HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Yama Finance
Date: March 08, 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Yama Finance
Approved By	Noah Jelich | Lead Solidity SC Auditor at Hacken OU
Type	ERC20 token; Lending Platform; Auction; Bridge
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://yama.finance/
Changelog	16.02.2023 - Initial Review 08.03.2023 - Second Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Yama Finance (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is review and security analysis of smart contracts in the repository:

Initial review scope

Repository	https://github.com/Yama-Finance/stablecoin-contracts
Commit	21345d1853c79f78e7519f2e196a1bea5b6e41bc
Whitepaper	
Functional Requirements	
Technical Requirements	
Contracts	File: ./src/interfaces/IBalanceSheetHandler.sol SHA3: ef9ddae0f91cbaf7fa1cc10ec9c5a258df3fc997e3373895fa5ad0e53e679013 File: ./src/interfaces/IBridgeReceiver.sol SHA3: 14546aa45a58f7b2ca9c46f366669e6e9143a5199d5bf964201168925cd792d4 File: ./src/interfaces/ICollateralManager.sol SHA3: eec57035c10fb81efdde2556cf7df3e4b37573f72bed357a76b22c4b9ee920bc File: ./src/interfaces/ILiquidator.sol SHA3: 66768c4de5fb0bfc40d28b8c69e0dafc821efb1ae7b7db739773395bf737fc97 File: ./src/interfaces/IPriceSource.sol SHA3: 47aec1a4f1f0425b33658faea73f73943023c6b094609aaa425e1190e1f2873e File: ./src/ModularToken.sol SHA3: a74318bebe1ef3ffeb5f1af4431e82ddd7142695409de4cac9d7a3d0138c11c3 File: ./src/modules/BalanceSheetModule.sol SHA3: aa52aa7c02a891c5533c207f84261165fcb2afb10f964a268b1aa03a15314675 File: ./src/modules/BridgeModule.sol SHA3: 041f0386d5a8ce5a0d3d5070322b2f8b66fb42bbb40cf9c9e027e85e21da7066 File: ./src/modules/CDPModule.sol SHA3: 6ec57c98684f05b27db271947e6415f0b61bb3fd51b15e5b6d34dae98e725d81 File: ./src/modules/FlashMintModule.sol SHA3: 6adad18e6ec566471812bf1667ade6b1a5e784184438e2abaaa30b5f07d7b9aa File: ./src/modules/PegStabilityModule.sol SHA3: 9f668b37a74538e0a67100e8a88787a367ec9109dc9c017cc9355d3ec3e95370

	File: ./src/modules/templates/GenericModule.sol SHA3: a7d81eb85af30b96c6d951cde5e6b6d7e93471f71d796a95b5af5d550b19dbd9 File: ./src/modules/templates/Module.sol SHA3: 7f3bb3f105866638250b26462178de7fa7c3a7f902802078bffc5d6b6515d621 File: ./src/modules/templates/YSSModule.sol SHA3: 9d06c6d3fd814a42704d25a7cc2d2515f26ccfe77386da7502f4ba0b5ae25fe3 File: ./src/modules/templates/YSSModuleExtended.sol SHA3: 3a499f2032134c973fdadc21347c3d672adab3cb8dd3a9666d250f9b9e326b11 File: ./src/periphery/DutchAuctionLiquidator.sol SHA3: e0ea811f041470676f3f2b42800e832d029da5ca21c1e2d331b01f4dba82ad6e File: ./src/periphery/PSMLockup.sol SHA3: 773ddefd01633f82314b05e16cadd271e747b7ff9e13ff3b5b1e575847151db2 File: ./src/periphery/SimpleBSH.sol SHA3: 4af952411b3bf3574aa79c5c203f0efd817d849feb14986418585dd04ef9055f File: ./src/utility/ISwapper.sol SHA3: b17f978bbd1eea759c0636e06e1973d63d2299b886a30558c16498500efbbe47 File: ./src/utility/LeverageProxy.sol SHA3: 924cf37ef22ade3fe6e39db000330bcf9fef5f648a61c5125c26ffcb33f13adb File: ./src/YSS.sol SHA3: c6e419486ca0954b171d8ca5a077fc8bb5f6365c1cac8e382fd9815f32e17275

