



SOLIDProof
Bring trust into your projects

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY

IceCreamSwap Bridge Audit

**Security Assessment
13. June, 2023**

For



SolidProof_io



@solidproof_io

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	11
Risk Level	11
Capabilities	12
Inheritance Graph	13
CallGraph	14
Scope of Work/Verify Claims	15
Modifiers and public functions	18
Source Units in Scope	19
Critical issues	20
High issues	20
Medium issues	20
Low issues	21
Informational issues	23
Audit Comments	25
SWC Attacks	26

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing 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. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	12. June 2023	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Bitgert

Core

XDC

Binance smart chain (only bridge deployed for now)

Dogechain

Fuse

Website

<https://icecreamswap.com/?chainId=1116>

Telegram

https://t.me/Icecreamswap_com

Twitter

https://twitter.com/icecream_swap

Description

Trade, Earn, Bridge and Launch on CORE, XDC, Binance smart chain (BSC), Bitgert (Brise), Shardeum, Dogechain, Doken and Fuse with our decentralized smart contracts.

Project Engagement

During the 26th of May 2023, **IceCreamSwap Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo



Contract Link

v1.0

- Dex
 - <https://github.com/IceCreamSwapCom/IceCreamSwap-smart-contracts/tree/master/projects/bridge>
 - Commit: da446c3fee322e3d57d540d572f82a2a04daeb34

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
 - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.
2. Testing and automated analysis that includes the following:
 - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

-

Note:

- The core folder is a fork of the ChainSafe project. The only changes are the addition of the Migration Handler address and a calculate fee function.



Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

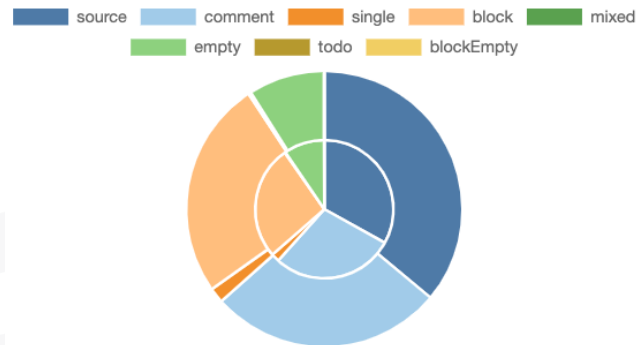
File Name	SHA-1 Hash
bridge/contracts/ERC20Safe.sol	bf01adcab464891140e2a b7551ae3b5351eff4d7
bridge/contracts/Bridge.sol	fa6ef4f5782a5335d360ae 2af6287211d4de9836
bridge/contracts/interfaces/ IGenericHandler.sol	a397bc3b99bde45c3a2ae 78b6f10bba497d1004e
bridge/contracts/utils/SafeMath.sol	5ad67690570012fcb1828 f2ceab27c23d8239c06
bridge/contracts/interfaces/ IERCHandler.sol	a4c5ccff64946c9d595cc8 65c30a9d5ebb4d0f8a
bridge/contracts/interfaces/IBridge.sol	76429df00b53dce69d07fc 1281d14bb36b4219c9
bridge/contracts/handlers/ HandlerHelpers.sol	77fd797ce8d6dd33bf767 21610373a01ec19a596
bridge/contracts/interfaces/ IDepositExecute.sol	4e2e60f565aa5a7377883 6aafa2a8888ff6ab101
bridge/contracts/utils/Pausable.sol	fe9995e9dddffc777b6a9 d2e104c021659189c32
bridge/contracts/utils/SafeCast.sol	4ec3c6777a4b68888d210 caf8defa1f872d5ae58
bridge/contracts/utils/AccessControl.sol	6109cfaaa4880efe1db7d 35d9d46ad27961d9e70

bridge/contracts/handlers/ERC20/ ERC20HandlerPercentageFee.sol	13128653dd8c7a37084ca f2ffa5077f4c41c43f
bridge/contracts/handlers/ERC20/ NativeToken/ NativeHandlerPercentageFee.sol	35b2ab178c26679469b2 e60145ec5376154020d7

