QuillAudits

Audit Report
May, 2022

For

CYBER CITY

Table of Content

Executive Summary

Project Name	CyberCity
Timeline	May 10th, 2022 to May 19th, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse CyberCity codebase for quality, security, and correctness.
Git Repo link	*https://github.com/cybercity-official/cyber-city-back/blob/main/contracts/Auction.sol* *https://github.com/cybercity-official/cyber-city-back/blob/main/contracts/AuctionFactory.sol*
Git Branch	Main
Commit Hash	0f34b6bc4c2bd8eef0b7e8a86b829961712aff7e
Fixed In	d03b4c3253c530ef81656d0b81142d97d84fc7f5

High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	1
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	1	5	4

Types of Severities

High

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 be fixed before moving to a live environment.

Medium

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

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 issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, 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 analysed 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.

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 the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the 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 the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract - Auction.sol

High Severity Issues

No issues found

Medium Severity Issues

1. Uncheck transfer

Description

The return value of an external transfer/transferFrom call is not checked since the external tokens do not revert in case of failure and return false. We've found the following return values are not checked.

- L192: payTokenInstance.transferFrom(_msgSender(), address(this), needAmount);
- L208: payTokenInstance.transferFrom(_msgSender(), address(comissionWallet), _comission);
- L209: payTokenInstance.transferFrom(_msgSender(), address(owner), _remain);
- L227: erc721Instance.transferFrom(address(this), owner, tokenId);
- L268: payTokenInstance.transfer(address(comissionWallet), _comission);
- L269: payTokenInstance.transfer(owner, _remain);
- L282: payTokenInstance.transfer(_msgSender(), withdrawalAmount);

Remediation

Please consider adding the require() check for those external calls or using SafeERC20, or ensure that the transfer/transferFrom return value is checked.

Status

Fixed

Low Severity Issues

2. Lack of event emissions

Description

The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:
- setTimestampAdmin()

Recommendation

We recommend emitting an event to log the update of the above variables for the above-mentioned function.

Status

Fixed

3. Missing check for comissionPercent

Line #70

```
70      constructor(
71          address _owner,
72          address _admin,
73          uint256 _duration,
74          address nftToken,
75          uint256 nftId,
76          address _payTokenAddress,
77          uint256 _buyValue,
78          Type _auctionType,
79          address _comissionWallet,
80          uint256 _comissionPercent
81      ) {
82          owner = _owner;
83          admin = _admin;
84          erc721Instance = IERC721(nftToken);
85          tokenId = nftId;
86          duration = _duration;
87          payTokenInstance = IERC20(_payTokenAddress);
88          buyValue = _buyValue;
89          currentBid = _buyValue;
90          auctionType = _auctionType;
91          auctionState = States.Initialize;
92          comissionPercent = _comissionPercent;
93          comissionWallet = _comissionWallet;
94      }
```

Description

While calculating the transaction fee, there is no check for the comissionPercent that could be easily set to more than 100.

Recommendation

To solve the issue, a check should be placed in the function that makes sures that the comissionPercent is always less than 100.

Status

Fixed

Informational Issues

4. Unused variables

Description

It was discovered that the _operator and _data variables in the onERC721Received() function are not used.

Unused code is allowed in Solidity, and they do not pose a direct security issue. It is best practice, though, to avoid them as they can:

- cause an increase in computations (and unnecessary gas consumption)
- indicate bugs or malformed data structures, and they are generally a sign of poor code quality
- cause code noise and decrease the readability of the code

Recommendation

We recommend removing all unused variables/code from the codebase.

Status

Acknowledged

5. Public function that could be declared external

Description

The following public functions that are never called by the contract should be declared external to save gas:

- placeBid()
- withdraw()

Recommendation

Use the external attribute for functions never called from the contract.

Status

Fixed

6. Typos

Description

We recommend fixing the following typos:
- comissionPercent should be commissionPercent
- comissionWallet should be commissionWallet

Status

Fixed

B. Contract - AuctionFactory.sol

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

1. Lack of event emissions

Description

The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:
- changeAdmin()
- changeComissionPercent()
- changeComissionWallet()

Recommendation

We recommend emitting an event to log the update of the above variables for the above-mentioned function.

