Predict RAM

Smart Contract Review

Deliverable: Smart Contract Audit Report

Security Report

March 2022

Report Summary

Title	**Predict RAM Smart Contract Audit**		
Project Owner	Predict RAM		
Type	Private		
Reviewed by	Kyohei Ito	**Revision date**	26/03/2022
		Approval date	28/03/2022
		N° Pages	**15**

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant effect on the security of the DeFiproject, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It isstrongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It isrecommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whetherthese vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

The Full List of Check Items:

Category	Check Item
Basic Coding Bugs	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	MONEY-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead of Transfer
	Costly Loop
	(Unsafe) Use of Untrusted Libraries
	(Unsafe) Use of Predictable Variables
	Transaction Ordering Dependence
	Deprecated Uses
Semantic Consistency Checks	Semantic Consistency Checks
	Business Logics Review
	Functionality Checks

Advanced DeFi Scrutiny	Authentication Management
	Access Control & Authorization
	Oracle Security
	Digital Asset Escrow
	Kill-Switch Mechanism
	Operation Trails & Event Generation
	ERC20 Idiosyncrasies Handling
	Frontend-Contract Integration
	Deployment Consistency
	Holistic Risk Management
Additional Recommendations	Avoiding Use of Variadic Byte Array
	Using Fixed Compiler Version
	Making Visibility Level Explicit
	Making Type Inference Explicit
	Adhering To Function Declaration Strictly
	Following Other Best Practices

Common Weakness Enumeration (CWE) Classifications Used in This Audit:

Category	Summary
Configuration	Weaknesses in this category are typically introduced during the configuration of the software.
Data Processing Issues	Weaknesses in this category are typically found in functionality that processes data.
Numeric Errors	Weaknesses in this category are related to improper calculation or conversion of numbers.
Security Features	Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)
Time and State	Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiplesystems, processes, or threads.
Error Conditions, Return Values, Status Codes	Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code,or if the application does not handle all possible return/status codes that could be generated by a function.
Resource Management	Weaknesses in this category are related to improper management of system resources.

Behavioral Issues	Weaknesses in this category are related to unexpected behaviors from code that an application uses.
Business Logics	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application.
Initialization and Cleanup	Weaknesses in this category occur in behaviors that are used for initialization and breakdown.
Arguments and Parameters	Weaknesses in this category are related to improper use arguments or parameters within function calls.
Expression Issues	Weaknesses in this category are related to incorrectly written expressions within code.
Coding Practices	Weaknesses in this category are related to coding practices that are deemed unsafe and increase the chances that an ex pilotable vulnerability will be present in the application. They may not directly introduce a vulnerability, but indicate the product has not been carefully developed or maintained.

Findings

Summary

Here is a summary of the findings after analyzing the Predict RAM Smart Contract Review. During the first phase of the audit, I studied the smart contract source code and ran my in-house static code analyzer through the Specific tool. The purpose here is to statically identify known coding bugs, and then manually verify (reject or confirm) issues reported by tool. I further manually review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

Severity	No. of Issues
Critical	0
High	0
Medium	0
Low	3
Total	3

We have so far identified that there are potential issues with severity of **0 Critical, 0 High, 0 Medium, and 3 Low**. Overall, these smart contracts are well-designed and engineered.

