Report for:

PWN Finance

March 2022

Version: 1.0

Prepared By: Extropy.IO
Email: info@extropy.io
Telephone: +44 1865261424

Table of Contents

1. Executive Summary 3

1.1. Assessment Summary 3

2. Using This Report 4

2.1. Disclaimer 4
2.2. Client Confidentiality 4
2.3. Proprietary Information 4

3. Technical Summary 5

3.1. Scope 5
3.2. Inheritance graph 5
3.3. Call Graph 6
3.4. Test Coverage 8

4. Technical Findings 9

4.1. Create Loan - invalid asset category 9
4.2. Claim Loan - loan status 9
4.3. Optimisation - Error messages 10
4.4. Optimisation - Use external functions where possible 10
4.5. Incomplete documentation 12

5. Tool List 13

6. General Audit Goals 14

1. Executive Summary

Extropy was contracted to conduct a code review and vulnerability assessment of the development branch of the PWN project. The review was carried out between 24th and 30th March 2022.
The contracts have no significant issues, minor issues are detailed in section 4.

1.1. Assessment Summary

The contracts are well designed and implemented and follow known best practices. Test coverage and documentation is good.
Some minor issues mainly concerning optimisation and edge cases are detailed below.

Static Analysis tools did not find any further issues.

Phase	Description	Critical	High	Medium	Low	Info	Total
1	Initial Audit	0	0	0	2	3	5

2. Using This Report

To facilitate the dissemination of the information within this report throughout your organisation, this document has been divided into the following clearly marked and separable sections.

Executive Summary	Management level, strategic overview of the assessment and the risks posed to the business
Technical Summary	An overview of the assessment from a more technical perspective, including a defined scope and any caveats which may apply
Technical Findings	Detailed discussion (including evidence and recommendations) for each individual security issue which was identified
Methodologies	Audit process and tools used

2.1. Disclaimer

The audit makes no statements or warranty about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only

2.2. Client Confidentiality

This document contains Client Confidential information and may not be copied without written permission.

2.3. Proprietary Information

The content of this document should be considered proprietary information. Extropy gives permission to copy this report for the purposes of disseminating information within your organisation or any regulatory agency.

Document Version Control			
Data Classification	Client Confidential		
Client Name	PWN Finance		
Document Title	PWN Finance Audit		
Author	Extropy Audit Team		

Document History			
Issue No.	**Issue Date**	**Issued By**	**Change Description**
1.0	31/03/2022	Laurence Kirk	Released to client

