AudiTBlock

PreciousHour

v0.8.18+commit.c7e474f2

Low-Risk	Medium-Risk	High-Risk
low-risk code	medium-risk code	high-risk code

Disclaimer

AudiTBlock is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

AudiTBlock is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your ownresearch.

The information provided in this audit is for informational purposes only and should not be considered investment advice. we does not endorse, recommend, support or suggest to invest in any project.

AudiTBlock can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Tokenomics

Source Code

- **AudiTBlock** was complete audit phases to perform an audit based on the following smart contract

Gas Cost

```
PreciousHour.getCurrentWeek() (contracts/PreciousHour.sol#112-116) uses a
 weak PRNG: "currentWeek = (elapsedTime / WEEK_IN_SECONDS) %
 TOTAL_WEEKS + 1 (contracts/PreciousHour.sol#114)"

PreciousHour.getCurrentOffset() (contracts/PreciousHour.sol#118-122) uses
 a weak PRNG: "offset = (currentWeek - 1) % 24 + 1
 (contracts/PreciousHour.sol#120)"
Reference: https://github.com/crytic/slither/wiki/Detector-
 Documentation#weak-PRNG


Math.mulDiv(uint256,uint256,uint256)
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#55-134)
 performs a multiplication on the result of a division:
      - denominator = denominator / twos
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#101)
      - inverse = (3 * denominator) ^ 2
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#116)
Math.mulDiv(uint256,uint256,uint256)
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#55-134)
 performs a multiplication on the result of a division:
      - denominator = denominator / twos
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#101)
      - inverse *= 2 - denominator * inverse
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#120)
Math.mulDiv(uint256,uint256,uint256)
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#55-134)
 performs a multiplication on the result of a division:
      - denominator = denominator / twos
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#101)
      - inverse *= 2 - denominator * inverse
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#121)
Math.mulDiv(uint256,uint256,uint256)
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#55-134)
 performs a multiplication on the result of a division:
      - denominator = denominator / twos
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#101)
      - inverse *= 2 - denominator * inverse
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#122)
Math.mulDiv(uint256,uint256,uint256)
 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#55-134)
 performs a multiplication on the result of a division:
```

Gas requirement of 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)

- denominator = denominator / twos
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#101)
- inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#123)
Math.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#55-134) performs a multiplication on the result of a division:
- denominator = denominator / twos
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#101)
- inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#124)
Math.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#55-134) performs a multiplication on the result of a division:
- denominator = denominator / twos
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#101)
- inverse *= 2 - denominator * inverse
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#125)
Math.mulDiv(uint256,uint256,uint256)
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#55-134) performs a multiplication on the result of a division:
- prod0 = prod0 / twos
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#104)
- result = prod0 * inverse
(node_modules/@openzeppelin/contracts/utils/math/Math.sol#131)
Base64.encode(bytes) (contracts/Base64.sol#8-46) performs a multiplication on the result of a division:
- result = new string(4 * ((data.length + 2) / 3)) (contracts/Base64.sol#13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in PreciousHour.mint() (contracts/PreciousHour.sol#29-40):
External calls:
- _safeMint(msg.sender,supply + 1) (contracts/PreciousHour.sol#38)
- IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,data)
(node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#406-417)
State variables written after the call(s):
- tokensSold += 1 (contracts/PreciousHour.sol#39)
PreciousHour.tokensSold (contracts/PreciousHour.sol#20) can be used in cross function reentrancies:
- PreciousHour.constructor() (contracts/PreciousHour.sol#22-27)
- PreciousHour.mint() (contracts/PreciousHour.sol#29-40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

ERC721._checkOnERC721Received(address,address,uint256,bytes)
(node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#399-421)
ignores return value by
IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,data)
(node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#406-417)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise,
after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed
condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5
Contracts/PreciousHour.sol Contracts/Base64.sol	a95dcfd07bf2ec8a609ded6b023fc4fb

Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source Sha1 Hash
Contracts /ERC721, Ownable ERC721Enumerable,ReentrancyGuard,SafeMath,Strings	306190b67f5c094c437966915a2dafb468cad8ee

0.2 SOLIDITY UNIT TESTING

Progress: 2 finished (of 2)
PASS ✓
Tested (✓)

