
Armors Labs
HaykerDAO
Smart Contract Audit

HaykerDAO Audit Summary

Project name : HaykerDAO Contract

Project address: http://www.hkrdao.com

Code URL : https://github.com/HaykerDAO

Projct target : HaykerDAO Contract Audit

Blockchain : Huobi ECO Chain（Heco）

Test result : PASSED

Audit Info

Audit NO : 0X202104210006

Audit Team : Armors Labs

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

HaykerDAO Audit

The HaykerDAO team asked us to review and audit their HaykerDAO 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
HaykerDAO Audit	Rock, Hosea, Rushairer, Rico, David, Alice	1.0.0	2021-04-21

Audit results

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the HaykerDAO 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 3 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
./ds-proxy/src/proxy.sol	f220f9312d3b92cd3146c39500955317
./flipper-mom/src/DelayingFlipperMom.sol	3accc4c74109d12dcd939d46aac99632
./flipper-mom/src/FlipperMom.sol	31badfa78ebee3145b6c394b858acd0d
./dss-cdp-manager/src/GetCdps.sol	6c684a90f26618b6d3b6fa9c2238c7ad
./dss-cdp-manager/src/DssCdpManager.sol	d9370cbf0b86bcec12db0712502b2d8b
./ds-token/src/token.sol	2f2694b67d6c2b2e7f9fb855864e97a1
./ds-pause/src/pause.sol	8ea76d84c73cf9a69c834e7861d89244
./ds-note/src/note.sol	cff077720b1444b4e848afcabf67cc2d
./osm/src/value.sol	81e8ffc8b4ce7eeade00aaa3cc6d23ed
./osm/src/osm.sol	0f484672dbf032dcfbb49feb40bc55de
./esm/src/ESM.sol	5f6baf06aeb94086db07757428f6c716
./ds-roles/src/roles.sol	90c9beb562625cd4411cbbc420b593a1
./ds-math/src/math.sol	8c550fcfa8bd4f7feb87aaf59c230caf
./ds-chief/src/chief.sol	0e585c7f3b12366977ff5d259c9ad871
./ds-stop/src/stop.sol	6ab15ee97d2d16971c1eb621fbb73fa8
./dss-auto-line/src/DssAutoLine.sol	b49c929b196a217387141b08329ac0a8
./dss/src/dai.sol	bd2de4e5fb0a46b6a6d6ba737edd721f
./dss/src/jug.sol	f5729a5022cfca7b7f413866e5651ecf
./dss/src/flap.sol	63098e63084187a34022ce6b2b0d9ca3
./dss/src/spot.sol	c180036e00bb65782c4f98f45b43f134
./dss/src/join.sol	f8ef938661e0136d898ce779c92d36cd
./dss/src/vat.sol	f4e091ea31367956edcc750fb1f4031a
./dss/src/end.sol	1450699da8f8ac2ee0e193ae37b13090
./dss/src/lib.sol	67b2c0e39951a65a43d9123f49207f7d
./dss/src/vow.sol	6518a7a6f239899c83a94e36378701d5
./dss/src/pot.sol	0ef6fc97edf088a56da3657ad9d59bef
./dss/src/cat.sol	5f294285cb4c00499173cde40b909078
./dss/src/flip.sol	db342750530d16461b5a0341d7f4d09d
./dss/src/flop.sol	e0c38bacc70609d2c3d3ceb78622f74a
./osm-median-chainlink/src/osm-median-chainlink.sol	71a2a4c7a79709666d1cba1cf4e6e629