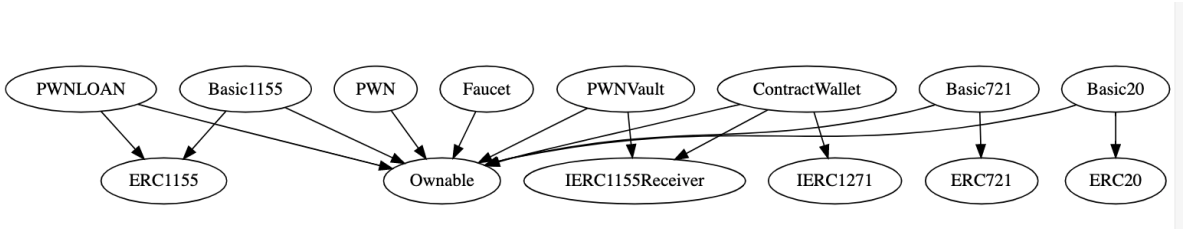
3. Technical Summary

3.1. Scope

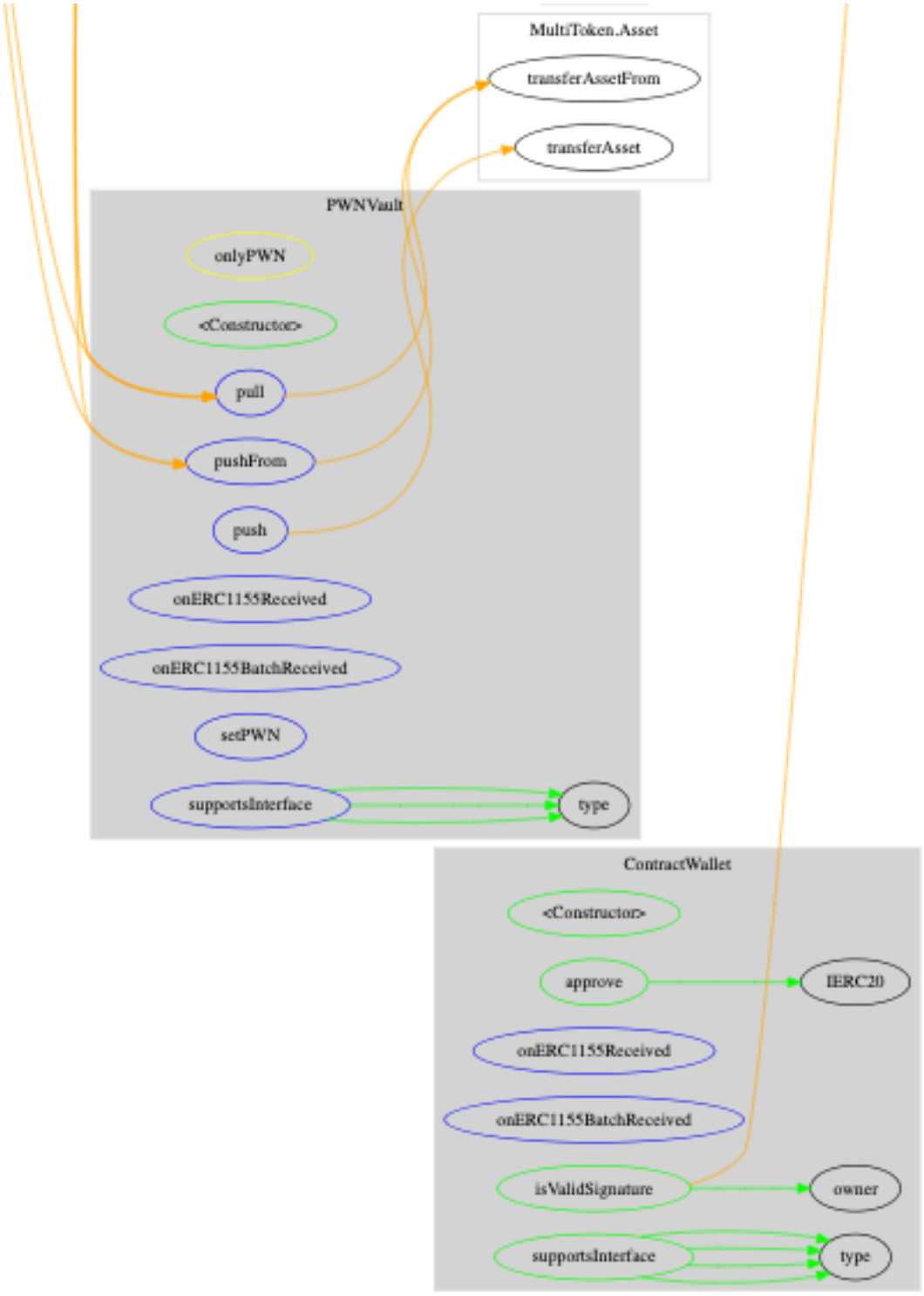
The code audited is from the [pwn_contracts github repo]
at commit fcd21428ebdfe12c9f90f975e83f124734ffb7cc on the develop branch.

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
	contracts/PWNVault.sol	1		164	142	52	85	37
	contracts/PWN.sol	1		177	167	77	59	46
	contracts/PWNLOAN.sol	1		614	583	265	246	133
	contracts/Testing tokens/ContractWallet.sol	1		74	49	32	14	29
	contracts/Testing tokens/Faucet.sol	1		86	86	79	4	70
	contracts/Testing tokens/Basic1155.sol	1		24	24	15	4	14
	contracts/Testing tokens/Basic721.sol	1		38	38	22	8	18
	contracts/Testing tokens/Basic20.sol	1		24	24	15	4	14
	Totals	**8**		**1201**	**1113**	**557**	**424**	**361**

3.2. Inheritance graph



3.3. Call Graph


MultiToken.Asset
transferAssetFrom
transferAsset
PWNVault
onlyPWN
<Constructor>
pull
pushFrom
push
onERC1155Received
onERC1155BatchReceived
setPWN
supportsInterface
type
ContractWallet
<Constructor>
approve
IERC20
onERC1155Received
onERC1155BatchReceived
isValidSignature
owner
supportsInterface
type

3.4. Test Coverage

100% Statements `120/120` **97.62%** Branches `41/42` **100%** Functions `39/39` **100%** Lines `122/122`

File		Statements		Branches		Functions		Lines	
PWN.sol		100%	32/32	75%	3/4	100%	6/6	100%	31/31
PWNLOAN.sol		100%	74/74	100%	36/36	100%	24/24	100%	76/76
PWNVault.sol		100%	14/14	100%	2/2	100%	9/9	100%	15/15

4. Technical Findings

The remainder of this document is technical in nature and provides additional detail about the items already discussed, for the purposes of remediation and risk assessment.

