Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	MVHQ Token
Platform:	Ethereum
Language:	Solidity
Date:	April 27th, 2022

Table of contents

Email: audit@EtherAuthority.io

`

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the MVHQ team to perform the Security audit of the MVHQ Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on April 27th, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

The MVHQ is a NFT token smart contract which uses an upgradeable proxy contract, which means the MVHQ contract can be changed after the mainnet deployment. The MVHQ contract inherits the Initializable, AccessControlUpgradeable, StringsUpgradeable, ERC1155, IERC1155Receiver standard smart contracts from the OpenZeppelin library. These OpenZeppelin contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

Audit scope

Name	**Code Review and Security Analysis Report for MVHQ Token Smart Contract**
Platform	**Ethereum / Solidity**
File	MVHQ.sol
File MD5 Hash	C885E5B4D4AD580E767AF745FF8800A4
Online Code - Proxy	https://rinkeby.etherscan.io/address/0xECcc2594956AF12Ef783E3376a91A9191f7201D3#code
Online Code - Implementation	https://rinkeby.etherscan.io/address/0x2809a8737477a534df65c4b4cae43d0365e52035#code
Audit Date	April 27th, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
ERC1967Proxy.sol • ERC1967 Proxy standard • Forwards all read/write calls to implementation • Implementation contract can be changed by the owner of proxy	**YES, This is valid.** **This proxy contract allows the owner to change the main contract logic after mainnet deployment**
MVHQ.sol • Upgradeable Contracts • Token standard: ERC721A - upgradable • Token Name: MVHQ • Token Symbol: MVHQ • Whale Requirement: 5 Tokens • The owner can flag any keys and addresses	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. This token contract does contain owner control, which does not make it fully decentralized.

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 3 low and some very low level issues.
All these issues have been acknowledged.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 2 smart contract files. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in MVHQ Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the MVHQ Token.

The MVHQ Token team has provided scenario and unit test scripts, which have helped to determine the integrity of the code in an automated way.

Code parts are **well** commented on smart contracts.

Documentation

We were given a MVHQ Token smart contract code in the form of a File.The hash of that code is mentioned above in the table.

As mentioned above, code parts are **well** commented. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	Passed	No Issue
3	claimKeys	external	Passed	No Issue
4	isWhale	external	Passed	No Issue
5	isKeyFlagged	read	Infinite loop possibility	use limited array elements
6	getFlaggedKeys	external	Passed	No Issue
7	isAddressFlagged	read	Infinite loop possibility	use limited array elements
8	getFlaggedAddresses	read	Passed	No Issue
9	setWhaleRequirement	external	access by manager or a default admin	No Issue
10	flagAddress	external	access by manager or a default admin	No Issue
11	unflagAddress	external	Infinite loop possibility	use limited array elements
12	flagKey	external	access by manager or a default admin	No Issue
13	unflagKey	external	Infinite loop possibility	use limited array elements
14	adminTransfer	external	access only Role	No Issue
15	transferLegacyKeys	external	access only Role	No Issue
16	burn	write	access only Role	No Issue
17	setClaimActive	external	access only Role	No Issue
18	setBaseURI	external	access only Role	No Issue
19	withdraw	external	access only Role	No Issue
20	transferOwnership	external	access only Role	No Issue
21	_startTokenId	internal	Passed	No Issue
22	tokenURI	read	Passed	No Issue
23	supportsInterface	read	Passed	No Issue
24	_beforeTokenTransfers	internal	Infinite loop possibility	use limited array elements
25	onERC1155Received	external	Passed	No Issue

26	onERC1155BatchReceived	external	Passed	No Issue
27	managed	modifier	Passed	No Issue
28	initializer	modifier	Passed	No Issue
29	reinitializer	modifier	Passed	No Issue
30	onlyInitializing	modifier	Passed	No Issue
31	_disableInitializers	internal	Passed	No Issue
32	_setInitializedVersion	write	Passed	No Issue
33	__AccessControl_init	internal	Passed	No Issue
34	__AccessControl_init_unchained	internal	Passed	No Issue
35	onlyRole	modifier	Passed	No Issue
36	hasRole	read	Passed	No Issue
37	_checkRole	internal	Passed	No Issue
38	getRoleAdmin	read	Passed	No Issue
39	grantRole	write	Passed	No Issue
40	revokeRole	write	Passed	No Issue
41	renounceRole	write	Passed	No Issue
42	_setupRole	internal	Passed	No Issue
43	_setRoleAdmin	internal	Passed	No Issue
44	_grantRole	internal	Passed	No Issue
45	_revokeRole	internal	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) High gas consuming loops

```
function unflagAddress(address address_) external managed {
    for (uint256 i = 0; i < flaggedAddresses.length; i++) {
        if (flaggedAddresses[i] == address_) {
            flaggedAddresses[i] = flaggedAddresses[flaggedAddresses.length - 1];
            flaggedAddresses.pop();
            break;
        }
    }
    emit AddressUnflagged(msg.sender, address_);
```