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Second review scope

Repository	https://github.com/Yama-Finance/stablecoin-contracts
Commit	9091b8553ad4b81cd3281a1654afc0767a1a39f1
Whitepaper	
Functional Requirements	https://beamish-taffy-de2c16.netlify.app/technical-summary
Technical Requirements	https://beamish-taffy-de2c16.netlify.app/technical-summary https://github.com/Yama-Finance/stablecoin-contracts/blob/main/README.md
Contracts	File: ./src/interfaces/IBalanceSheetHandler.sol SHA3: 8ed32d5eb4f82609ec6cac39b95cf16e82e826ee82c41453fb1cd24f437c5907 File: ./src/interfaces/IBridgeReceiver.sol SHA3: 4dab4f82e52fd8812e826645430ba30681b61c40d08a64c49c1979ea408b7788 File: ./src/interfaces/ICollateralManager.sol SHA3: 0e26fb5442b8ba0d440a235b587b251a3ec9815915a99d3181755f01214b2952 File: ./src/interfaces/ILiquidator.sol SHA3: ecce71df49bc709d3490fec76bb408517dc16869510e8c0ae8723e31200634f3 File: ./src/interfaces/IPriceSource.sol SHA3: f83ecd8bfc7b5697ac2d56344764adf7e7ed35b2114773fa059dd6fa00e89e72 File: ./src/ModularToken.sol SHA3: 02eb09bd01a7676111404718d536fd3a1b0d833ab0f9e2a849a1840e10939000 File: ./src/modules/BalanceSheetModule.sol SHA3: 723133cb20f09069eb5d48694a4b2876ef45d5c0fa69ff3d9c7df43e5eb66624 File: ./src/modules/BridgeModule.sol SHA3: 92ae4f4d43c205d82a21b020d0bc88e2a27048c967bff7cae0c4d04f1651f0c1 File: ./src/modules/CDPModule.sol SHA3: 9a36f88f16f5585fc375ef67ca48c8c68b60ed329e9aa54c40df7b5f2c7c5852 File: ./src/modules/FlashMintModule.sol SHA3: c33292fa0c48b94aa03dd62efd10156ad761aadd1f9f268dd72f121ace7c5f25 File: ./src/modules/PegStabilityModule.sol SHA3: 8511f1696bf15d24bb827cbd64850e68f99c215e2f0fd755381b0b5ba2874fc9 File: ./src/modules/templates/GenericModule.sol

	SHA3: 166009541b255742347180ead4a5411a91fc2e793e0c980146827cda219cba71 File: ./src/modules/templates/Module.sol SHA3: e12b2a2518cc9675d51f7cb5ce591277891be9032b587904aa2b70f1a3fa051d File: ./src/modules/templates/YSSModule.sol SHA3: dbcf2c25c0cb4a6c3d56229caabbdec1a9f3d24b1d5ab941d279a498b4ca7449 File: ./src/modules/templates/YSSModuleExtended.sol SHA3: 96e04b2357387748a4f7716468673354d380afa84d1f370154a54e56dbad212c File: ./src/periphery/DutchAuctionLiquidator.sol SHA3: 4a7130bba5307972a7ec085ab12ff7518ff4d68d0db25ba3a148245980354837 File: ./src/periphery/PSMLockup.sol SHA3: eed029b17d3dcddaeba1695cb3268191f97c98c1049bfb7d2b3781d019708ab0 File: ./src/periphery/PSMPriceSource.sol SHA3: bb1df6bee9773fcc598be602a9c751e98cabf25bcc1cf6cf53b37b6c7750c9a5 File: ./src/periphery/SimpleBSH.sol SHA3: 12e83370c896b77bab5460fd9a082a0f4c828e24586a1b8bfd2aff4c916b840c File: ./src/utility/ISwapper.sol SHA3: 57fa2e62fe86a57bb9ad832a29e2f29c45c1d9fb789583c232542b381b884c52 File: ./src/utility/LeverageProxy.sol SHA3: a4e1aab085ada028a58135422a3d9804cd634c1d8b59b5fedef0bec544846640 File: ./src/YSS.sol SHA3: 412c377a42c4b105e2c5f4194c22004704ea4942dd9c0c113305f7c0bd5e73d1

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor gas optimization. These issues won't have a significant impact on code execution but affect code quality

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are provided.
- Technical description is provided.

