MintableDelegationERC20.sol
│   │   ├── MintableERC20.sol
│   │   └── WETH9Mocked.sol
│   └── upgradeability
│       ├── MockAToken.sol
│       ├── MockStableDebtToken.sol
│       └── MockVariableDebtToken.sol
└── protocol
    ├── configuration
    │   ├── LendingPoolAddressesProvider.sol
    │   └── LendingPoolAddressesProviderRegistry.sol
    ├── lendingpool
    │   ├── DefaultReserveInterestRateStrategy.sol
    │   ├── LendingPool.sol
    │   ├── LendingPoolCollateralManager.sol
    │   ├── LendingPoolConfigurator.sol
    │   └── LendingPoolStorage.sol
    ├── libraries
    │   ├── aave-upgradeability
    │   │   ├── BaseImmutableAdminUpgradeabilityProxy.sol
    │   │   ├── InitializableImmutableAdminUpgradeabilityProxy.sol
    │   │   └── VersionedInitializable.sol
    │   ├── configuration
    │   │   ├── ReserveConfiguration.sol
    │   │   └── UserConfiguration.sol
    │   ├── helpers
    │   │   ├── Errors.sol
    │   │   └── Helpers.sol
    │   ├── logic
    │   │   ├── GenericLogic.sol
    │   │   ├── ReserveLogic.sol
    │   │   └── ValidationLogic.sol
    │   ├── math
    │   │   ├── MathUtils.sol
    │   │   ├── PercentageMath.sol
    │   │   └── WadRayMath.sol
    │   └── types
    │       └── DataTypes.sol
    └── tokenization
        ├── AToken.sol
        ├── DelegationAwareAToken.sol
        ├── IncentivizedERC20.sol
        ├── StableDebtToken.sol
        ├── VariableDebtToken.sol
        └── base
            └── DebtTokenBase.sol

35 directories, 103 files
```

Analysis of audit results

Re-Entrancy

- **Description:**
  One of the features of smart contracts is the ability to call and utilise code of other external contracts. Contracts also typically handle Blockchain Currency, and as such often send Blockchain Currency to various external user addresses. The operation of calling external contracts, or sending Blockchain Currency to an address, requires the contract to submit an external call. These external calls can be hijacked by attackers whereby they force the contract to execute further code (i.e. through a fallback function) , including calls back into itself. Thus the code execution "re-enters" the contract. Attacks of this kind were used in the infamous DAO hack.
- **Detection results:**

  ```
  PASSED!
  ```

- **Security suggestion:**
  no.

Arithmetic Over/Under Flows

- **Description:**
  The Virtual Machine (EVM) specifies fixed-size data types for integers. This means that an integer variable, only has a certain range of numbers it can represent. A uint8 for example, can only store numbers in the range

[0,255]. Trying to store 256 into a uint8 will result in 0. If care is not taken, variables in Solidity can be exploited if user input is unchecked and calculations are performed which result in numbers that lie outside the range of the data type that stores them.

- **Detection results:**

  ```
  PASSED!
  ```

- **Security suggestion:**
  no.

Unexpected Blockchain Currency

- **Description:**
  Typically when Blockchain Currency is sent to a contract, it must execute either the fallback function, or another function described in the contract. There are two exceptions to this, where Blockchain Currency can exist in a contract without having executed any code. Contracts which rely on code execution for every Blockchain Currency sent to the contract can be vulnerable to attacks where Blockchain Currency is forcibly sent to a contract.
- **Detection results:**

  ```
  PASSED!
  ```

- **Security suggestion:** no.

Delegatecall

- **Description:**
  The CALL and DELEGATECALL opcodes are useful in allowing developers to modularise their code. Standard external message calls to contracts are handled by the CALL opcode whereby code is run in the context of the external contract/function. The DELEGATECALL opcode is identical to the standard message call, except that the code executed at the targeted address is run in the context of the calling contract along with the fact that msg.sender and msg.value remain unchanged. This feature enables the implementation of libraries whereby developers can create reusable code for future contracts.
- **Detection results:**

  ```
  PASSED!
  ```

- **Security suggestion:** no.

Default Visibilities

- **Description:**
  Functions in Solidity have visibility specifiers which dictate how functions are allowed to be called. The visibility determines whBlockchain Currency a function can be called externally by users, by other derived contracts, only internally or only externally. There are four visibility specifiers, which are described in detail in the Solidity Docs. Functions default to public allowing users to call them externally. Incorrect use of visibility specifiers can lead to some devestating vulnerabilities in smart contracts as will be discussed in this section.
- **Detection results:**

