HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Warped Games
Date: 07 July, 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 Warped Games
Approved By	Noah Jelich | Lead Solidity SC Auditor at Hacken OU
Tags	ERC20 token
Platform	EVM
Language	Solidity
Methodology	Link
Website	http://warped.games/
Changelog	21.06.2023 - Initial Review 07.07.2023 - Second Review

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

Table of contents

Introduction **4**
System Overview **4**
Executive Summary **7**
Checked Items **8**
Findings **11**
Critical 11
C01. Denial of Service 11
High 11
H01. Front Running 11
H02. Denial Of Service 12
Medium 12
M01. Data Consistency 12
M02. Data Consistency 12
M03. Requirement Violation - Data Consistency 13
M04. Missing Event 13
M05. Unchecked Transfer 14
M06. Modification Of Well Known Contract 14
Low 14
L01. Missing Zero Address Validation 14
L02. Empty Constructor 15
L03. State Variable That Should be Constant 15
L04. Inefficient Gas Model 15
Informational 16
I01. Floating Pragma 16
I02. Incorrect Contract Description 16
I03. Redundant Code 16
Disclaimers **17**
Appendix 1. Severity Definitions **18**
Risk Levels 18
Impact Levels 19
Likelihood Levels 19
Informational 19
Appendix 2. Scope **20**

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

Introduction

Hacken OÜ (Consultant) was contracted by Warped Games (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.

System Overview

Warped Games is a ERC-20 token protocol with the following contracts:

- *WarpedPoolManager* – is a contract that enables the management of a pool list. The contract allows for adding and removing pools, as well as setting and updating the primary pool (during setting or updating, the pool should be added to the pools list). Additionally, the contract provides a function to verify if a given address is present in the pool list.
- *WarpedTaxHandler* – is a smart contract that provides tax management for token transfers. This contract calculates tax on transactions based on the level of the user, determined by their ownership of NFTs (Non-Fungible Tokens). The contract includes the following main features:
    - Tax Management: The contract manages taxes on transactions. The tax rate is determined by the user's level, which is associated with their ownership of non-fungible tokens (NFTs).
    - NFT Ownership-based Levels: The contract establishes different levels of users based on their ownership of certain NFTs. The level assigned to a user affects the tax they need to pay on transactions.
    - Tax Calculation: The contract calculates the amount of tax a user has to pay on a transaction. Tax is only charged on transactions involving the exchange pool, not between regular users or between different pools.
    - Custom Tax Rates: The contract allows for setting custom tax rates for different levels of users.
    - Add/Remove NFTs: The contract includes functionality for adding and removing NFTs. This helps in managing which NFTs are considered when determining a user's level.
    - Tax Pause and Resume: The contract includes functions to disable and enable tax calculations. This could be useful during special events or certain time periods.
- *WarpedTreasuryHandler* – is a smart contract that serves as a treasury handler, primarily designed to manage tokens accumulated through a mechanism like transaction tax. On every token transaction, it's notified to handle the tax part of the transferred tokens. If the tokens collected in the contract exceed a certain threshold, the contract then sells a portion

of the tokens for ETH, considering a limit to avoid severe price impact. A part of the tokens are used to add liquidity to a liquidity pool. The ETH obtained from selling tokens and the remaining ETH after adding liquidity are then sent to the treasury address.
This contract also provides functionalities to adjust key parameters like the percentage of tokens that should be added as liquidity (liquidityBasisPoints) and maximum price impact allowed for a sell (priceImpactBasisPoints).
Additionally, it allows the owner to change the treasury address, withdraw stuck tokens or ETH, and modify the threshold limit (_taxSwap) which triggers the token selling and liquidity addition process.
This contract essentially manages the accumulation and distribution of transaction taxes, helping maintain liquidity and stabilize the token price.