Code quality

The total Code Quality score is **10** out of **10**.

- The development environment is configured.
- Code follows Solidity guidelines.

Test coverage

Code coverage of the project is **100%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Negative cases are covered.
- Interactions with several users are tested.

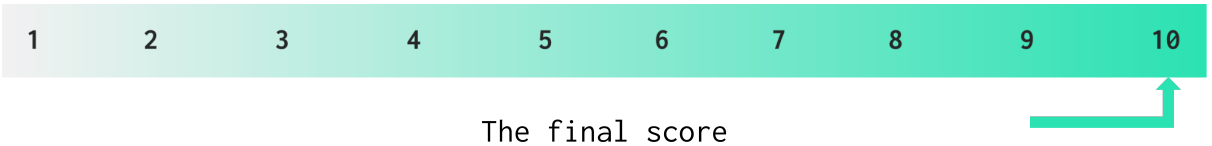
Security score

As a result of the audit, the code contains **no** issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **10.0**.



The final score

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
16 February 2023	8	4	7	1
08 March 2023	0	0	0	0

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Passed
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Not Relevant
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Passed
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Not Relevant
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP Standards Violation	EIP	EIP standards should not be violated.	Passed
Assets Integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Passed
Token Supply Manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style Guide Violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

System Overview

Yama Finance is an omnichain CDP protocol. It is a system of smart contracts that work together to maintain the health of the Yama stablecoin. The Yama Finance protocol is designed so that users can create a CDP, use it to generate the stablecoin, and then use the stablecoin to purchase goods and services, or to pay for fees. The protocol is also designed to be extendable, so governance can build additional modules on top of it. This allows for the protocol to be adaptable in the long-term.
The files in the scope:

- CDPModule.sol - is a contract which manages the creation and maintenance of collateralized debt pools (CDPs, also called vaults). Users can borrow additional funds, repay the loan, add collateral, or remove collateral from their vaults as long as the vaults do not become undercollateralized. The CDP module provides users with an easy way to borrow the Yama stablecoin by using their ETH, staked GLP, or other supported tokens as collateral. This allows users to leverage their crypto holdings to gain access to additional liquidity.
- BridgeModule.sol - is a contract which allows users to move funds between different blockchains. This allows users to move funds from one chain to another, for example Arbitrum to Fuel. This is a useful feature for users that want to transfer funds from one chain to another, or for users that want to take advantage of arbitrage opportunities between different blockchains.
- DutchAuctionLiquidator.sol - is a contract that handles collateral auctions.
- LeverageProxy.sol - opens a CDP vault on behalf of the user when LeverageProxy.createVault() is called. It acts as a tool to automate the process of leveraging up. Anyone can deploy their own version of the Leverage Proxy and use it to manage their vaults.
- PSMLockup.sol - incentivizes locking up money in the PSM. Locked tokens accrue value over time as the PSM lockup gets revenue from the protocol.
- PegStabilityModule.sol - converts YSS to/from an external stablecoin to stabilize the peg
- FlashMintModule.sol - allows anyone to mint YSS up to a limit with the one condition that they pay it all back in the same transaction. It’s used in the LeverageProxy.sol to simplify the process.
- ModularToken.sol - an ERC20 token that is the core of the protocol.
- SimpleBSH.sol - responsible for transferring the revenue to the PSM lockup if the protocol’s total surplus is not negative.
- BalanceSheetModule.sol - is a contract which keeps track of the protocol's deficit/surplus. Whenever a CDP is liquidated, the Balance Sheet module will update the protocol's deficit/surplus accordingly.