file	md5
./dss-proxy-actions/src/DssProxyActions.sol	5c4d69cf0daf5d0107b6b26fd02053a9
./ilk-registry/test/fixtures/UnDai.sol	8b2254affd8572850436f1bf6a8e5a5f
./ilk-registry/src/IlkRegistry.sol	fffda399ccc89bde5cb42a5c50ee2c8a
./ds-value/src/value.sol	9145f5e5bfde5e3524234c5c0bce94eb
./ds-auth/src/auth.sol	f30eceb4d4be1cec4e9054acda2c7b2f
./osm-mom/src/OsmMom.sol	84eb1b67a268e19341f4062117df0367
./proxy-registry/src/ProxyRegistry.sol	6d84aef5c9b50bd0d1bff336203910f8
./dsr-manager/src/DsrManager.sol	85a605b193260e6ad345ca550af55c6a
./ds-thing/src/thing.sol	b052eeca40898b9c09b644c21e6144e7
./vote-proxy/src/VoteProxy.sol	6310784a12c64bb1fe54c3799637ac04
./vote-proxy/src/VoteProxyFactory.sol	42ca4f966b66112d4eaf2935a9f69dd6
./mkr-authority/src/MkrAuthority.sol	d251452bcd5e61a9b09fc4640d9c3821
./mkr-authority/src/flap.sol	aaaea380061c478249fad3f7c35a108f
./mkr-authority/src/lib.sol	6aebec3a3cc826b6ed2df026ab6ef3e9
./mkr-authority/src/flop.sol	ceb4f2270419a97241ecddd34be1d5d0
./dss-gem-joins/src/join-8.sol	aa0a44be505324db14603ad652d192e6
./dss-gem-joins/src/join-4.sol	5b9c3984aca2b50515bbcd090a827e2f
./dss-gem-joins/src/join-5.sol	69940d2629e10d6f41251d27de6ae23c
./dss-gem-joins/src/join-7.sol	7adf1687d6e3ded254b58ca26bf2201f
./dss-gem-joins/src/join-6.sol	b4c0729fa41098b209c2a29fc35ef410
./dss-gem-joins/src/join-2.sol	cc98ee307b0346e124b226f5712ee600
./dss-gem-joins/src/join-3.sol	f74eceaf0bcc9a65e57756895babf235
./dss-gem-joins/src/join-auth.sol	7d9bb0941904a1e59de178f3e9911af9
./dss-gem-joins/src/tokens/RENBTC.sol	2471e82b8978004f7743169ddb3d6af8
./dss-gem-joins/src/tokens/DGD.sol	3903530640519fa69040d4c68a38a1cc
./dss-gem-joins/src/tokens/GUSD.sol	7d5cb35263471ed2598e88452d84b60e
./dss-gem-joins/src/tokens/KNC.sol	4aa6744a86ed4a9bb96a98851662b39e
./dss-gem-joins/src/tokens/COMP.sol	2c3bcf26476e1b9e7f33f2c3b805ef21
./dss-gem-joins/src/tokens/MANA.sol	9557c8558a057a5ad155c1283eba9429
./dss-gem-joins/src/tokens/BAT.sol	5d6d41d4718b07981f71cbda1b44bdd2
./dss-gem-joins/src/tokens/AAVE.sol	a28dc897d80285ff8b05239ecd49e6a9
./dss-gem-joins/src/tokens/REP.sol	289e6c9cc01746c7c0ce1b3f7d5e41a8

file	md5
./dss-gem-joins/src/tokens/UNI.sol	103e38be1df5908f5152d27186064ad7
./dss-gem-joins/src/tokens/USDT.sol	73ae737bab1d2419b12c2df5b1a2628f
./dss-gem-joins/src/tokens/USDC.sol	bb5385c93264d83b2eb05c199a363344
./dss-gem-joins/src/tokens/ZRX.sol	e81e9cdbe7a85a5a58d46c7e4580d497
./dss-gem-joins/src/tokens/BAL.sol	2eaa4108ed6d0bc5da360e48df1ca0cd
./dss-gem-joins/src/tokens/GNT.sol	8904ddce198dc17c9c1e2ebf7930ff03
./dss-gem-joins/src/tokens/LINK.sol	deab0e19a398e0e30f6d46461aca8fc8
./dss-gem-joins/src/tokens/OMG.sol	811e3d405a690812d02f35b3baac7781
./dss-gem-joins/src/tokens/WBTC.sol	18ba5cd3c5d19c1bd01e461c7b0c21d0
./dss-gem-joins/src/tokens/TUSD.sol	f3bc66790df162cecabacd3ca9bc0aba
./dss-gem-joins/src/tokens/LRC.sol	267bb686ca770a1a2c398e9e0bd82a5f
./dss-gem-joins/src/tokens/YFI.sol	774c93e18834bfbe2ae7103212e496f1
./dss-gem-joins/src/tokens/PAXUSD.sol	cc558c132dcd62b5086bda4e4a9e2d84
./token-faucet/src/RestrictedTokenFaucet.sol	1a751cefc97dd1fcb0f410961d02f8fe
./token-faucet/src/lib.sol	d8bd11c943d8c115786cb4039b8fc83b
./token-faucet/src/TokenFaucet.sol	e2490b5c20626385ef7b817f8762ff0c
./multicall/src/Multicall.sol	2fa6135ceebf777fa5ee1028a3ecc4be