✓ Check winning proposal
✓ Check winnin proposal with return value
✓ Before all
✓ Check success
✓ Check success2
✓ Check sender and value

Result for tests Passed:

0Time Taken: 0.18s

0.3 TESTING

```
Different versions of Solidity are used:
	- Version used: ['>=0.7.0<0.9.0', '^0.8.0', '^0.8.1']
	- >=0.7.0<0.9.0 (contracts/PreciousHour.sol#2)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4)
	- ^0.8.0
	(node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4)
	- ^0.8.0
	(node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerab
	le.sol#4)
	- ^0.8.0
	(node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumera
	ble.sol#4)
	- ^0.8.0
	(node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata
	.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SignedMath.sol#4)
	- ^0.8.0 (contracts/Base64.sol#3)
	- ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
	pragma-directives-are-used
Function ERC721.__unsafe_increaseBalance(address,uint256)
	(node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#463-465) is not in
	mixedCase
Function PreciousHour.buildAnimation_url() (contracts/PreciousHour.sol#46-54) is not in
	mixedCase
Parameter PreciousHour.setTimeZoneOffset(uint256)._timeZoneOffset
	(contracts/PreciousHour.sol#104) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
	solidity-naming-conventions

ERC721._checkOnERC721Received(address,address,uint256,bytes)
	(node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#399-421) uses
	assembly
	- INLINE ASM
	(node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#413-415)
Address._revert(bytes,string)
	(node_modules/@openzeppelin/contracts/utils/Address.sol#231-243) uses assembly
	- INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#236-239)
Strings.toString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#19-39)
	uses assembly
	- INLINE ASM (node_modules/@openzeppelin/contracts/utils/Strings.sol#25-27)
	- INLINE ASM (node_modules/@openzeppelin/contracts/utils/Strings.sol#31-33)
Math.mulDiv(uint256,uint256,uint256)
	(node_modules/@openzeppelin/contracts/utils/math/Math.sol#55-134) uses assembly
	- INLINE ASM (node_modules/@openzeppelin/contracts/utils/math/Math.sol#62-66)
	- INLINE ASM (node_modules/@openzeppelin/contracts/utils/math/Math.sol#85-92)
	- INLINE ASM (node_modules/@openzeppelin/contracts/utils/math/Math.sol#99-108)
Base64.encode(bytes) (contracts/Base64.sol#8-46) uses assembly
	- INLINE ASM (contracts/Base64.sol#15-44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

Contract Gas Snapshot

Gas costs and Onchain
Gas requirement of function that many 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)

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

preciousHour.buildAnimation_url() (contracts/PreciousHour.sol#46-54) uses literals with too many digits:
- Base64.encode(bytes(abi.encodePacked(<html><head><style>@keyframes rotateArms{from{transform:rotate(0)}to{transform:rotate(360deg)}}@-webkit-keyframes rotateArms{from{transform:rotate(0)}to{transform:rotate(360deg)}}@-moz-keyframes rotateArms{from{transform:rotate(0)}to{transform:rotate(360deg)}}body{background-color:#fff}.cl{width:200px;height:200px;box-shadow:0 0 0 10px #000;border-radius:200px;position:relative;background-color:#fff;zoom:140%}.c,.m>div{position:absolute;top:50%;left:50%;transform:translate(-50%,-50%)}.k{width:13.33333px;height:13.33333px;background:#a00;border-radius:13.33333px;z-index:4}.a{position:absolute;top:50%;left:50%;transform-origin:top left 0;z-index:2;width:0;animation:rotateArms linear 60s infinite;-webkit-animation:rotateArms linear 60s infinite;-moz-animation:rotateArms linear 60s infinite}.mi.a{height:76.92308px;box-shadow:0 0 0 2px #000;animation-duration:3600s;-webkit-animation-duration:3600s;-moz-animation-duration:3600s}.h.a{height:55.55556px;box-shadow:0 0 0 3.33333px #000;animation-duration:43200s;-webkit-animation-duration:43200s;-moz-animation-duration:43200s}.s.a{height:90.90909px;box-shadow:0 0 0 1.33333px #a00}.box{position:absolute;top:0;left:0;width:100%;height:100%}.m{width:100%;height:100%}.m>div::before,.m>div::after{content:"";position:absolute;width:0;height:5px;top:88.88889px;box-shadow:0 0 0 .5px #000}.m>div::after{bottom:88.88889px;top:auto}.m>div::after{bottom:88.88889px;top:auto}div.long::before,div.long::after{height:9.09091px;top:84.4773px}div.long::after{bottom:84.4773px;top:auto}div.dark::before,div.dark::after{box-shadow:0 0 0 1px #000}.m>.m0{transform:rotate(0deg)}.m>.m1{transform:rotate(6deg)}.m>.m2{transform:rotate(12deg)}.m>.m3{transform:rotate(18deg)}.m>.m4{transform:rotate(24deg)}.m>.m5{transform:rotate(30deg)}.m>.m6{transform:rotate(36deg)}.m>.m7{transform:rotate(42deg)}.m>.m8{transform:rotate(48deg)}.m>.m9{transform:rotate(54deg)}.m>.m10{transform:rotate(60deg)}.m>.m11{transform:rotate(66deg)}.m>.m12{transform:rotate(72deg)}.m>.m13{transform:rotate(78deg)}.m>.m14{transform:rotate(84deg)}.m>.m15{transform:rotate(90deg)}.m>.m16{transform:rotate(96deg)}.m>.m17{transform:rotate(102deg)}.m>.m18{transform:rotate(108deg)}.m>.m19{transform:rotate(114deg)}.m>.m20{transform:rotate(120deg)}.m>.m21{transform:rotate(126deg)}.m>.m22{transform:rotate(132deg)}.m>.m23{transform:rotate(138deg)}.m>.m24{transform:rotate(144deg)}.m>.m25{transform:rotate(150deg)}.m>.m26{transform:rotate(156deg)}.m>.m27{transform:rotate(162deg)}.m>.m28{transform:rotate(168deg)}.m>.m29{transform:rotate(174deg)}svg{position:relative;height:20%;left:40%;top:16%;fill:none;stroke:#000;stroke-width:4;stroke-miterlimit:10}</style></head><body><div class="cl
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

0.1 Auto Debugging