- YSSModuleExtended.sol - an abstract contract for implementing allow list functionality and the capacity to update BalanceSheetModule.sol contract address for CDPModule.sol, DutchAuctionLiquidator.sol, PSMLockup.sol and SimpleBSH.sol contracts.
- Module.sol - converts an amount between tokens with different decimal places.
- GenericModule.sol - an abstract contract for implementing allow list functionality for the BridgeModule.sol contract.
- YSSModule.sol - an abstract contract for implementing allow list functionality for BalanceSheetModule.sol, FlashMintModule.sol, PegStabilityModule.sol and LeverageProxy.sol contracts.
- ISwapper.sol - interface for swapper.
- ICollateralManager.sol - interface for CollateralManager.
- IBridgeReceiver.sol - interface for BridgeReceiver.
- YSS.sol - YSS stablecoin.
- IBalanceSheetHandler.sol - interface for the BalanceSheetHandler.sol contract.
- ILiquidator.sol - interface for the liquidator contract.
- IPriceSource.sol - interface for the PriceSource.

Privileged roles

- General:
  allowlist - An address from the allowlist of the Yama stablecoin has the following permissions:
  - YSS.sol (ModularToken): mint, burn and approve to/from any address and add/remove addresses to/from the allowlist
  - CDPModule.sol: sets the liquidators for this module, enable/disable borrowing, sets an allowed borrower for a vault, to set the collateral manager, add/update a collateral type, transfer any token from the CDPModule.sol contract and can update the BalanceSheet contract for the CDPModule.sol.
  - BalanceSheetModule.sol: can set the handler and protocol surplus, can add protocol surplus, protocol deficit.
  - BridgeModule.sol: can set the decimals of another chain, the address of a bridge contract on another chain, an alternate remote bridge address to accept messages from and change/update hyperlane parameters
  - FlashMintModule.sol: can set the maximum flash loan amount
  - PegStabilityModule.sol: can transfer any token from the PegStabilityModule.sol contract and set the debt ceiling
  - DutchAuctionLiquidator.sol: can set the liquidation parameters for a collateral type, set/update the default liquidation parameters, liquidate a vault of the CDPModule.sol and can update the BalanceSheet contract.

    - PSMLockup.sol: can update the BalanceSheet contract.
    - SimpleBSH.sol: can set the revenue share and can update the BalanceSheet contract.
- Additional:
    - BridgeModule.sol:
        - mailbox - handles cross-chain transfers.
    - CDPModule.sol:
        - vault owner/altOwner - can borrow from a vault, can repay to a vault and can add/remove collateral from a vault
    - SimpleBSH.sol:
        - balanceSheet - can call the method onAddSurplus()
    - LeverageProxy.sol:
        - flashMintModule - can call the method onFlashLoan()

Risks

- The system has a Coarse-grained Authorization Model. Most of the functionality is **accessible by any contract/EOAs from the allow list**. If any allowed list address is **compromised** this will lead to **full protocol control loss**.
- The system is **fully centralized**. Addresses from the allow list have ambiguous privileges.
- The system implements **bridge functionality** with help from the **Hyperlane** protocol. It's an **out of scope** project and its stability and **safety can’t be guaranteed**.
- Part of the protocol contracts are **out of scope** and their stability and **safety can’t be guaranteed**.

Findings

■■■■ Critical

C01. Coarse-grained Authorization Model

The main functionality is accessible by any contract/EOAs from the YSS stablecoin allow list.

A project should have a fine-grained access control system if it has multiple layers of auth-related functionality.

In case an address from the allow list is compromised it will put the whole system at risk.

Path: ./src/*.sol

Recommendation: Implement an access control system with restricted access for protocol contracts and functions.

Status: Mitigated (This is a design decision. The EOA deploying the contracts will be removed from the allowlist.)

■■■ High

H01. Undocumented Behavior; Highly Permissive Role Access

YSSModuleExtended is inherited by CDPModule.sol, DutchAuctionLiquidator.sol, PSMLockup.sol, SimpleBSH.sol. According to the documentation “The Balance Sheet module is a contract which keeps track of the protocol's deficit/surplus”. YSSModuleExtended.sol implements a setter for the Balance Sheet module which will allow whitelisted EOAs/contracts to change it at any time.