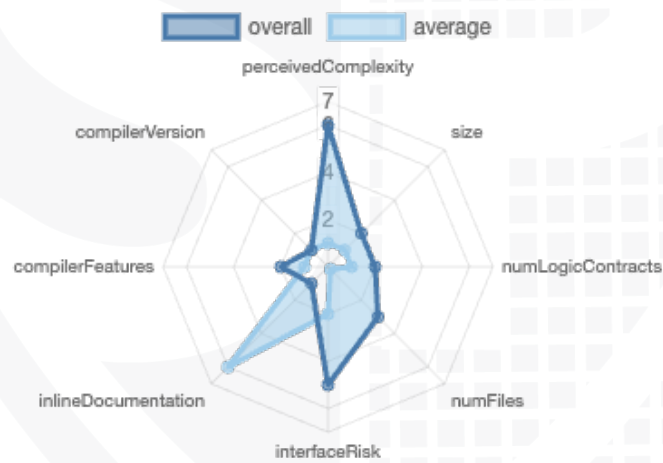


Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

 Contracts	 Libraries	 Interfaces	 Abstract
6	1	4	2

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.





 Public	 Payable
59	6







External	Internal	Private	Pure	View
46	91	11	7	27


StateVariables

Total	 Public
25	21

Capabilities

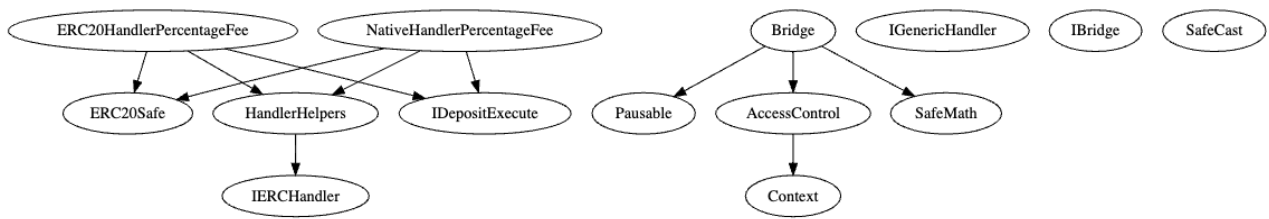
Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
0.8.11	ABIEncoderV2	yes	yes (4 asm blocks)	

 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRecover	 New/Create/Create2
yes			yes		