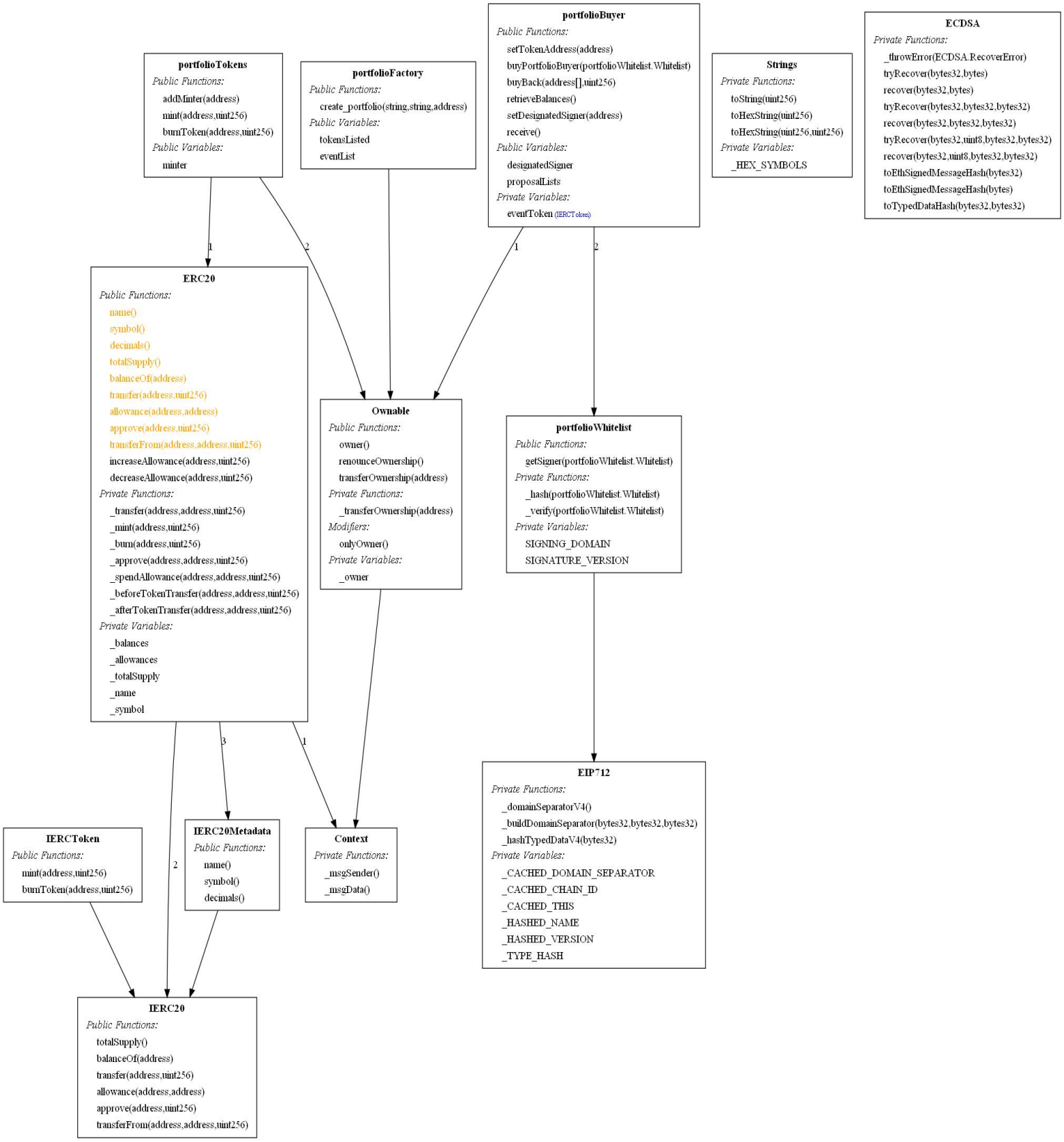
Functional Overview

($) = payable function # = non-constant function	[Pub] public [Ext] external [Prv] private [Int] internal

+ [Int] IERCToken (IERC20)
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #
  - [Ext] mint
  - [Ext] burnToken
+ portfolioBuyer (Ownable, porfolioWhitelist)
  - [Pub] <Constructor> #
  - [Prv] eventToken
  - [Pub] designatedSigner
  - [Pub] proposalLists
  - [Ext] setTokenAddress
    - modifiers: onlyOwner
  - [Ext] buyPotfolioBuyer $
  - [Ext] buyBack
    - modifiers: onlyOwner
  - [Ext] retrieveBalances
    - modifiers: onlyOwner

```
  - [Ext] setDesinatedSigner
    - modifiers: onlyOwner

  - [Ext] receive $

+  portfolioFactory (Ownable)

  - [Pub] tokensListed

  - [Pub] eventList

  - [Ext] create_portfolio
    - modifiers: onlyOwner

 +  portfolioTokens (ERC20, Ownable)

   - [Pub] <Constructor> #

   - [Pub] minter

   - [Ext] addMinter
     - modifiers: onlyOwner

   - [Ext] mint

   - [Ext] burnToken

 +  portfolioWhitelist (EIP712)

    - [Pub] <Constructor> #

    - [Prv] SIGNING_DOMAIN

    - [Prv] SIGNATURE_VERSION

    - [Pub] getSigner

    - [Int] _hash

    - [Int] _verify
```

Inheritance



Detailed Results

Issues Checking Status

1. Local Variables Shadowing

- Severity: Low
- Location:
  - portfolioFactory.sol#14
    create_portfolio.owner shadows Ownable.owner
- Description: variable shadowing occurs when a variable declared within a certain scope (decision block, method, or inner class) has the same name as a variable declared in an outer scope
- Remediations: Rename the local variables that shadow another component.