- *WarpedToken* – is a contract that extends the standard ERC20 token contract, adding a taxation and treasury handling system, which are managed by two external contracts referred to as the Tax Handler and the Treasury Handler. It also has a feature to prevent re-entrancy attacks using the LenientReentrancyGuard.
    - Initialization: When the WARPED token contract is deployed, it is initialized with the address of the deployer, the Tax Handler contract, and the Treasury Handler contract. All the initial tokens (10 billion in total) are minted to the deployer's address.
    - Token Transfers: Every time a token transfer occurs, two additional steps are taken:
      Before the transfer, the contract calls the processTreasury function of the Treasury Handler contract, allowing it to execute any necessary logic before the token transfer.
      After the transfer, the contract calculates the tax amount to be deducted using the getTax function of the Tax Handler contract, and the tax is transferred to the address of the Treasury Handler contract.
    - Updating Handlers: The contract owner can update the address of the Tax Handler or the Treasury Handler at any time using the updateTaxHandler and updateTreasuryHandler functions respectively. This provides flexibility to replace the handling logic as needed.
- *WarpedTokenManager* – is a contract responsible for managing the WARPED token, including its creation and the addition of liquidity to Uniswap. The contract uses the WarpedPoolManager, which seems to manage a set of exchange pools for the WARPED token.

Here is a brief overview of how it works:
Initialization: During the initialization of the WarpedTokenManager contract, the contract creates a WarpedTreasuryHandler, a WarpedTaxHandler, and the WarpedToken itself. The addresses of these newly created contracts are stored and their ownership is transferred to the deployer of the WarpedTokenManager contract. The WarpedTreasuryHandler is also initialized with the provided treasury address and the address of the WarpedToken.
Adding Liquidity: There is a function, addLiquidity, that can only be called by the owner of the contract. This function allows the owner to add liquidity to the Uniswap pool for the WARPED token. In this process, the contract:
Receives tokens from the deployer's wallet, approves the Uniswap Router to use the tokens, creates a Uniswap pair with the WARPED token and WETH (Wrapped Ether), adds liquidity to the pair, stores the Uniswap pair address as an exchange pool and sets it as the primary pool.

Privileged roles

- The owner of the *WarpedPoolManager* contract can add, delete pool addresses and set, update a primary pool address.
- The owner of the *WarpedTaxHandler* contract can set tax rates, add and remove nfts, pause and unpause tax.
- The owner of the *WarpedToken* contract can update tax handler and treasury handler addresses.
- The owner of the *WarpedTreasuryHandler* contract can initialize the contract, set liquidity basis points, set price impact basis points, set treasury address, withdraw coins and ERC-20 tokens from contract, update tax swap value.

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 complete
- Technical Description is complete
- NatSpecs are satisfactory.
- Readme contains the needed information.

Code quality

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

- The code quality is very good.
- The development environment is configured.

Test coverage

Code coverage of the project is **100%** (branch coverage), with a mutation score of **73,3%.**

- Deployment and basic user interactions are covered with tests.
- Negative cases coverage is complete.

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**.

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
21 June 2023	4	6	2	1
7 July 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	Description	Status	Related Issues
Default Visibility	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed	
Integer Overflow and Underflow	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant	
Outdated Compiler Version	It is recommended to use a recent version of the Solidity compiler.	Passed	
Floating Pragma	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed	
Unchecked Call Return Value	The return value of a message call should be checked.	Passed	
Access Control & Authorization	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	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant	
Check-Effect-Interaction	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed	
Assert Violation	Properly functioning code should never reach a failing assert statement.	Passed	
Deprecated Solidity Functions	Deprecated built-in functions should never be used.	Passed	
Delegatecall to Untrusted Callee	Delegatecalls should only be allowed to trusted addresses.	Not Relevant	
DoS (Denial of Service)	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	Race Conditions and Transactions Order Dependency should not be possible.	Passed	
Authorization through tx.origin	tx.origin should not be used for authorization.	Passed	
Block values as a proxy for time	Block numbers should not be used for time calculations.	Passed	
Signature Unique Id	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	State variables should not be shadowed.	Passed	
Weak Sources of Randomness	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant	
Incorrect Inheritance Order	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	All external calls should be performed only to trusted addresses.	Passed	
Presence of Unused Variables	The code should not contain unused variables if this is not justified by design.	Passed	
EIP Standards Violation	EIP standards should not be violated.	Passed	
Assets Integrity	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed	
User Balances Manipulation	Contract owners or any other third party should not be able to access funds belonging to users.	Passed	
Data Consistency	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	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. Contracts shouldn't rely on values that can be changed in the same transaction.	Passed	
Token Supply Manipulation	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	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	Style guides and best practices should be followed.	Passed	
Requirements Compliance	The code should be compliant with the requirements provided by the Customer.	Passed	
Environment Consistency	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	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	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	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

Findings

■■■■ Critical

C01. Denial of Service

Impact	High
Likelihood	High

In the function removeNFTs(), the NFTs are removed from the array using the delete keyword. Therefore, the zero address will be present in the array instead of the removed one.

