



Cyberscope

Audit Report

Avatar a5

January 2023

Github https://github.com/oxvimmmm/avatar_a5

Commit [8b660c876718a7e9c935e602d8776a766d693f64012a522efc60b1ac4a9b43c4aab19705a1140a47](#)

Audited by © cyberscope

Table of Contents

Table of Contents	1
Review	2
Audit Updates	2
Source Files	2
Diagnostics	3
PMV - Potential Misleading Value	4
Description	4
Recommendation	4
Team Update	4
RV - Randomization Vulnerability	5
Description	5
Recommendation	5
DTM - Data Type Misuse	6
Description	6
Recommendation	6
Team Update	6
L04 - Conformance to Solidity Naming Conventions	7
Description	7
Recommendation	7
L14 - Uninitialized Variables in Local Scope	9
Description	9
Recommendation	9
Functions Analysis	10
Inheritance Graph	13
Flow Graph	14
Summary	15
Disclaimer	16
About Cyberscope	17

Review

Repository	https://github.com/oxvimmmm/avatar_a5
Commit	8b660c876718a7e9c935e602d8776a766d693f64 012a522efc60b1ac4a9b43c4aab19705a1140a47

Audit Updates

Initial Audit	03 Feb 2023
----------------------	-------------

Source Files

Filename	SHA256
Avatar.sol	be2054bd5885b5f9ea92d09a8e0aa9ef93 09c465eee512e81aba8a40ca32d000
Bucket.sol	d44fad02536763eb73fe5851557b6ca721 70ed68abd2c0dfd5f4ce9e8de9bdea
PaymentSplitter.sol	3d6f05b7f58486efc348b9e636c1431b7c b7250aa4362ddbec8822589df52cb6

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	PMV	Potential Misleading Value	Resolved
●	RV	Randomization Vulnerability	Unresolved
●	DTM	Data Type Misuse	Acknowledged
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved

PMV - Potential Misleading Value

Criticality	Minor / Informative
Location	Bucket.sol#L103
Status	Resolved

Description

The contract is utilizing a binary search in order to achieve $O(\log n)$ time complexity in the search of the ordered array. If the value is matched, then the index of the array is returned. The return value is cast to `uint16`. That means if the array length is more than 2^{16} then the function will yield misleading values.

```
function binarySearch(uint16[] storage array, uint16 target) internal view
returns (uint16) {
    ...
    if (target < array[left]) {
        return uint16(left);
    } else {
        return uint16(right);
    }
}
```

Recommendation

The team is advised to implement a `require()` check in the length of the array so it will guarantee that the binary search algorithm will always yield the correct values.

Team Update

The team has added a check to guarantee that the `array` will never be greater than 2^{16} .

```
require(stockPrefixSum.length <= 2e16, "stock length too long");
```

RV - Randomization Vulnerability

Criticality	Minor / Informative
Location	Bucket.sol#L79
Status	Unresolved

Description

The contract is using an on-chain technique in order to determine random numbers. The blockchain runtime environment is fully deterministic, as a result, the pseudo-random numbers could be predicted.

```
function _getRandomIndex(uint256 seed, uint16 size) internal view returns
(uint16) {
    keccak256(
        abi.encodePacked(
            block.difficulty,
            block.timestamp,
            ...
        )
    )
}
```

Recommendation

The contract could use an advanced randomization technique that guarantees an acceptable randomization factor. For instance, the Chainlink VRF (Verifiable Random Function). <https://docs.chain.link/docs/chainlink-vrf>

DTM - Data Type Misuse

Criticality	Minor / Informative
Status	Acknowledged

Description

The contract is using the `ledgerType` an enumerated integer. It accept values between 0 and 6. The `ledgerType` data type is `uint256`. Since it will always be less than 6, then the 2^{256} data type necessary reserves space.

```
require(ledgerType < 6, "Invalid ledger type");
```

Recommendation

The team is advised to use the minimum data type that can model a maximum of 6 integer variable. A suggested implementation could be to utilize an `uint8` data type.