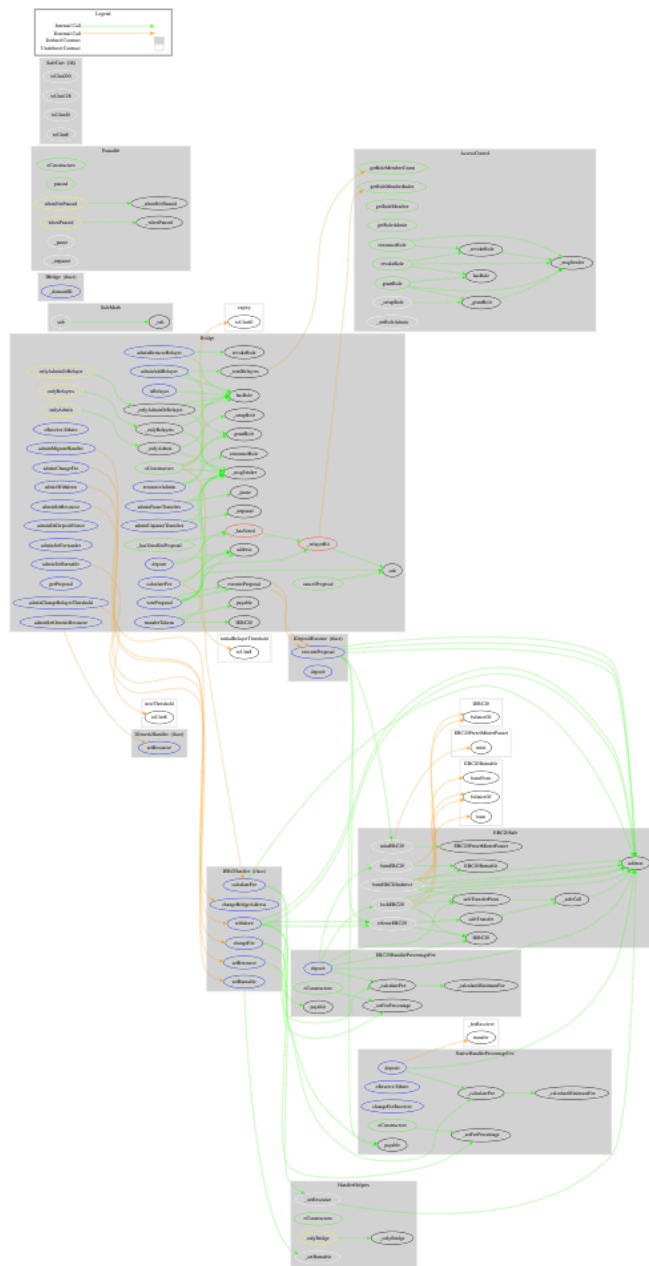
 TryCatch	Σ Unchecked
yes	

Inheritance Graph

v1.0



CallGraph v1.0



Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Is contract an upgradeable
2. Overall checkup (Smart Contract Security)