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

Vulnerability	status
Delegatecall	safe
Default Visibilities	safe
Entropy Illusion	safe
External Contract Referencing	safe
Short Address/Parameter Attack	safe
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

```
.
├── ds-auth
│   └── src
│       └── auth.sol
├── ds-chief
│   └── src
│       └── chief.sol
├── ds-math
│   └── src
│       └── math.sol
├── ds-note
│   └── src
│       └── note.sol
├── ds-pause
│   └── src
│       └── pause.sol
├── ds-proxy
│   └── src
│       └── proxy.sol
├── ds-roles
│   └── src
│       └── roles.sol
├── ds-stop
│   └── src
│       └── stop.sol
├── ds-thing
│   └── src
│       └── thing.sol
├── ds-token
│   └── src
│       └── token.sol
├── ds-value
```

```
│   └── src
│       └── value.sol
├── dsr-manager
│   └── src
│       └── DsrManager.sol
├── dss
│   └── src
│       ├── cat.sol
│       ├── dai.sol
│       ├── end.sol
│       ├── flap.sol
│       ├── flip.sol
│       ├── flop.sol
│       ├── join.sol
│       ├── jug.sol
│       ├── lib.sol
│       ├── pot.sol
│       ├── spot.sol
│       ├── vat.sol
│       └── vow.sol
├── dss-auto-line
│   └── src
│       └── DssAutoLine.sol
├── dss-cdp-manager
│   └── src
│       ├── DssCdpManager.sol
│       └── GetCdps.sol
├── dss-gem-joins
│   └── src
│       ├── join-2.sol
│       ├── join-3.sol
│       ├── join-4.sol
│       ├── join-5.sol
│       ├── join-6.sol
│       ├── join-7.sol
│       ├── join-8.sol
│       ├── join-auth.sol
│       └── tokens
│           ├── AAVE.sol
│           ├── BAL.sol
│           ├── BAT.sol
│           ├── COMP.sol
│           ├── DGD.sol
│           ├── GNT.sol
│           ├── GUSD.sol
│           ├── KNC.sol
│           ├── LINK.sol
│           ├── LRC.sol
│           ├── MANA.sol
│           ├── OMG.sol
│           ├── PAXUSD.sol
│           ├── RENBTC.sol
│           ├── REP.sol
│           ├── TUSD.sol
│           ├── UNI.sol
│           ├── USDC.sol
│           ├── USDT.sol
│           ├── WBTC.sol
│           ├── YFI.sol
│           └── ZRX.sol
├── dss-proxy-actions
│   └── src
│       └── DssProxyActions.sol
├── esm
│   └── src
│       └── ESM.sol
```

```
├── flipper-mom
│   └── src
│       ├── DelayingFlipperMom.sol
│       └── FlipperMom.sol
├── ilk-registry
│   ├── src
│   │   └── IlkRegistry.sol
│   └── test
│       └── fixtures
│           └── UnDai.sol
├── mkr-authority
│   ├── src
│   │   ├── MkrAuthority.sol
│   │   ├── flap.sol
│   │   ├── flop.sol
│   │   └── lib.sol
│   └── test.sh
├── multicall
│   └── src
│       └── Multicall.sol
├── osm
│   └── src
│       ├── osm.sol
│       └── value.sol
├── osm-median-chainlink
│   └── src
│       └── osm-median-chainlink.sol
├── osm-mom
│   └── src
│       └── OsmMom.sol
├── proxy-registry
│   └── src
│       └── ProxyRegistry.sol
├── token-faucet
│   └── src
│       ├── RestrictedTokenFaucet.sol
│       ├── TokenFaucet.sol
│       └── lib.sol
└── vote-proxy
    └── src
        ├── VoteProxy.sol
        └── VoteProxyFactory.sol
```


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 vulernabilities 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