Status

Fixed

2. Missing check for comissionPercent

Line #36

```
function changeComissionPercent(uint256 _newPercent) external onlyAdmin {
    comissionPercent = _newPercent;
}
```

Description

While calculating the transaction fee, there is no check for the comissionPercent that could be easily set to more than 100.

Recommendation

To solve the issue, a check should be placed in the function that makes sures that the comissionPercent is always less than 100.

Status

Fixed

3. Lack of Zero address validation

Description

To favor explicitness, consider adding a check for all functions that are taking address parameters in the entire codebase. Although most of the functions throughout the codebase properly validate function inputs, there are some instances of functions that do not. One example is:

- changeAdmin()
- changeComissionWallet()

Remediation

Consider implementing require statements where appropriate to validate all user-controlled input, including governance functions, to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

Status

Fixed

Informational Issues

4. Unused variables

Description

We recommend fixing the following typos:
- changeComissionPercent should be changeCommissionPercent
- changeComissionWallet should be changeCommissionWallet
- comissionPercent should be commissionPercent
- comissionWallet should be commissionWallet

Status

Fixed

5. Public function that could be declared external

Description

The following public functions that are never called by the contract should be declared external to save gas:

- addPayToken()
- removePayToken()
- createAuction()

Recommendation

Use the external attribute for functions never called from the contract.

Status

Fixed

Functional Testing

Contract - Auction.sol

- ✓ getInfo should return all data
- ✓ isVisible should return status of the caller
- ✓ getTime should return the current timestamp
- ✓ setTimestampAdmin should be called only by the admin
- ✓ getNeededAllowancePaytoken should return the sub of currentBid and FunsByBidder
- ✓ placeBid(should be called onlyStarted
- ✓ placeBid should be called onlyBeforeEnd
- ✓ placeBid should be called by anyone except the owner
- ✓ call placeBid when auctionType == Type.Auction
- ✓ call placeBid when auctionType != Type.Auction
- ✓ cancelAuction can be called only by the Owner
- ✓ cancelAuction can be called when not cancelled
- ✓ cancelAuction can be called only before start or only trade
- ✓ withdraw can be called when not cancelled
- ✓ withdraw can be called only ended time
- ✓ withdraw called by the owner
- ✓ withdraw called by the highestBidder
- ✓ withdraw called by who did not win the auction
- ✓ onERC721Received should revert if states != Initialize
- ✓ onERC721Received should revert if _from == address(0)
- ✓ onERC721Receivedshould revert if _msgSender() == address(erc721Instance) && _tokenId == tokenId
- ✓ onERC721Receivedshould revert if erc721Instance.ownerOf(tokenId) == address(this)

Contract - AuctionFactory.sol

- changeAdmin should be called only by the admin
- changeComissionPercent should be called only by the admin
- changeComissionWallet should be called only by the admin
- pause should be called only by the admin
- addAcceptableNft should be called only by the admin
- removeAcceptableNft should be called only by the admin
- getAcceptableNfts should return a correct value
- addPayToken should be called only by the admin
- removePayToken should revert if the is not payToken
- removePayToken should called only by the admin
- getPayTokens should return correct values
- createAuction should revert when paused
- createAuction should revert when payToken is False
- createAuction should revert when nftToken is not accepted
- allAuctions should return correct values