Is contract an upgradeable

Name	
Is contract an upgradeable?	No



Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✗
Not available	—

Modifiers and public functions

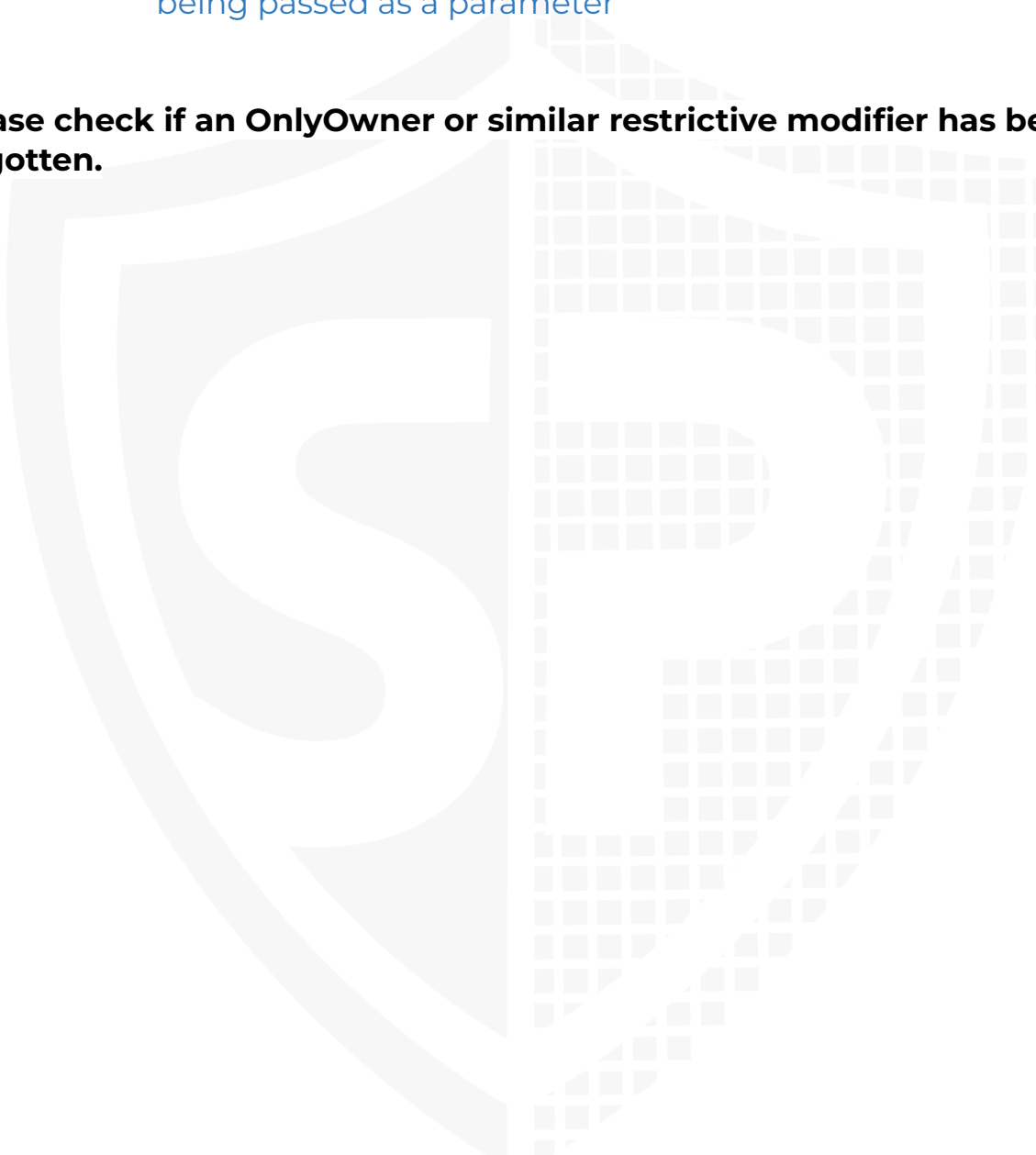
v1.0

The modifiers and public functions are the same as the ChainSafe project.

The only differences are:

- The fee is calculated and managed by another contract
- Now the fees are calculated in the contract rather than being passed as a parameter

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.



Source Units in Scope

v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
bridge/contracts/ERC20Safe.sol	1	—————	147	117	50	55	41
bridge/contracts/Bridge.sol	1	—————	586	540	283	191	240
bridge/contracts/interfaces/IGenericHandler.sol	—————	1	24	17	3	13	3
bridge/contracts/utills/SafeMath.sol	1	—————	46	42	11	26	3
bridge/contracts/interfaces/IERCHandler.sol	—————	1	53	17	6	33	13
bridge/contracts/interfaces/IBridge.sol	—————	1	14	13	3	9	3
bridge/contracts/handlers/HandlerHelpers.sol	1	—————	79	79	37	27	22
bridge/contracts/interfaces/IDepositExecute.sol	—————	1	26	14	3	14	8
bridge/contracts/utills/Pausable.sol	1	—————	97	97	34	51	12
bridge/contracts/utills/SafeCast.sol	1	—————	25	25	19	1	9
bridge/contracts/utills/AccessControl.sol	1	—————	214	214	59	131	47
bridge/contracts/handlers/ERC20/ERC20HandlerPercentageFee.sol	1	—————	179	161	95	43	77
bridge/contracts/handlers/ERC20/NativeToken/NativeHandlerPercentageFee.sol	1	—————	202	182	104	49	86
Totals	9	4	1692	1518	707	643	564

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

Critical issues

No critical issues

High issues

High issues found

Issue	File	Type	Line	Description	Status
#1	HandlerHelpers.sol	Bridge renouncing	45	<p>Since the bridge has the privileges for every "onlyBridge" function you should take care of the "_bridgeAddress". It can be set accidentally to zero/dead address.</p> <p>This will cause for example that it cannot be added new whitelist contract addresses anymore or the token fees of the depositor will be sent to the address of the bridge.</p>	Open

Medium issues

Medium issues found

Issue	File	Type	Line	Description	Status
#1	NativeHandlerPercentageFee.sol	Fee receiver	45	<p>If the fee receiver accidentally set the new receiver as dead/zero address every funds while depositing are sent to these addresses. These funds are lost.</p> <p>It is recommended to implement a renounceFeeReceiver function to achieve this and check the zero/dead address in the "changeFeeReceiver" function.</p>	Open

Low issues

Issue	File	Type	Line	Description	Status
#1	Bridge.sol	Missing Zero Address Validation (missing-zero-check)	304, 314	Check that the address is not zero or dead because this will change the "_bridgeAddress" in the handle which can cause massive problems in the contract.	Open
#2	Bridge.sol	Missing Arithmetic Events	293, 304	Emit Events for critical parameter changes	Open
#3	Bridge.sol	Missing Contract Address Check	309, 239	Make sure that the address is a contract, and not an EOA as it can disrupt the intended behavior of the bridge.	Open

#4	Bridge.sol	Contract will be overridden	See Description	<p>A bridge contract will be overridden in the <code>_resourceIdToTokenContractAddress</code> by calling the <code>setResource</code> function in the <code>HandlerHelpers</code>.</p> <p>That is not so good because the <code>ERC20HandlerPercentageFee</code> is using this variable to make sure that the <code>tokenAddress</code> is a whitelisted.</p> <p>Now we assume that a Bridge Contract with Address 1 is whitelisted. Now that there is no check for the resource ID with 1 which means that I can pass another address with this resource ID. The Bridge Contract still be whitelisted but cannot call e.g. "deposit" function because it is not only checking the <code>resourceId</code>, it is getting the <code>tokenaddress</code> from the resource which will not be the expected bridge contract</p> <p>And when the Bridge is expecting Address 1 but will get address 2 because it was overridden accidentally.</p>	Open
#5	HandlerHelpers.sol	Missing General Checks	68	In the <code>HandlerHelpers</code> there is not a check that the resources are already set or contract is whitelisted. Every value which is being set must be checked again.	Open