If the flaggedAddresses array size becomes larger, then it will cost more and more gas to to flag/unflag the wallets. In case this array length becomes so large that it will hit the block's limit and stop this admin function. This is also true for the flaggedKeys array.

Resolution: We suggest adjusting the logic such that it avoids using loops to add/remove elements from the array. For example, if you can store the array index in the mapping, then the array index for a particular wallet can be obtained from the mapping and there would not be any need for the "loop" to remove the element from this array.

Status: Acknowledged

(2) Owner can move anyone's tokens

```
function adminTransfer(
    address from_,
    address to_,
    uint256 tokenId_
) external onlyRole(DEFAULT_ADMIN_ROLE) {
    _adminTransferFrom(from_, to_, tokenId_);
    emit AdminTransfer(msg.sender, from_, to_, tokenId_);
}
```

Using the function adminTransfer, the owner can transfer any user's tokens. If this is a desired feature, then this point can be ignored. But this may create fear in users that their tokens can be taken away anytime, if the owner's wallet private key would be compromised. The same thing with the "Burn" function. The owner can burn any user's tokens.

Resolution: If this is not a required feature, then we suggest removing it.

Status: Acknowledged

(3) Missing event logs

It is recommended to fire an event where the significant state of the smart contract is being changed. This helps clients (UI) to interact with the blockchain. Following function can have events:

- Withdraw

Resolution: Add an event in above function

Status: Acknowledged

Very Low / Informational / Best practices:

(1) Redundant withdraw function

```
function withdraw() external payable onlyRole(DEFAULT_ADMIN_ROLE) {
    // solhint-disable-next-line avoid-low-level-calls
    (bool success, ) = payable(msg.sender).call{value: address(this).balance}("");
    if (!success) revert FailedToWithdraw();
}
```

The contract does not have a fallback or receive function. So, there is no way the ether can enter into the smart contract. Therefore the withdraw function will never be used and thus it becomes redundant.

Resolution: We suggest removing it, if that does not serve any purpose.

Status: Acknowledged

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- setWhaleRequirement: An admin / manager can be used to set the whale requirement values.
- flagAddress: It is used to flag an address and remove the ability for it to transfer keys callable by admin or manager.
- unflagAddress: It is used to remove the flag of an address and restore the ability for it to transfer keys callable by admin or manager.
- flagKey: It is used to flag a key and make it non transferrable callable by admin or manager.
- unflagKey: It is used to remove the flag of a key and restore the ability for it to be transferable callable by the admin or manager.
- adminTransfer: Admin can transfer a token from one address to another and is meant to be used with extreme care only callable from an address with the admin role.
- transferLegacyKeys: Admin can be used to transfer legacy keys to an address.
- burn: Admin will burn the keys minted from this address.
- setClaimActive: Admin can toggle the claiming functionality.
- setBaseURI: Admin can set the baseURI for metadata.
- withdraw: Admin can withdraw amount in case anyone sends ETH to contract by mistake.
- transferOwnership: Admin can be used to set a new owner value.
- _authorizeUpgrade: Admin can UUPS Upgradeable authorization.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We have observed some issues and they are acknowledged by the MVHQ team. So, **it's good to go to production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