.

4.1. Create Loan - invalid asset category

Risk Rating	Low

Affects : PWN.sol

Description:

The asset category for provided collateral is not validated when creating a loan, it would be possible to create a loan with an incorrect category type, leading to unexpected behaviour.

Recommendation :

Add validation for the asset category

A similar issue applies to the createDeed function on the master branch, where there could be a mismatch between the actual asset specified with its address, and the asset category supplied.

4.2. Claim Loan - loan status

Risk Rating	Low

Affects : PWN.sol

Description:

The loan status is stored at the beginning of the claimLoan function, then changed by calling the LOAN.claim() function; the value that was first stored before calling LOAN.claimed() is then used for some business logic:

```
function claimLoan(uint256 _loanId) external returns (bool) {
        uint8 status = LOAN.getStatus(_loanId);

        LOAN.claim(_loanId, msg.sender);
```

```
        if (status == 3) {
            MultiToken.Asset memory LoanAsset = LOAN.getLoanAsset(_loanId);
            LoanAsset.amount = LOAN.getLoanRepayAmount(_loanId);

            vault.push(LoanAsset, msg.sender);
        } else if (status == 4) {
            vault.push(LOAN.getCollateral(_loanId), msg.sender);
        }
```

The claim function in PWNLoan however, resets the loanId to 0 should the loan status number be 3 or higher:

```
function claim(
        uint256 _loanId,
        address _owner
    ) external onlyPWN {
        require(balanceOf(_owner, _loanId) == 1, "Caller is not the loan
owner");
        require(getStatus(_loanId) >= 3, "Loan can't be claimed yet");
```

If the status happened to be higher than 4, the transaction would burn the token and never pay the lender back principal or collateral amount.

Recommendation :

Add further checks for the status

4.3. Optimisation - Error messages

Risk Rating	Informational

Affects : PWNLoan.sol

Description:

Error messages longer than 32 bytes increase deployment costs

Recommendation :

Reduce the size of error messages within require statements, if needed, one approach is to give the error message a short code which is explained in the documentation.

4.4. Optimisation - Use external functions where possible

Risk Rating	Informational

Affects : PWNLoan.sol

Description:

The following functions could be declared as external to reduce gas costs.

- getExpiration
- getDuration
- getBorrower
- getCollateral
- getLoanAsset
- getLoanRepayAmount
- isRevoked

Recommendation :

Set the function visibility to external if internal access is not needed.

4.5. Incomplete documentation

Risk Rating	Informational

Affects : PWNLoan.sol

Description:

There is a TODO in the comments

```
   * Construct defining a LOAN which is an acronym for: ... (TODO)
```

```
also line 460 has the incorrect function name
```

```
 getLoan instead of getLoanRepayAmount
```

Recommendation :

Correct the documentation

5. Tool List

The following tools were used during the assessment:

Tools Used	Description	Resources
SWC Registry	Vulnerability database	https://swcregistry.io/
Solidity Metrics	Static Analysis	https://github.com/ConsenSys/ solidity-metrics

6. General Audit Goals

We audit the code in accordance with the following criteria:

Sound Architecture

This audit includes assessments of the overall architecture and design choices. Given the subjective nature of these assessments, it will be up to the development team to determine whether any changes should be made.

Smart Contract and Rust Best Practices

This audit will evaluate whether the codebase follows the current established best practices for smart contract development.

Code Correctness

This audit will evaluate whether the code does what it is intended to do.

Code Quality

This audit will evaluate whether the code has been written in a way that ensures readability and maintainability.

Security

This audit will look for any exploitable security vulnerabilities, or other potential threats to the users.

Although we have commented on the application design, issues of crypto-economics, game theory and suitability for business purposes as they relate to this project are beyond the scope of this audit.