  ```
  PASSED!
  ```

- **Security suggestion:**
  no.

Entropy Illusion

- **Description:**
  All transactions on the blockchain are deterministic state transition operations. Meaning that every transaction modifies the global state of the ecosystem and it does so in a calculable way with no uncertainty. This ultimately means that inside the blockchain ecosystem there is no source of entropy or randomness. There is no rand() function in Solidity. Achieving decentralised entropy (randomness) is a well established problem and many ideas have been proposed to address this (see for example, RandDAO or using a chain of Hashes as described by Vitalik in this post).
- **Detection results:**

```
PASSED!
```

- **Security suggestion:**
  no.

External Contract Referencing

- **Description:**
  One of the benefits of the global computer is the ability to re-use code and interact with contracts already deployed on the network. As a result, a large number of contracts reference external contracts and in general operation use external message calls to interact with these contracts. These external message calls can mask malicious actors intentions in some non-obvious ways, which we will discuss.
- **Detection results:**

```
PASSED!
```

- **Security suggestion:**
  no.

Unsolved TODO comments

- **Description:**
  Check for Unsolved TODO comments
- **Detection results:**

```
PASSED!
```

- **Security suggestion:**
  no.

Short Address/Parameter Attack

- **Description:**
  This attack is not specifically performed on Solidity contracts themselves but on third party applications that may interact with them. I add this attack for completeness and to be aware of how parameters can be manipulated in contracts.

- **Detection results:**

```
PASSED!
```

- **Security suggestion:**
  no.

Unchecked CALL Return Values

- **Description:**
  There a number of ways of performing external calls in solidity. Sending Blockchain Currency to external accounts is commonly performed via the transfer() method. However, the send() function can also be used and, for more versatile external calls, the CALL opcode can be directly employed in solidity. The call() and send() functions return a boolean indicating if the call succeeded or failed. Thus these functions have a simple caveat, in that the transaction that executes these functions will not revert if the external call (intialised by call() or send()) fails, rather the call() or send() will simply return false. A common pitfall arises when the return value is not checked, rather the developer expects a revert to occur.
- **Detection results:**

```
PASSED!
```

- **Security suggestion:**
  no.

Race Conditions / Front Running

- **Description:**
  The combination of external calls to other contracts and the multi-user nature of the underlying blockchain gives rise to a variety of potential Solidity pitfalls whereby users race code execution to obtain unexpected states. Re-Entrancy is one example of such a race condition. In this section we will talk more generally about different kinds of race conditions that can occur on the blockchain. There is a variety of good posts on this subject, a few are: Wiki - Safety, DASP - Front-Running and the Consensus - Smart Contract Best Practices.
- **Detection results:**

```
PASSED!
```

- **Security suggestion:**
  no.

Denial Of Service (DOS)

- **Description:**
  This category is very broad, but fundamentally consists of attacks where users can leave the contract inoperable for a small period of time, or in some cases, permanently. This can trap Blockchain Currency in these contracts forever, as was the case with the Second Parity MultiSig hack
- **Detection results:**

```
PASSED!
```

- **Security suggestion:**
  no.

Block Timestamp Manipulation

- **Description:**
  Block timestamps have historically been used for a variety of applications, such as entropy for random numbers (see the Entropy Illusion section for further details), locking funds for periods of time and various state-changing conditional statements that are time-dependent. Miner's have the ability to adjust timestamps slightly which can prove to be quite dangerous if block timestamps are used incorrectly in smart contracts.
- **Detection results:**

  ```
  PASSED!
  ```

- **Security suggestion:**
  no.

Constructors with Care

- **Description:**
  Constructors are special functions which often perform critical, privileged tasks when initialising contracts. Before solidity v0.4.22 constructors were defined as functions that had the same name as the contract that contained them. Thus, when a contract name gets changed in development, if the constructor name isn't changed, it becomes a normal, callable function. As you can imagine, this can (and has) lead to some interesting contract hacks.
- **Detection results:**

  ```
  PASSED!
  ```

- **Security suggestion:**
  no.

Unintialised Storage Pointers

- **Description:**
  The EVM stores data either as storage or as memory. Understanding exactly how this is done and the default types for local variables of functions is highly recommended when developing contracts. This is because it is possible to produce vulnerable contracts by inappropriately intialising variables.
- **Detection results:**

  ```
  PASSED!
  ```

- **Security suggestion:**
  no.

Floating Points and Numerical Precision

- **Description:**
  As of this writing (Solidity v0.4.24), fixed point or floating point numbers are not supported. This means that floating point representations must be made with the integer types in Solidity. This can lead to errors/vulnerabilities if not implemented correctly.
- **Detection results:**

```
PASSED !
```

- **Security suggestion:**
  no.

tx.origin Authentication

- **Description:**
  Solidity has a global variable, tx.origin which traverses the entire call stack and returns the address of the account that originally sent the call (or transaction). Using this variable for authentication in smart contracts leaves the contract vulnerable to a phishing-like attack.
- **Detection results:**

```
PASSED !
```

- **Security suggestion:**
  no.

armors.io

contact@armors.io