Team Update

"It truly saved storage costs when we could pack the variable or call data. However, almost all function parameters throughout the whole contract, there are no cases where we can pack uints with other variables that are full 256 size. Therefore, there is no significant benefit in our project, especially when doing comparisons, as it might cost extra gas for doing implicit conversion between uints and uint256. Therefore, we decided not to modify the data types for ledgerType."

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Avatar.sol#L23,206,218
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
uint256[6] public DEFAULT_TARGET_AMOUNTS = [13e22, 25e22, 35e22, 50e22, 75e22, 125e22]
bool _paused
address _operator
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention>.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	Avatar.sol#L741,1005
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 payoutAmount  
ReferrerSearch memory search
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

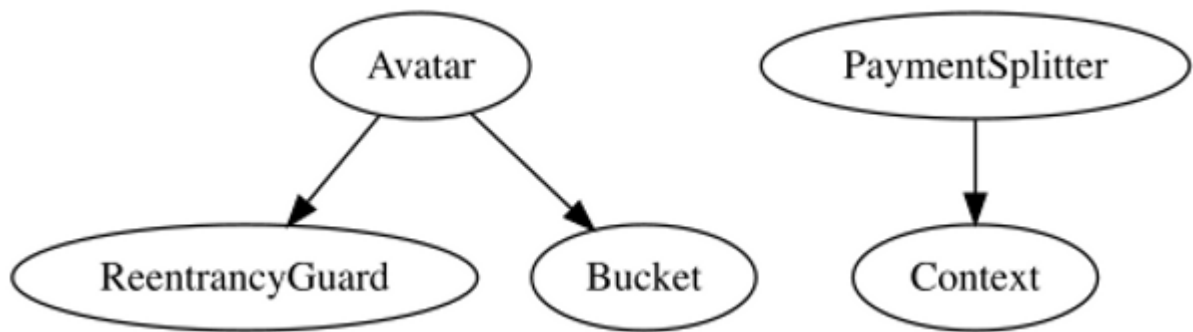
Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Avatar	Implementation	Reentrancy Guard, Bucket		
		Public	✓	-
	setPause	External	✓	-
	transferOperator	External	✓	-
	dropTempAdmin	External	✓	-
	batchSetReferrerInfo	External	✓	-
	setStock	External	✓	-
	openPosition	External	Payable	notContract nonReentrant
	closePosition	External	✓	notContract nonReentrant
	batchClosePositions	External	✓	nonReentrant
	batchClaimPositionIncentiveReward	External	✓	notContract nonReentrant
	batchReportSales	External	✓	-
	claimReferrerReward	External	✓	notContract nonReentrant
	getLinkedPositionInfo	External		-
	getUserRounds	External		-
	getUserRoundsLength	External		-
	getUserRoundLedgers	External		-
	getUserRoundLedgersLength	External		-
	getChildren	External		-
	getLedgerRoundToUserRoundIndex	External		-
	getChildrenLength	External		-