Automated Tests

Auction.sol

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Slither

```
INFO:Detectors:
Auction.startTimestamp (Auction.sol#43) is never initialized. It is used in:
        - Auction.getInfo() (Auction.sol#96-120)
Auction.endTimestamp (Auction.sol#44) is never initialized. It is used in:
        - Auction.getInfo() (Auction.sol#96-120)
Auction.highestBindingBid (Auction.sol#54) is never initialized. It is used in:
        - Auction.getInfo() (Auction.sol#96-120)
Auction.highestBidder (Auction.sol#55) is never initialized. It is used in:
        - Auction.getInfo() (Auction.sol#96-120)
Auction.erc721present (Auction.sol#60) is never initialized. It is used in:
        - Auction.getInfo() (Auction.sol#96-120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Auction.getInfo().data (Auction.sol#97) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Auction.constructor(address,address,uint256,address,uint256,address,uint256,Type,address,uint256)._owner (Auction.sol#71) lacks a zero-check on :
                - owner = _owner (Auction.sol#82)
Auction.constructor(address,address,uint256,address,uint256,address,uint256,Type,address,uint256)._admin (Auction.sol#72) lacks a zero-check on :
                - admin = _admin (Auction.sol#83)
Auction.constructor(address,address,uint256,address,uint256,address,uint256,Type,address,uint256)._comissionWallet (Auction.sol#79) lacks a zero-check on :
                - comissionWallet = _comissionWallet (Auction.sol#93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
Auction.endTimestamp (Auction.sol#44) should be constant
Auction.erc721present (Auction.sol#60) should be constant
Auction.highestBidder (Auction.sol#55) should be constant
Auction.highestBindingBid (Auction.sol#54) should be constant
Auction.startTimestamp (Auction.sol#43) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
getInfo() should be declared external:
        - Auction.getInfo() (Auction.sol#96-120)
isVisible() should be declared external:
        - Auction.isVisible() (Auction.sol#152-154)
getTime() should be declared external:
        - Auction.getTime() (Auction.sol#156-158)
setTimestampAdmin(uint256) should be declared external:
        - Auction.setTimestampAdmin(uint256) (Auction.sol#160-162)
placeBid() should be declared external:
        - Auction.placeBid() (Auction.sol#177-217)
withdraw() should be declared external:
        - Auction.withdraw() (Auction.sol#248-289)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/security/ReentrancyGuard.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0
.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0
.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/math/SafeMath.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.
7.6
Pragma version0.8.9 (Auction.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Auction (Auction.sol#11-358) does not implement functions:
        - Auction._calculateFee(uint256) (Auction.sol#232-246)
        - Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23)
        - Context._msgSender() (@openzeppelin/contracts/utils/Context.sol#17-19)
        - Auction.cancelAuction() (Auction.sol#219-230)
        - Auction.checkVisible(address) (Auction.sol#122-150)
        - Auction.getNeededAllowancePaytoken() (Auction.sol#173-175)
        - Auction.getTime() (Auction.sol#156-158)
        - Auction.isVisible() (Auction.sol#152-154)
        - Auction.onERC721Received(address,address,uint256,bytes) (Auction.sol#291-313)
        - Auction.placeBid() (Auction.sol#177-217)
        - Auction.setNewCurrentBid() (Auction.sol#168-171)
        - Auction.setTimestampAdmin(uint256) (Auction.sol#160-162)
        - Auction.withdraw() (Auction.sol#248-289)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
```

```
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - ^0.8.0 (@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/math/SafeMath.sol#4)
        - 0.8.9 (Auction.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Auction._calculateFee(uint256) (Auction.sol#232-246) is never used and should be removed
Auction.checkVisible(address) (Auction.sol#122-150) is never used and should be removed
Auction.setNewCurrentBid() (Auction.sol#160-171) is never used and should be removed
Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Context._msgSender() (@openzeppelin/contracts/utils/Context.sol#17-19) is never used and should be removed
SafeMath.add(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#93-95) is never used and should be removed
SafeMath.div(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#135-137) is never used and should be removed
SafeMath.div(uint256,uint256,string) (@openzeppelin/contracts/utils/math/SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (@openzeppelin/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed
SafeMath.mul(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#121-123) is never used and should be removed
SafeMath.sub(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#107-109) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (@openzeppelin/contracts/utils/math/SafeMath.sol#168-177) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

Solhint