#6	HandleHelpers.sol	Whitelist addresses	68	<p>An address can be added to the “_resourceIDToTokenContractAddress” as whitelist but since it is possible to override existing data, the overridden address will not be in the list anymore.</p> <p>For example the resource id can be passed with another contract address to the function. That means the resourceID will be overridden with Contract B and the resource id will be set to “_tokenContractAddressToResourceID”. When this happens, the previous address will still have the same resource id as the new one. Additionally, the old contract address is still whitelisted. The bridge is not able to exclude an address from the whitelist after that.</p>	Open
#7	Bridge.sol	Missing value check	379	<p>The contract is not checking the msg.value which was send to the deposit function.</p> <p>In the NativeHandlerPercentageFee L72 there is a check for the passed msg.value L81 but in the ERC20HandlerPercentageFee L43 there is no check.</p>	Open
#8	NativeHandlerPercentageFee.sol	Missing state visibility	28	Specify the variables with a specific visibility. If you want to use a private/internal visibility then make sure to implement a function which returns these variables.	Open

Informational issues

Issue	File	Type	Line	Description	Status
#1	Bridge.sol	Misleading Function name	170	The name of the function should be related to the functionality which is to transfer the admin role, and not explicitly renouncing it.	Open

#2	Bridge.sol	Calculation Problem	98	<p>Here the calculation of</p> <pre>"sub(AccessControl.getRoleMemberIndex(RELAYER_ROLE, relayer), 1)"</pre> <p>should not be over 256 ($256 - 1 = 255$ bits to the left, on the first bit will be the 1 and the whole binary will have the length of 256) .</p> <p>The reason for it is the following: 1 will be left shifted. That means if the result of the calculation above will be 3 result will be 4.</p> <p>Why? Because it is the binary representation. $1 \ll 3$ means we shift the 1, 3 steps to the left. This would be 100. And 100 in binary is 4 in decimal. Everything above 256 Bits (because the uint256 can only have 256 bits) will be zero because the 1 is out of range.</p>	Open
#3	HandlerHelpers.sol	Naming convention	12-24	<p>Start variables/functions with an “_” for every private/internal types.</p> <p>Constants should start with an “_” and uppercase letters.</p> <p>If you are going to change these variables, make sure to change it also everywhere else.</p>	Open
#4	NativeHandlerPercentageFee.sol	Naming convention	16-25	<p>Start variables/functions with an “_” for every private/internal types.</p> <p>Constants should start with an “_” and uppercase letters.</p> <p>If you are going to change these variables, make sure to change it also everywhere else.</p>	Open
#5	NativeHandlerPercentageFee.sol	Revert	172	<p>Instead of using the “require” statement you can use directly the “revert” keyword if you want to revert it directly.</p>	Open

#6	ERC20HandlerPercentageFee.sol	Revert	149	Instead of using the “require” statement you can use directly the “revert” keyword if you want to revert it directly.	Open
#7	HandlerHelpers.sol	Wrong comment	46-56	<p>The function does not verify “_resrouceIDToContract[resourceID]” and “_contractAddressToResourceID” is already set.</p> <p>“_contractAddressToResourceID” variable does not exist here, it is called “_tokenContractAddressToResourceID” instead. And there is no check for already set variables.</p>	Open

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/latest/natspec-format.html>) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variable, functions etc. do.

13. June 2023:

- Read whole report and modifiers section for more information
- The IcecreamSwap team pointed out a vulnerability in the "voteProposal". Since the contract is a fork from the ChainSafe Bridge, the bug is also in the contract. Chainsafe has also been made aware of the bug, but this will not be fixed as they are no longer working on their bridge's v2. The IcecreamSwap team will fix the error.

SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-1 27	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 1	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

*Solid
Proofed*

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**


MADE IN GERMANY