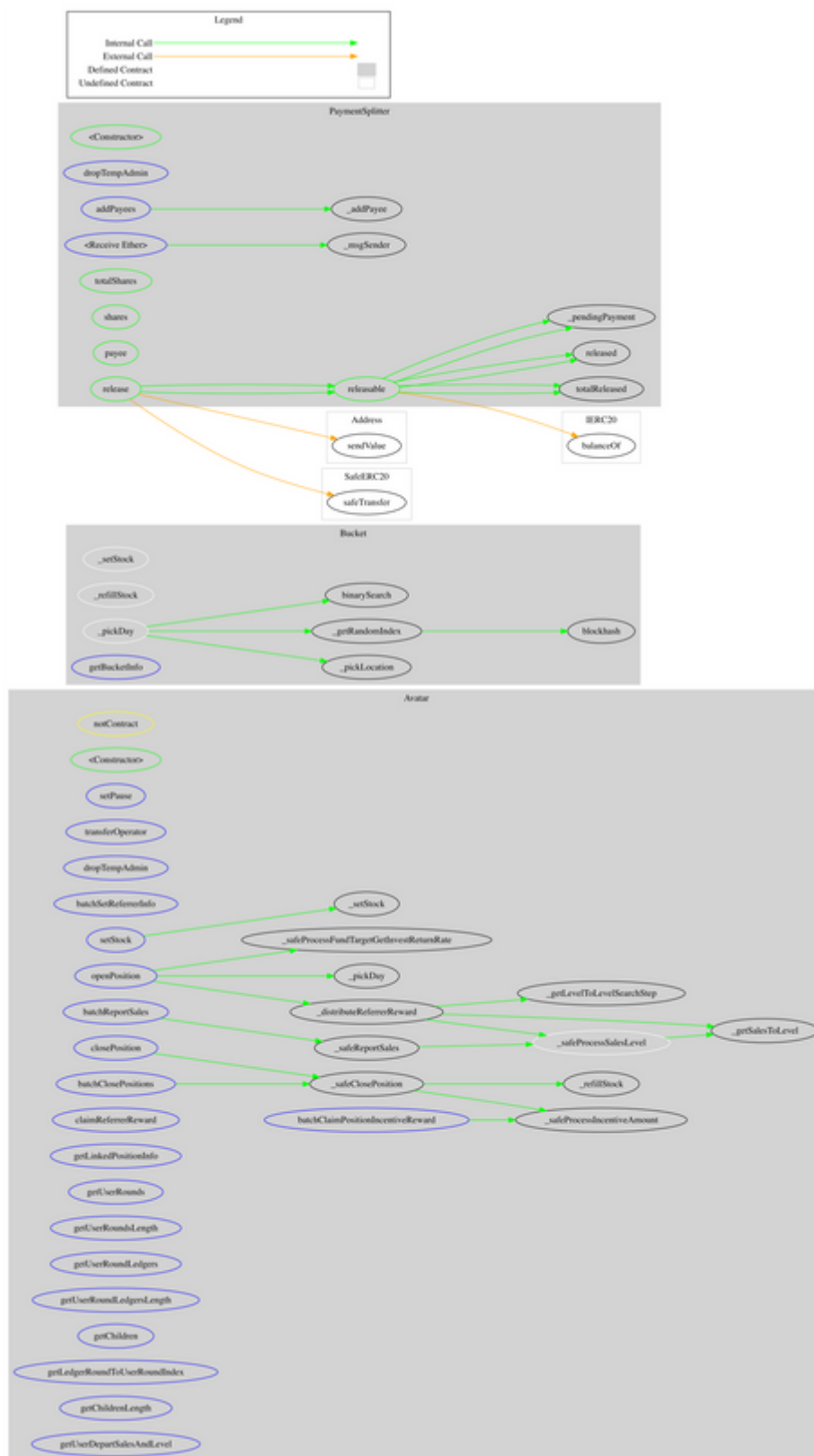
	getUserDepartSalesAndLevel	External		-
	_safeClosePosition	Internal	✓	
	_safeProcessFundTargetGetInvestReturnRate	Internal	✓	
	_safeProcessIncentiveAmount	Internal	✓	
	_safeProcessSalesLevel	Internal	✓	
	_safeReportSales	Internal	✓	
	_distributeReferrerReward	Internal	✓	
	_getSalesToLevel	Internal		
	_getLevelToLevelSearchStep	Internal		
Bucket	Implementation			
	_setStock	Internal	✓	
	_refillStock	Internal	✓	
	_pickDay	Internal	✓	
	_pickLocation	Internal	✓	
	_getRandomIndex	Internal		
	getBucketInfo	External		-
	binarySearch	Internal		
PaymentSplitter	Implementation	Context		
		Public	Payable	-
	dropTempAdmin	External	✓	-
	addPayees	External	✓	-
		External	Payable	-
	totalShares	Public		-
	totalReleased	Public		-
	totalReleased	Public		-
	shares	Public		-
	released	Public		-

	released	Public		-
	payee	Public		-
	releasable	Public		-
	releasable	Public		-
	release	Public	✓	-
	release	Public	✓	-
	_pendingPayment	Private		
	_addPayee	Private	✓	

Inheritance Graph



Flow Graph



Summary

Avatar a5 contract implements a financial mechanism. This audit investigates security issues, business logic concerns and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



The Cyberscope team

<https://www.cyberscope.io>