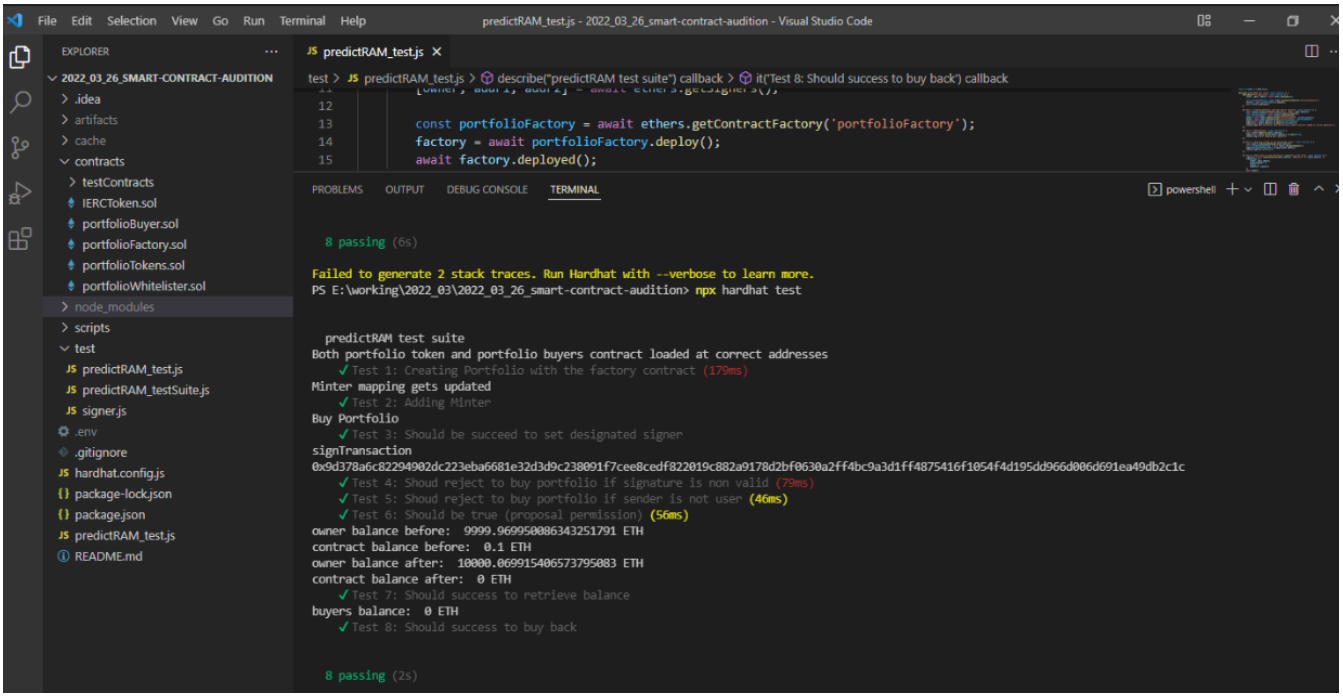
2. External Calls inside a loop

- Severity: Low
- Location:
  - portfolioBuyer.sol#34-36
- Description:Calls inside a loop might lead to a denial-of-service attack. If one of the destinations has a fallback function that reverts, it will always revert
- Remediations: Favor pull over push strategy for external calls.
  see https://eth.wiki/en/howto/smart-contract-safety#favor-pull-over-push-for-external-calls

3. Missing zero address validation

- Severity: Low
- Location:
  - portfolioBuyer.sol#44
- Description: If the parameter address is non-available, the signer of the portfolioBuyer will get lost.
- Remediations: Check that the address is not zero.

Unit Testing Results



Basic Coding Bugs

No.	Name	Description	Severity	Result
1.	Constructor Mismatch	Whether the contract name and its constructor are not identical to each other.	Critical	PASSED
2.	Ownership Takeover	Whether the set owner function is not protected.	Critical	PASSED
3.	Redundant Fallback Function	Whether the contract has a redundant fallback function.	Critical	PASSED
4.	Overflows & Underflows	Whether the contract has general overflow or underflow vulnerabilities	Critical	PASSED
5.	Reentrancy	Reentrancy is an issue when code can call back into your contract and change state, such as withdrawing ETHs	Critical	PASSED
6.	MONEY-Giving Bug	Whether the contract returns funds to an arbitrary address	High	PASSED
7.	Blackhole	Whether the contract locks ETH indefinitely: merely in without out	High	PASSED
8.	Unauthorized Self-Destruct	Whether the contract can be killed by any arbitrary address	Medium	PASSED

9.	Revert DoS	Whether the contract is vulnerable to DoS attack because of unexpected revert	Medium	**PASSED**
10.	Unchecked External Call	Whether the contract has any external call without checking the return value	Medium	**PASSED**
11.	Gasless Send	Whether the contract is vulnerable to gasless send	Medium	**PASSED**
12.	Send Instead of Transfer	Whether the contract uses send instead of transfer	Medium	**PASSED**
13.	Costly Loop	Whether the contract has any costly loop which may lead to Out-Of-Gas exception	Medium	**PASSED**
14.	(Unsafe) Use of Untrusted Libraries	Whether the contract use any suspicious libraries	Medium	**PASSED**
15.	(Unsafe) Use of Predictable Variables	Whether the contract contains any randomness variable, but its value can be predicated	Medium	**PASSED**
16.	Transaction Ordering Dependence	Whether the final state of the contract depends on the order of the transactions	Medium	**PASSED**

17.	Deprecated Uses	Whether the contract use the deprecated tx.origin to perform the authorization	Medium	PASSED
18.	Semantic Consistency Checks	Whether the semantic of the white paper is different from the implementation of the contract	Critical	PASSED

Conclusion

In this audit, I thoroughly analyzed Predict RAM's Smart Contract. The current code base is well organized but there are promptly some low-level issues found in this phase of Smart Contract Audit.

Meanwhile, we need to emphasize that smart contracts as a whole are still in an early, but exciting stage of development. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.