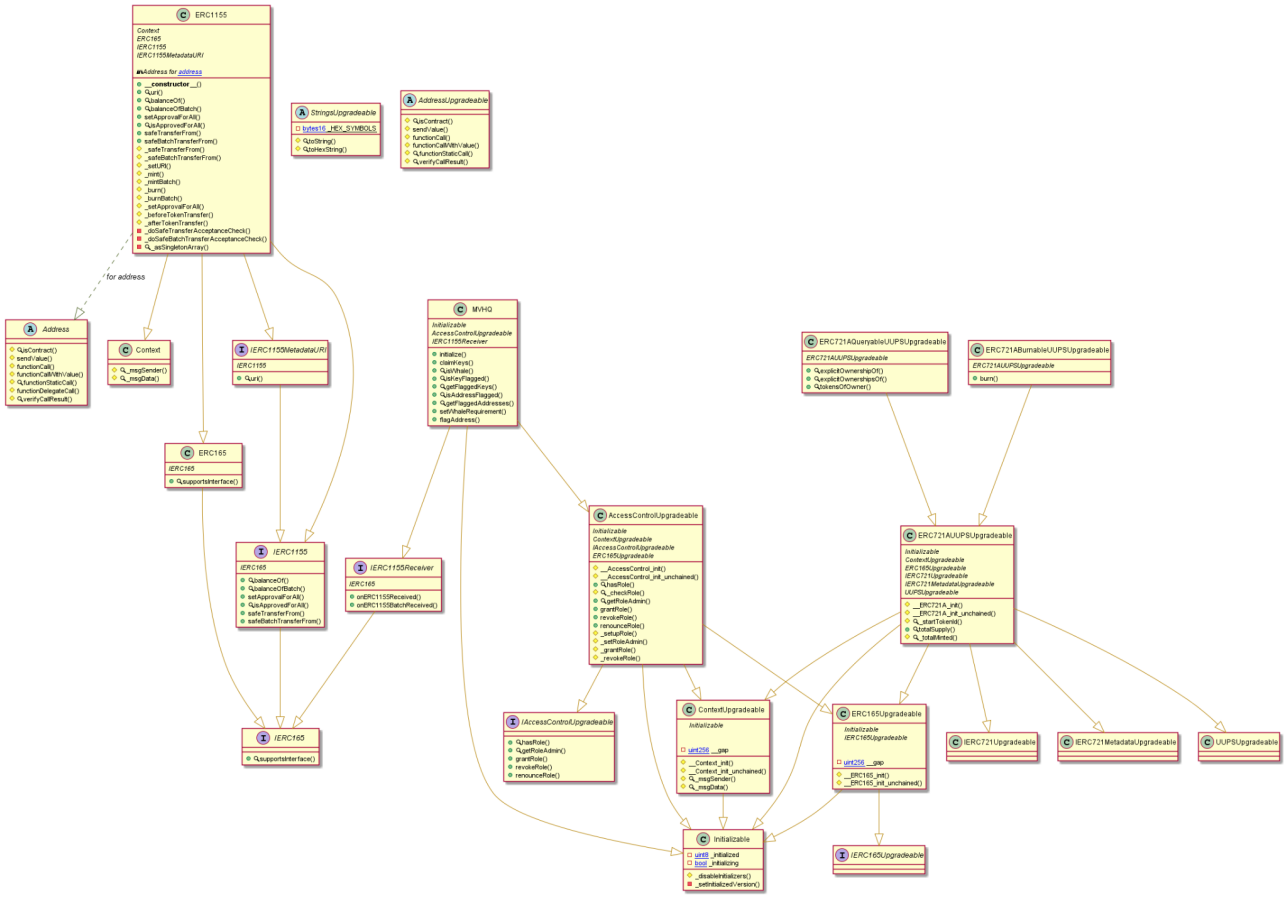
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - MVHQ Token