This will lead to the balanceOf call to zero address in _getTaxBasisPoints() which will fail and make the function inoperable.

Path: ./contracts/WarpedTaxHandler.sol : removeNFTs()

Recommendation: do not leave gaps when removing the values from the arrays.

Found in: 168ad70

Status: Fixed (9b59231)

■■■ High

H01. Front Running

Impact	High
Likelihood	Medium

The amount out values are set to 0 when interacting with Uniswap.

This allows attackers to perform front running attacks and the operations may result in unexpected amounts of out tokens.

Path: ./contracts/WarpedTokenManager.sol : addLiquidity()

./contracts/WarpedTreasuryHandler.sol : _addLiquidity(), _swapTokensForEth()

Recommendation: specify the amount out values when interacting with Uniswap.

Found in: 168ad70

Status: Fixed (9b59231)

H02. Denial Of Service

Impact	High
Likelihood	Medium

The _getTaxBasisPoints function loops over all the NFTs and tax rates.

In case the arrays are large enough to exceed the block Gas limit, the execution may fail.

Path: ./contracts/WarpedTokenManager.sol : _getTaxBasisPoints()

Recommendation: do not rely on the arrays` lengths.

Found in: 168ad70

Status: Fixed (9b59231)

Medium

M01. Data Consistency

Impact	Medium
Likelihood	Medium

The primary pool address may be removed from the _exchangePools and be not updated in the primaryPool variable.

This will result in inconsistent contract state.

Path: ./contracts/WarpedPoolManager.sol

Recommendation: ensure that the primaryPool and _exchangePools values are consistent.

Found in: 168ad70

Status: Fixed (9b59231)

M02. Data Consistency

Impact	Low
Likelihood	Medium

The NFTs are not being checked for the uniqueness when they are added.

This may result in duplicates and inconsistent contract state.

Path: ./contracts/WarpedTaxHandler.sol : addNFTs()

Recommendation: check if the NFTs addresses are unique.

Found in: 168ad70

Status: Fixed (9b59231)

M03. Requirement Violation - Data Consistency

Impact	Medium
Likelihood	Medium

In the function setTaxRates(), the requirement says :

- values of `thresholds` must be placed in ascending order.

There is no check done to verify this requirement.

Path: ./contracts/WarpedTaxHandler.sol : setTaxRates()

Recommendation: ensure that the tax rates are stored in a correct order.

Found in: 168ad70

Status: Fixed (9b59231)

M04. Missing Event

Impact	Low
Likelihood	Medium

Events for critical state changes should be emitted for tracking things off-chain.

Path: ./contracts/WarpedTaxHandler.sol : setTaxRates(), addNFTs(), removeNFTs(), pauseTax(), resumeTax()

./contracts/WarpedToken.sol : updateTaxHandler(), updateTreasuryHandler()

./contracts/WarpedTreasuryHandler.sol : updateTaxSwap()

Recommendation: emit events for critical state changes.

Found in: 168ad70

Status: Fixed (9b59231)

M05. Unchecked Transfer

Impact	Medium
Likelihood	Medium

The function withdraw() does not use SafeERC20 library for checking the result of ERC20 token transfer.

Path: ./contracts/WarpedTreasuryHandler.sol : withdraw()

Recommendation: use SafeERC20 library to interact with tokens safely.

Found in: 168ad70

Status: Fixed (9b59231)

M06. Modification Of Well Known Contract

The *nonReentrant* modifier does not revert in case of reentrancy, but executes return.

Such behavior may be unexpected. Not reverting will break the key functionality of non-reentrancy.

Path: ./contracts/LenientReentrancyGuard.sol : nonReentrant()

Recommendation: do not modify the well known contract.

Found in: 168ad70

Status: Fixed (9b59231)

Low

L01. Missing Zero Address Validation

Impact	Medium
Likelihood	Low

Address parameters are being used without checking against the possibility of 0x0.