This behavior is not documented and can lead to incorrect counting of protocol deficit/surplus.

Path: /src/modules/templates/YSSModuleExtended.sol : setBalanceSheet()

Recommendation: remove the setter for the Balance Sheet module for the mentioned contract or/and add public documentation with detailed explanation why this is needed.

Status: Mitigated (This functionality has been implemented so that the protocol can upgrade in the future.)

H02. Data Consistency

Decimals of the external stablecoin and YAMA of the PegStabilityModule.sol contract are constructor parameters.

In the constructor of the BridgeModule.sol contract token decimals is a constructor parameter.

This can lead to incorrect setup of the decimals and as a result to invalid calculation of token conversion.

Path: /src/modules/PegStabilityModule.sol : constructor()

/src/modules/BridgeModule.sol : constructor()

Recommendation: set decimals by calling corresponding methods on token contracts to avoid the possibility of an incorrect setup.

Status: Mitigated (This has been implemented to limit gas consumption (an external call to decimals() would be needed each time), as it is assumed decimals will not change.)

H03. Undocumented Behavior; Highly Permissive Role Access

The function *transfer* of the PegStabilityModule.sol and CDPModule.sol contracts allows the whitelisted EOAs/contracts to transfer any token from the contract to the chosen address including external stablecoins deposited by users or collateral tokens.

This behavior is not documented and can lead to an inability to withdraw deposited funds by the user.

Path: /src/modules/PegStabilityModule.sol : transfer()

/src/modules/CDPModule.sol : transfer()

Recommendation: remove the mentioned method or add a check that the token to transfer is not one of the protocol tokens to cover cases when an alien token was transferred directly to the contract by mistake.

Status: Mitigated (This is used by DutchAuctionLiquidator and by governance in case users' funds are locked.)

H04. Undocumented Behavior; Highly Permissive Role Access

The Balance Sheet module is a contract which keeps track of the protocol's deficit/surplus. The method *setSurplus* allows whitelisted EOAs/contracts to specify the totalSurplus despite the actual state.

Additionally, the methods *addSurplus* and *addDeficit* can be called not by intended contracts but by EOAs with allowlist privilege roles.

This functionality is undocumented.

Path: /src/modules/BalanceSheetModule.sol : setSurplus(), addSurplus(), addDeficit()

Recommendation: remove the possibility of direct updates of the totalSurplus.

Status: Mitigated (The EOA deploying the contracts will be removed from the allowlist.)

H05. Undocumented Behavior; Highly Permissive Role Access

EOAs/contracts from the allow list have the possibility to change initial Hyperlane parameters at any time.

This functionality is undocumented.

Path: /src/modules/BridgeModule.sol : setHyperlaneParameters()

Recommendation: remove the possibility to change initial hyperlane parameters or/and add public documentation with detailed explanation why this is needed.

Status: Mitigated (Contracts on the allowlist already require root-level access and thus there is no security drawback from letting them adjust the Hyperlane parameters.)

H06. Token Supply Manipulation; Highly permissive role

Addresses from allowlist can *mint*, *burn* and *approve* any amount of the YSS stablecoin from/to any address.

Despite the fact that this functionality is used by the BridgeModule.sol and FlashMintModule.sol contracts it can be implemented with actual user allowance.

Path: /src/ModularToken.sol : mint(), burn(), approve()

Recommendation: implement an access control system with restricted access for protocol contracts and functions and use actual user allowance for BridgeModule.sol and FlashMintModule.sol.

Status: Mitigated (The EOA deploying the contracts will be removed from the allowlist.)

H07. Non-Finalized Code; Undocumented Behavior

The ISwapper.sol interface is used in SimpleLiquidator.sol, LeverageProxy.sol. Functions *swapToYama*, *swapToCollateral* are not implemented.

The ICollateralManager.sol interface is used in CDPModule, their functionality is not implemented. Functions *handleCollateralDeposit*, *handleCollateralWithdrawal* are not implemented.

The function *onAddDeficit* of the SimpleBSH does not contain logic.