Slither Results Log

Slither log >> MVHQ.sol

```
INFO:Detectors:
MVHQ.transferOwnership(address).newOwner_ (MVHQ.sol#902) lacks a zero-check on :
                - owner = newOwner_ (MVHQ.sol#904)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (MVHQ.sol#713)' in ER
C1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (MVHQ.sol#704-723) potentially used before
 declaration: response != IERC1155Receiver.onERC1155Received.selector (MVHQ.sol#714)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (MVHQ.sol#717)' in ERC1
155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (MVHQ.sol#704-723) potentially used before d
eclaration: revert(string)(reason) (MVHQ.sol#718)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (MVHQ.sol#73
5)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (MVHQ.sol#725-746) poten
tially used before declaration: response != IERC1155Receiver.onERC1155BatchReceived.selector (MVHQ.sol#737)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (MVHQ.sol#740)
' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (MVHQ.sol#725-746) potenti
ally used before declaration: revert(string)(reason) (MVHQ.sol#741)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in MVHQ.claimKeys() (MVHQ.sol#794-810):
        External calls:
        - OSSF.safeTransferFrom(msg.sender,address(this),OSMVHQ_TOKENID,claimable,bytes(0x0)) (MVHQ.sol#807)
        Event emitted after the call(s):
        - KeysClaimed(msg.sender,claimable) (MVHQ.sol#809)
Reentrancy in MVHQ.transferLegacyKeys(address) (MVHQ.sol#879-884):
        External calls:
        - OSSF.safeTransferFrom(address(this),to_,OSMVHQ_TOKENID,balance,bytes(0x0)) (MVHQ.sol#882)
        Event emitted after the call(s):
        - LegacyKeysTransferred(msg.sender,to_,balance) (MVHQ.sol#883)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (MVHQ.sol#72-90) uses assembly
        - INLINE ASM (MVHQ.sol#82-85)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (MVHQ.sol#281-299) uses assembly
        - INLINE ASM (MVHQ.sol#291-294)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
AccessControlUpgradeable.__AccessControl_init_unchained() (MVHQ.sol#383-384) is never used and should be removed
AccessControlUpgradeable._setRoleAdmin(bytes32,bytes32) (MVHQ.sol#435-439) is never used and should be removed
AccessControlUpgradeable._setupRole(bytes32,address) (MVHQ.sol#432-434) is never used and should be removed
Address.functionCall(address,bytes) (MVHQ.sol#17-19) is never used and should be removed
Address.functionCall(address,bytes,string) (MVHQ.sol#20-26) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MVHQ.sol#27-33) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (MVHQ.sol#34-45) is never used and should be removed
Address.functionDelegateCall(address,bytes) (MVHQ.sol#59-61) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (MVHQ.sol#62-71) is never used and should be removed
Address.functionStaticCall(address,bytes) (MVHQ.sol#46-48) is never used and should be removed
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (MVHQ.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MVHQ.sol#11-16):
        - (success) = recipient.call{value: amount}() (MVHQ.sol#14)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (MVHQ.sol#34-45):
        - (success,returndata) = target.call{value: value}(data) (MVHQ.sol#43)
Low level call in Address.functionStaticCall(address,bytes,string) (MVHQ.sol#49-58):
        - (success,returndata) = target.staticcall(data) (MVHQ.sol#56)
Low level call in Address.functionDelegateCall(address,bytes,string) (MVHQ.sol#62-71):
        - (success,returndata) = target.delegatecall(data) (MVHQ.sol#69)
Low level call in AddressUpgradeable.sendValue(address,uint256) (MVHQ.sol#233-238):
        - (success) = recipient.call{value: amount}() (MVHQ.sol#236)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (MVHQ.sol#256-267):
        - (success,returndata) = target.call{value: value}(data) (MVHQ.sol#265)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (MVHQ.sol#271-280):
        - (success,returndata) = target.staticcall(data) (MVHQ.sol#278)
Low level call in MVHQ.withdraw() (MVHQ.sol#898-901):
        - (success) = address(msg.sender).call{value: address(this).balance}() (MVHQ.sol#899)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function ContextUpgradeable.__Context_init() (MVHQ.sol#353-354) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (MVHQ.sol#356-357) is not in mixedCase
Variable ContextUpgradeable.__gap (MVHQ.sol#365) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (MVHQ.sol#370-371) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (MVHQ.sol#373-374) is not in mixedCase
Variable ERC165Upgradeable.__gap (MVHQ.sol#376) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init() (MVHQ.sol#380-381) is not in mixedCase
```

```
Variable ERC165Upgradeable.__gap (MVHQ.sol#376) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init() (MVHQ.sol#380-381) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init_unchained() (MVHQ.sol#383-384) is not in mixedCase
Variable AccessControlUpgradeable.__gap (MVHQ.sol#452) is not in mixedCase
Parameter MVHQ.setClaimActive(bool)._claimActive (MVHQ.sol#889) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
MVHQ (MVHQ.sol#756-918) does not implement functions:
        - IERC1155Receiver.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) (MVHQ.sol#121-127)
        - IERC1155Receiver.onERC1155Received(address,address,uint256,uint256,bytes) (MVHQ.sol#114-120)
        - IERC165.supportsInterface(bytes4) (MVHQ.sol#104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
```