This can lead to unwanted external calls to 0x0.

Path: ./contracts/WarpedTreasuryHandler.sol : initialize()

./contracts/WarpedToken.sol : constructor()

Recommendation: implement zero address checks.

Found in: 168ad70

Status: Fixed (9b59231)

L02. Empty Constructor

Impact	Low
Likelihood	Medium

The constructor is empty.

Redundant code decreases the code readability.

Path: ./contracts/WarpedPoolManager.sol : constructor()

Recommendation: remove the redundant code.

Found in: 168ad70

Status: Fixed (9b59231)

L03. State Variable That Should be Constant

Impact	Low
Likelihood	Medium

State variables that do not change their value should be declared constant to save Gas.

Path: ./contracts/WarpedTaxHandler.sol : maxTaxRate

Recommendation: declare variable as constant.

Found in: 168ad70

Status: Fixed (9b59231)

L04. Inefficient Gas Model

Impact	Low
Likelihood	Medium

String variables are attributed with the keyword unicode. This is not necessary and will cost additional Gas.

Path: ./contracts/WarpedToken.sol : _NAME, _SYMBOL

Recommendation: remove the “unicode” keyword when it is not necessary.

Found in: 168ad70

Status: Fixed (9b59231)

Informational

I01. 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.

Paths: ./contracts/*

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

Found in: 168ad70

Status: Fixed (9b59231)

I02. Incorrect Contract Description

The WarpedPoolManager contract is described with the *"@title Exchange pool processor abstract contract."* comment.

However, it is not abstract.

Path: ./contracts/WarpedPoolManager.sol

Recommendation: align the documentation and the implementation.

Found in: 168ad70

Status: Fixed (9b59231)

I03. Redundant Code

The taxDisabled value is manually set to false, which is redundant.

Redundant code uses the extra Gas and decreases the code readability.

Path: ./contracts/WarpedTaxHandler.sol : constructor()

Recommendation: remove the redundant code.

Found in: 168ad70

Status: Fixed (9b59231)

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.

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

Appendix 1. Severity Definitions

When auditing smart contracts Hacken is using a risk-based approach that considers the potential impact of any vulnerabilities and the likelihood of them being exploited. The matrix of impact and likelihood is a commonly used tool in risk management to help assess and prioritize risks.

The impact of a vulnerability refers to the potential harm that could result if it were to be exploited. For smart contracts, this could include the loss of funds or assets, unauthorized access or control, or reputational damage.

The likelihood of a vulnerability being exploited is determined by considering the likelihood of an attack occurring, the level of skill or resources required to exploit the vulnerability, and the presence of any mitigating controls that could reduce the likelihood of exploitation.

Risk Level	High Impact	Medium Impact	Low Impact
High Likelihood	**Critical**	**High**	**Medium**
Medium Likelihood	**High**	**Medium**	**Low**
Low Likelihood	**Medium**	**Low**	**Low**

Risk Levels

Critical: Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.

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.

Medium: Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.

Low: Major deviations from best practices or major Gas inefficiency. These issues won't have a significant impact on code execution, don’t affect security score but can affect code quality score.

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

Impact Levels

High Impact: Risks that have a high impact are associated with financial losses, reputational damage, or major alterations to contract state. High impact issues typically involve invalid calculations, denial of service, token supply manipulation, and data consistency, but are not limited to those categories.

Medium Impact: Risks that have a medium impact could result in financial losses, reputational damage, or minor contract state manipulation. These risks can also be associated with undocumented behavior or violations of requirements.

Low Impact: Risks that have a low impact cannot lead to financial losses or state manipulation. These risks are typically related to unscalable functionality, contradictions, inconsistent data, or major violations of best practices.

Likelihood Levels

High Likelihood: Risks that have a high likelihood are those that are expected to occur frequently or are very likely to occur. These risks could be the result of known vulnerabilities or weaknesses in the contract, or could be the result of external factors such as attacks or exploits targeting similar contracts.

Medium Likelihood: Risks that have a medium likelihood are those that are possible but not as likely to occur as those in the high likelihood category. These risks could be the result of less severe vulnerabilities or weaknesses in the contract, or could be the result of less targeted attacks or exploits.