The code should not contain undocumented functionality.

Path: /src/periphery/SimpleBSH.sol : onAddDeficit()

/src/modules/CDPModule.sol : handleCollateralDeposit(), handleCollateralWithdrawal()

/src/utility/LeverageProxy.sol : swapToYama(), swapToCollateral()

Recommendation: add documentation for mentioned functionalities and finalize the code.

Status: Mitigated (At launch, src/periphery/EmptyCollateralManager.sol will be used as the default collateral manager. It will have no functionality, but in the future, if governance desires to perform an action with deposited collateral (such as staking it on another platform), the collateral manager makes it possible to implement this functionality without replacing CDPModule.sol)

Medium

M01. Inefficient Gas model - Redundant interactions

The ModularToken.sol contract in the constructor adds a contract creator to the allowlist two times.

Path: ./src/ModularToken.sol : constructor()

Recommendation: remove redundant operations

Status: Mitigated (This is intentional. If the allowlist mapping is not set before setAllowlist() is called by the constructor, setAllowlist() will revert because msg.sender is not already on the allowlist. And if the mapping is set manually instead of calling setAllowlist(), the event will not be emitted.)

M02. Unchecked transfer or approve

The function does not use the SafeERC20 library to check the result of ERC20 token transfers and approvals. Tokens may not follow the ERC20 standard and return false in case of transfer failure or not returning any value at all.

Path: /src/utility/LeverageProxy.sol : createVault(), onFlashLoan()

/src/periphery/PSMLockup.sol : lockup(), withdraw()

/src/periphery/DutchAuctionLiquidator.sol : claim()

/src/modules/PegStabilityModule.sol : transfer(), deposit(), withdraw()

/src/modules/FlashMintModule.sol : flashLoan()

/src/modules/CDPModule.sol : addCollateral(), removeCollateral(), transfer()

/src/modules/BridgeModule.sol : transferFromRemote()

Recommendation: use the SafeERC20 library to interact with tokens safely.

Status: Fixed (Revised commit: 9091b8553ad4b81cd3281a1654afc0767a1a39f1)

M03. Inefficient Gas model - Redundant interactions

In the current implementation of the function *deposit* check, the total balance of the PegStabilityModule contract not exceeding debtCeiling is performed after external calls and calculation. Moving it to the beginning can save Gas for users.

Path: /src/modules/PegStabilityModule.sol : deposit()

Recommendation: create local variable to calculate the expected *externalStablecoin* balance on new deposits and move the require statement before external calls and further calculations. Additionally, in case the function *transfer* is disallowed for the externalStablecoin implement the actual total deposited amount storage variable to avoid risk of direct transfers to the contract which can cause a Denial of Service of the deposit method and funds lock.

Status: Mitigated (Checking the balance after the deposit is intentional, so that the PSM can't cumulatively mint more than debtCeiling amount of the stablecoin.)

M04. Inconsistent data - Variable is not limited

According to the documentation - “At launch, SimpleBSH is the handler and it will transfer 90% of the revenue to the PSM lockup if the protocol’s total surplus is not negative.”. Consider limiting the revenueShare value in order to prevent a Denial of Service in case of wrong setup.

Path: /src/periphery/SimpleBSH.sol : setRevenueShare()

Recommendation: provide conscious limits for stored configuration values.

Status: Fixed (Revised commit: 9091b8553ad4b81cd3281a1654afc0767a1a39f1)

Low

L01. Floating Pragma

Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Path: ./src/

Recommendation: consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Status: Fixed (Revised commit: 9091b8553ad4b81cd3281a1654afc0767a1a39f1)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

L02. Functions that can be declared external

"public" functions that are never called by the contract should be declared "external" to save gas.

Path: /src/periphery/DutchAuctionLiquidator.sol : getCollateralAmount(), getLastAuctionId()

/src/modules/CDPModule.sol : borrow(), repay(), addCollateral(), removeCollateral(), setLiquidators(), setBorrowingDisabled(), isLiquidated(), getAnnualInterest(), getPsInterest(), getCollateralRatio(), getDebtFloor(), getDebtCeiling(), getOwnedVaults(), getOwner(), getAltOwner(), getCollateralTypeId(), getCollateralToken()

/src/modules/BridgeModule.sol : transferFromRemote(), setHyperlaneParameters()

Recommendation: use the external attribute for functions never called from the contract.