```
block.chainid:0xd05
block.coinbase:0x0000000000000000000000000000000000000000
block.difficulty:0
block.gaslimit:83124
block.number:1
block.timestamp:1686928599
msg.sender:0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
msg.sig:0x60566050
msg.value:0 Wei
tx.origin:0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
block.basefee:1 Wei (1
```

- execution step:0
- add memory:
- gas:3
- remaining gas:28128
- loaded address:(Contract Creation - Step 0)

"0x60566050600b82828239805160001a6073146043577f4e487b710000000000000000000000000000000000000000000000000000000000000000600052600060045260246000fd5b30600052607381538281f3fe730000000000000000000000000000000000000000301460806040526000 80fdfea2646970667358221220cb6cd1843f247598bd0fd0ad1b8ca79feebe0e1b1f162470d87e8b8be631890d64736f6c634300080f0033"]

Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, AudiTBlock experts found **0 big Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, AudiTBlock experts found **3 High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, AudiTBlock experts found **2 Medium issues** in the code of the smart contract.

LOW ISSUES

During the audit, AudiTBlock experts found **1 Low issues** in the code of the smart contract.

INFORMATIONAL ISSUES

During the audit, AuditBlock experts found **2 Informational issues** in the code of the smart contract.

SWC Attacks

ID	Title		Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✔
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✔
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✔
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✔
SWC-127	Arbitrary Jump with Function TypeVariable	CWE-695: Use of Low-Level Functionality	✔
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✔
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✔
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✔

ID	Title		Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✔
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✔
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✔
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✔
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✔
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✔
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✔
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✔
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✔
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✔

Owner privileges

- Verify Claims
- Status: tested 1 and verified ✓
- Status: tested 2 and verified ✓
- Status: tested 3 and verified ✓
- Status: tested 4 and verified ✓
- Status: tested 5 and verified ✓
- Status: tested 6 and verified ✓

Executive Summary

Two (2) independent AuditBlock experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs
The overall code quality is good and not overloaded with unnecessary functions, these is greatly

benefiting the security of the contract. It correctly implemented widely used and reviewed contracts from OpenZeppelin. he main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work.

During the audit, no issues were found after the manual and automated security testing.

VERIFIED ✔

`PreciousHour.sol`