Low Likelihood: Risks that have a low likelihood are those that are unlikely to occur, but still possible. These risks could be the result of very specific or complex vulnerabilities or weaknesses in the contract, or could be the result of highly targeted attacks or exploits.

Informational

Informational issues are mostly connected to violations of best practices, typos in code, violations of code style, and dead or redundant code.

Informational issues are not affecting the score, but addressing them will be beneficial for the project.

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

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Initial review scope

Repository	https://github.com/warpedgames/contracts-warped-token
Commit	168ad70c69128527c1a8d15a3cd1e0109cd2790c
Whitepaper	
Requirements	
Technical Requirements	
Contracts	File: contracts/LenientReentrancyGuard.sol SHA3: d757ba8b8fef29988e0e6f1b2b83723a27c2d7fbc232e088eda123cb0ca11a76 File: contracts/WarpedPoolManager.sol SHA3: c7ba6a34394bb8e4cf3b176dad5167b40b929d22acd4616b1c38f0d99186d389 File: contracts/WarpedTaxHandler.sol SHA3: d8587c80d337a40d56929f58196b5c3d8e4476ab360d62508b873b6314076b12 File: contracts/WarpedToken.sol SHA3: 10a861d5e4eb4d46d75509bc1e3ce49d5cf3f20c03f0c5d5d14c798858663acc File: contracts/WarpedTokenManager.sol SHA3: 985ce3fb688fa80d0cd21bf8485491fb660eb169dfae3746b6c7d03c3bc14ad7 File: contracts/WarpedTreasuryHandler.sol SHA3: 2671cc6676bab750c8e0a285d72c79c0b0557219d3922144de27fa180fc81b8c File: contracts/interfaces/IPoolManager.sol SHA3: 9fca266e34c6d69fc0a2aa389192f774931d788e51b331e5f316e5accf4e5291 File: contracts/interfaces/ITaxHandler.sol SHA3: 9577f0a69632a38f2026558d99cb2feed0a70b89dae734ca67d26dbba322b07c File: contracts/interfaces/ITreasuryHandler.sol SHA3: bf10e6cb612e3a352b18106f16072e6e2755675c78eae242677491736567d4e8 File: contracts/interfaces/IUniswapV2Router02.sol SHA3: df997c8a54715ea63a7ad4c7162327b76061fff81d40afb8876c5ad253519e5e

Second review scope

Repository	https://github.com/warpedgames/contracts-warped-token
Commit	9b59231f2093f29d486d596c7891c311cb9ebe63
Whitepaper	

Requirements	https://github.com/warpedgames/contracts-warped-token/WARPED - Audit Techspec.pdf
Technical Requirements	https://github.com/warpedgames/contracts-warped-token/WARPED - Audit Techspec.pdf
Contracts	File: WarpedPoolManager.sol SHA3: ffb9657eca1c08fd4ba014b005ba38b26b27d0be5b4f7c412e04357c84abd4ac File: WarpedTaxHandler.sol SHA3: db17f51241df8f21d591753dd4bab2f0991a40355e69f82a71219a607e9b0d7a File: WarpedToken.sol SHA3: d404fa82b33689bac5705dab9a0c26cd8b915adaa793f5d6ee4dc71946f1d3b2 File: WarpedTokenManager.sol SHA3: fd96d78080e3d36faf552d1f3ad3de4c5ced7a5720bb6329d253cf89f9b106af File: WarpedTreasuryHandler.sol SHA3: 374160b0a25718f01c7b6c43a90bb7ca0f23a411f0ac09da680f8ad33aed6359 File: interfaces/IPoolManager.sol SHA3: 98bd4f8481d2547af269a1c744dfb6c2c7daca1a66ee3db71994425eacad89f9 File: interfaces/ITaxHandler.sol SHA3: 5f75f942182bbdb9d613be072026884ecaa2a93302f5d952973d02fd77ba6ad9 File: interfaces/ITreasuryHandler.sol SHA3: 42e88d8929ef57cdf7ba50601d5ecc8b3e5e376a68bf21bbfd4ed4fd71a70303 File: interfaces/IUniswapV2Router02.sol SHA3: f3b7349ebaf2c0a4afd04109cc09816f9da576a58d752f32ab5c394c099e6af3