Status: Fixed (Revised commit: 9091b8553ad4b81cd3281a1654afc0767a1a39f1)

L03. State variables default visibility

The visibility of the variable *target* is not specified. Specifying state variables visibility helps to catch incorrect assumptions about who can access the variable.

This improves the contract`s code quality and readability.

Path: /src/periphery/SimpleBSH.sol

Recommendation: specify variables as public, internal, or private. Explicitly define visibility for all state variables.

Status: Fixed (Revised commit: 9091b8553ad4b81cd3281a1654afc0767a1a39f1)

L04. State variables can be declared immutable

Compared to regular state variables, the gas costs of constant and immutable variables are much lower. Immutable variables are evaluated once at construction time and their value is copied to all the places in the code where they are accessed.

Variables *psm*, *psmToken* are set in the constructor of the PSMLockup.sol contract.

Variables *flashMintModule*, *cdpModule* are set in the constructor of the LeverageProxy.sol contract.

The variable *target* is set in the constructor of the SimpleBSH.sol contract.

The variable *cdpModule* is set in the constructor of the DutchAuctionLiquidator.sol contract.

Variables *externalStablecoin*, *yssDecimals*, *externalDecimals* are set in the constructor of the PegStabilityModule.sol contract.

Those variables can be declared immutable.

This will lower the Gas taxes.

Path: /src/periphery/PSMLockup.sol

/src/utility/LeverageProxy.sol

/src/periphery/SimpleBSH.sol

/src/periphery/DutchAuctionLiquidator.sol

/src/modules/PegStabilityModule.sol

Recommendation: declare mentioned variables as immutable.

Status: Fixed (Revised commit: 9091b8553ad4b81cd3281a1654afc0767a1a39f1)

L05. Unused modifier

The modifier *onlyVaultOwner* is declared, but never used.

This leaves redundant logic in code.

This will lower Gas taxes.

Path: /src/utility/LeverageProxy.sol : onlyVaultOwner()

Recommendation: remove redundant code.

Status: Fixed (Revised commit: 9091b8553ad4b81cd3281a1654afc0767a1a39f1)

L06. Missed internal/external imports

The LeverageProxy.sol contract inherits IERC3156FlashBorrower, YSSModule but those imports are not present in the contract.

This will increase code readability.

Path: /src/utility/LeverageProxy.sol

Recommendation: add missing imports.

Status: Fixed (Revised commit: 9091b8553ad4b81cd3281a1654afc0767a1a39f1)

L07. Redundant variable conversion

The arguments of the *addSurplus* method call are of uint256 type and converted to int256 before subtraction. This increases Gas cost.

Path: ./src/modules/CDPModule.sol:updateInterest()

Recommendation: Make conversion of variables over the result of the subtraction.

Status: Mitigated (This is intentional so that it is possible for governance to set negative interest)

L08. Style Guide Violation

The provided projects should follow the official guidelines.

Inside each contract, library or interface, use the following order:

1. Type declarations
2. State variables
3. Events
4. Modifiers
5. Functions

Functions should be grouped according to their visibility and ordered:

1. constructor
2. receive function (if exists)
3. fallback function (if exists)
4. external
5. public
6. internal
7. private

Within a grouping, place the view and pure functions last.

It's best practice to cover all functions with NatSpec annotation and to follow the Solidity naming convention. This will increase overall code quality and readability.

Path: /src/ModularToken.sol

/src/utility/LeverageProxy.sol

/src/periphery/DutchAuctionLiquidator.sol

/src/modules/PegStabilityModule.sol

/src/modules/CDPModule.sol

/src/modules/BridgeModule.sol

Recommendation: follow the official Solidity guidelines.

Status: Fixed (Revised commit: 9091b8553ad4b81cd3281a1654afc0767a1a39f1)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.