```
Auction.sol
   1:1   error    Compiler version 0.8.9 does not satisfy the ^0.5.8 semver requirement                          compiler-version
  11:1   warning  Contract has 19 states declarations but allowed no more than 15                               max-states-count
  70:5   warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
 144:24  warning  Avoid to make time-based decisions in your business logic                                     not-rely-on-time
 167:16  warning  Avoid to make time-based decisions in your business logic                                     not-rely-on-time
 186:65  warning  Avoid to make time-based decisions in your business logic                                     not-rely-on-time
 208:23  warning  Avoid to make time-based decisions in your business logic                                     not-rely-on-time
 212:19  warning  Avoid to make time-based decisions in your business logic                                     not-rely-on-time
 275:13  warning  Possible reentrancy vulnerabilities. Avoid state changes after transfer                       reentrancy
 284:13  warning  Possible reentrancy vulnerabilities. Avoid state changes after transfer                       reentrancy
 292:9   warning  Variable "_operator" is unused                                                               no-unused-vars
 295:9   warning  Variable "_data" is unused                                                                   no-unused-vars
 307:26  warning  Avoid to make time-based decisions in your business logic                                     not-rely-on-time
 336:17  warning  Avoid to make time-based decisions in your business logic                                     not-rely-on-time
 341:17  warning  Avoid to make time-based decisions in your business logic                                     not-rely-on-time
```

AuctionFactory.sol

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Slither