Email: audit@EtherAuthority.io

```
INFO:Detectors:
AccessControlUpgradeable.__gap (MVHQ.sol#452) is never used in MVHQ (MVHQ.sol#756-918)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
supportsInterface(bytes4) should be declared external:
        - ERC165.supportsInterface(bytes4) (MVHQ.sol#108-110)
grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (MVHQ.sol#421-423)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (MVHQ.sol#424-426)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (MVHQ.sol#427-431)
uri(uint256) should be declared external:
        - ERC1155.uri(uint256) (MVHQ.sol#465-467)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155.balanceOfBatch(address[],uint256[]) (MVHQ.sol#472-488)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155.setApprovalForAll(address,bool) (MVHQ.sol#489-491)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (MVHQ.sol#495-507)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (MVHQ.sol#508-520)
initialize(string) should be declared external:
        - MVHQ.initialize(string) (MVHQ.sol#787-793)
isKeyFlagged(uint256) should be declared external:
        - MVHQ.isKeyFlagged(uint256) (MVHQ.sol#813-818)
isAddressFlagged(address) should be declared external:
        - MVHQ.isAddressFlagged(address) (MVHQ.sol#823-828)
burn(uint256) should be declared external:
        - MVHQ.burn(uint256) (MVHQ.sol#886-888)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MVHQ.sol analyzed (17 contracts with 75 detectors), 88 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

MVHQ.sol

Security

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: 124:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MVHQ.transferLegacyKeys(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1795:4:

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: 1828:27:

Gas & Economy

Gas costs:

Gas requirement of function ERC1155.safeBatchTransferFrom is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 1207:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 1695:8:

Email: audit@EtherAuthority.io

Miscellaneous

Constant/View/Pure functions:

MVHQ._authorizeUpgrade(address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1872:4:

Similar variable names:

MVHQ.setBaseURI(string) : Variables have very similar names "baseURI" and "baseURI_". Note: Modifiers are currently not considered by this static analysis.
Pos: 1822:53:

No return:

IAccessControlUpgradeable.getRoleAdmin(bytes32): Defines a return type but never explicitly returns a value.
Pos: 533:4:

No return:

MVHQ.isWhale(address): Defines a return type but never explicitly returns a value.
Pos: 1688:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1450:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1471:8:

Solhint Linter

MVHQ.sol

```
MVHQ.sol:1250:18: Error: Parse error: missing ';' at '{'
MVHQ.sol:1292:22: Error: Parse error: missing ';' at '{'
MVHQ.sol:1417:18: Error: Parse error: missing ';' at '{'
MVHQ.sol:1451:22: Error: Parse error: missing ';' at '{'
MVHQ.sol:1586:6: Error: Parse error: missing 'constant' at
'ClaimingNotActive'
MVHQ.sol:1586:23: Error: Parse error: missing '=' at '('
MVHQ.sol:1587:6: Error: Parse error: missing 'constant' at
'ApprovalRequired'
MVHQ.sol:1587:22: Error: Parse error: missing '=' at '('
MVHQ.sol:1588:6: Error: Parse error: missing 'constant' at
'NoClaimableKeys'
MVHQ.sol:1588:21: Error: Parse error: missing '=' at '('
MVHQ.sol:1589:6: Error: Parse error: missing 'constant' at
'NoLegacyKeysToTransfer'
MVHQ.sol:1589:28: Error: Parse error: missing '=' at '('
MVHQ.sol:1590:6: Error: Parse error: missing 'constant' at
'FailedToWithdraw'
MVHQ.sol:1590:22: Error: Parse error: missing '=' at '('
MVHQ.sol:1591:6: Error: Parse error: missing 'constant' at
'FromFlaggedAddress'
MVHQ.sol:1591:24: Error: Parse error: missing '=' at '('
MVHQ.sol:1592:6: Error: Parse error: missing 'constant' at
'ToFlaggedAddress'
MVHQ.sol:1592:22: Error: Parse error: missing '=' at '('
MVHQ.sol:1593:6: Error: Parse error: missing 'constant' at
'KeyIsFlagged'
MVHQ.sol:1593:18: Error: Parse error: missing '=' at '('
MVHQ.sol:1594:6: Error: Parse error: missing 'constant' at
'Unauthorized'
MVHQ.sol:1594:18: Error: Parse error: missing '=' at '('
MVHQ.sol:1664:50: Error: Parse error: mismatched input '(' expecting
{';', '='}
MVHQ.sol:1665:86: Error: Parse error: mismatched input '(' expecting
{';', '='}
MVHQ.sol:1668:50: Error: Parse error: mismatched input '(' expecting
{';', '='}
MVHQ.sol:1797:55: Error: Parse error: mismatched input '(' expecting
{';', '='}
MVHQ.sol:1829:45: Error: Parse error: mismatched input '(' expecting
{';', '='}
MVHQ.sol:1885:111: Error: Parse error: mismatched input '(' expecting
{';', '='}
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io