```
INFO:Detectors:
Auction.startTimestamp (Auction.sol#43) is never initialized. It is used in:
        - Auction.getInfo() (Auction.sol#96-120)
Auction.endTimestamp (Auction.sol#44) is never initialized. It is used in:
        - Auction.getInfo() (Auction.sol#96-120)
Auction.highestBindingBid (Auction.sol#54) is never initialized. It is used in:
        - Auction.getInfo() (Auction.sol#96-120)
Auction.highestBidder (Auction.sol#55) is never initialized. It is used in:
        - Auction.getInfo() (Auction.sol#96-120)
Auction.erc721present (Auction.sol#60) is never initialized. It is used in:
        - Auction.getInfo() (Auction.sol#96-120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Auction.getInfo().data (Auction.sol#97) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Auction.constructor(address,address,uint256,address,uint256,address,uint256,Type,address,uint256)._owner (Auction.sol#71) lacks a zero-check on :
                - owner = _owner (Auction.sol#82)
Auction.constructor(address,address,uint256,address,uint256,address,uint256,Type,address,uint256)._admin (Auction.sol#72) lacks a zero-check on :
                - admin = _admin (Auction.sol#83)
Auction.constructor(address,address,uint256,address,uint256,address,uint256,Type,address,uint256)._comissionWallet (Auction.sol#79) lacks a zero-check on :
                - comissionWallet = _comissionWallet (Auction.sol#93)
AuctionFactory.constructor(address,address[],address[],address,uint256)._admin (AuctionFactory.sol#19) lacks a zero-check on :
                - addrAdmin = _admin (AuctionFactory.sol#25)
AuctionFactory.constructor(address,address[],address[],address,uint256)._comissionWallet (AuctionFactory.sol#22) lacks a zero-check on :
                - comissionWallet = _comissionWallet (AuctionFactory.sol#29)
AuctionFactory.changeAdmin(address).newAdminAddress (AuctionFactory.sol#32) lacks a zero-check on :
                - addrAdmin = newAdminAddress (AuctionFactory.sol#33)
AuctionFactory.changeComissionWallet(address)._newWallet (AuctionFactory.sol#40) lacks a zero-check on :
                - comissionWallet = _newWallet (AuctionFactory.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
getInfo() should be declared external:
        - Auction.getInfo() (Auction.sol#96-120)
isVisible() should be declared external:
        - Auction.isVisible() (Auction.sol#152-154)
getTime() should be declared external:
        - Auction.getTime() (Auction.sol#156-158)
setTimestampAdmin(uint256) should be declared external:
        - Auction.setTimestampAdmin(uint256) (Auction.sol#160-162)
placeBid() should be declared external:
        - Auction.placeBid() (Auction.sol#177-217)
withdraw() should be declared external:
        - Auction.withdraw() (Auction.sol#248-289)
addPayToken(address) should be declared external:
        - AuctionFactory.addPayToken(address) (AuctionFactory.sol#83-87)
removePayToken(address) should be declared external:
        - AuctionFactory.removePayToken(address) (AuctionFactory.sol#89-98)
createAuction(uint256,uint256,address,address,uint256,Type) should be declared external:
        - AuctionFactory.createAuction(uint256,uint256,address,address,uint256,Type) (AuctionFactory.sol#114-144)
allAuctions() should be declared external:
        - AuctionFactory.allAuctions() (AuctionFactory.sol#146-148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
Auction (Auction.sol#11-358) does not implement functions:
        - Auction._calculateFee(uint256) (Auction.sol#232-246)
        - Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23)
        - Context._msgSender() (@openzeppelin/contracts/utils/Context.sol#17-19)
        - Auction.cancelAuction() (Auction.sol#219-230)
        - Auction.checkVisible(address) (Auction.sol#122-150)
        - Auction.getNeededAllowancePaytoken() (Auction.sol#173-175)
        - Auction.getTime() (Auction.sol#156-158)
        - Auction.isVisible() (Auction.sol#152-154)
        - Auction.onERC721Received(address,address,uint256,bytes) (Auction.sol#291-313)
        - Auction.placeBid() (Auction.sol#177-217)
        - Auction.setNewCurrentBid() (Auction.sol#168-171)
        - Auction.setTimestampAdmin(uint256) (Auction.sol#160-162)
        - Auction.withdraw() (Auction.sol#248-289)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
Auction.endTimestamp (Auction.sol#44) should be constant
Auction.erc721present (Auction.sol#60) should be constant
Auction.highestBidder (Auction.sol#55) should be constant
Auction.highestBindingBid (Auction.sol#54) should be constant
Auction.startTimestamp (Auction.sol#43) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

```
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/security/Pausable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6
Pragma version^0.8.0 (@openzeppelin/contracts/security/ReentrancyGuard.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0
.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0
.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/math/SafeMath.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.
7.6
Pragma version0.8.9 (Auction.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (AuctionFactory.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter AuctionFactory.changeComissionPercent(uint256)._newPercent (AuctionFactory.sol#36) is not in mixedCase
Parameter AuctionFactory.changeComissionWallet(address)._newWallet (AuctionFactory.sol#40) is not in mixedCase
Parameter AuctionFactory.createAuction(uint256,uint256,address,address,uint256,Type)._type (AuctionFactory.sol#120) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - ^0.8.0 (@openzeppelin/contracts/security/Pausable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/math/SafeMath.sol#4)
        - 0.8.9 (Auction.sol#1)
        - 0.8.9 (AuctionFactory.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Auction._calculateFee(uint256) (Auction.sol#232-246) is never used and should be removed
Auction.checkVisible(address) (Auction.sol#122-150) is never used and should be removed
Auction.setNewCurrentBid() (Auction.sol#168-171) is never used and should be removed
Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
SafeMath.add(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#93-95) is never used and should be removed
SafeMath.div(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#135-137) is never used and should be removed
SafeMath.div(uint256,uint256,string) (@openzeppelin/contracts/utils/math/SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (@openzeppelin/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed
SafeMath.mul(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#121-123) is never used and should be removed
SafeMath.sub(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#107-109) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (@openzeppelin/contracts/utils/math/SafeMath.sol#168-177) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

Solhint

```
AuctionFactory.sol
   1:1  error    Compiler version 0.8.9 does not satisfy the ^0.5.8 semver requirement                compiler-version
   3:8  error    Use double quotes for string literals                                                quotes
  18:5  warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility

✖ 3 problems (2 errors, 1 warning)
```

Closing Summary

In this report, we have considered the security of the CyberCity smart contracts. We performed our audit according to the procedure described above.

Some issues of Medium, Low and informational severity were found, some suggestions and best practices are also provided in order to improve the code quality and security posture.

In the End,CyberCity Team Resolved all issues

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the CyberCity 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 CyberCity Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

Follow Our Journey

Audit Report
May, 2022

For

CYBER CITY

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com