



Bumper Finance
Protocol
SMART CONTRACT AUDIT
14.10.2021

Made in Germany by Chainsulting.de



Table of contents

1. Disclaimer	3
2. About the Project and Company	4
2.1 Project Overview	5
3. Vulnerability & Risk Level.....	6
4. Auditing Strategy and Techniques Applied	7
4.1 Methodology.....	7
4.2 Used Code from other Frameworks/Smart Contracts	8
4.3 Tested Contract Files.....	9
4.4 Metrics / CallGraph	10
4.4.1 Metrics / CallGraph	11
4.5 Metrics / Source Lines & Risk	12
4.6 Metrics / Capabilities.....	13
4.7 Metrics / Source Unites in Scope.....	14
5. Scope of Work.....	15
5.1 Manual and Automated Vulnerability Test	16
5.1.1 Upgrade chainlink AggregatorV3Interface to the newest version	16
5.1.2 Fix Spelling and Grammatical Errors	17
5.1.3 ABIEncoder v2.....	18
5.2 SWC Attacks	19
6. Executive Summary	23
7. Deployed Smart Contract.....	23



1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, 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.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Bumper Finance. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (02.10.2021)	Layout
0.2 (03.10.2021)	Test Deployment
0.5 (04.10.2021)	Automated Security Testing Manual Security Testing
0.6 (05.10.2021)	Testing SWC Checks
0.7 (05.10.2021)	Verify Claims
0.9 (06.10.2021)	Summary and Recommendation
1.0 (06.10.2021)	Final document
1.1 (09.10.2021)	Re-check e267a663
1.1 (12.10.2021)	Re-check 76029a80
1.2 (14.10.2021)	Added deployed contract addresses

2. About the Project and Company



Website: <https://www.bumper.fi>

Twitter: <https://twitter.com/bumperfinance>

Medium: <https://medium.com/bumper-finance>

Telegram: <https://t.me/bumperfinance>

Discord: <https://discord.gg/YyzRws4Ujd>

Github: <https://github.com/Bumper-Fi>

2.1 Project Overview

Bumper is a DeFi price protection protocol built on Ethereum which protects the price of crypto assets. Bumper provides a decentralised software facility for 'Takers' of protection to operate diametrically to 'Makers' of liquidity. Protected positions incur a floating daily premium, nominally 3% p.a, that is used to incentivise stablecoin depositors into a risk-free liquidity Reserve.

The Bumper protocol is a pure, decentralised market for on-chain asset price risk, which is transferred from a stablecoin Reserve through to cascading redundancy modules. At any point in time the Reserve has a measurable aggregate liability representing all positions. Should the liability exceed parameterized safety levels, the protocol rebalances, firstly by utilising first order dynamics, such as Premium/ Yield curves/ BUMP distributions and then by opening up to arbitrageur bots and if necessary DEX's. A separate risk pool, attracting a higher yield tranche, acts to backstop any realized losses caused by sharp volatility.

Conclusively, these redundancy measures make Bumper a highly productive tool to achieve efficient risk transfer pricing via liability pooling.

3. 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

4. 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.

4.1 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 Chainsulting 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 Chainsulting 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.

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

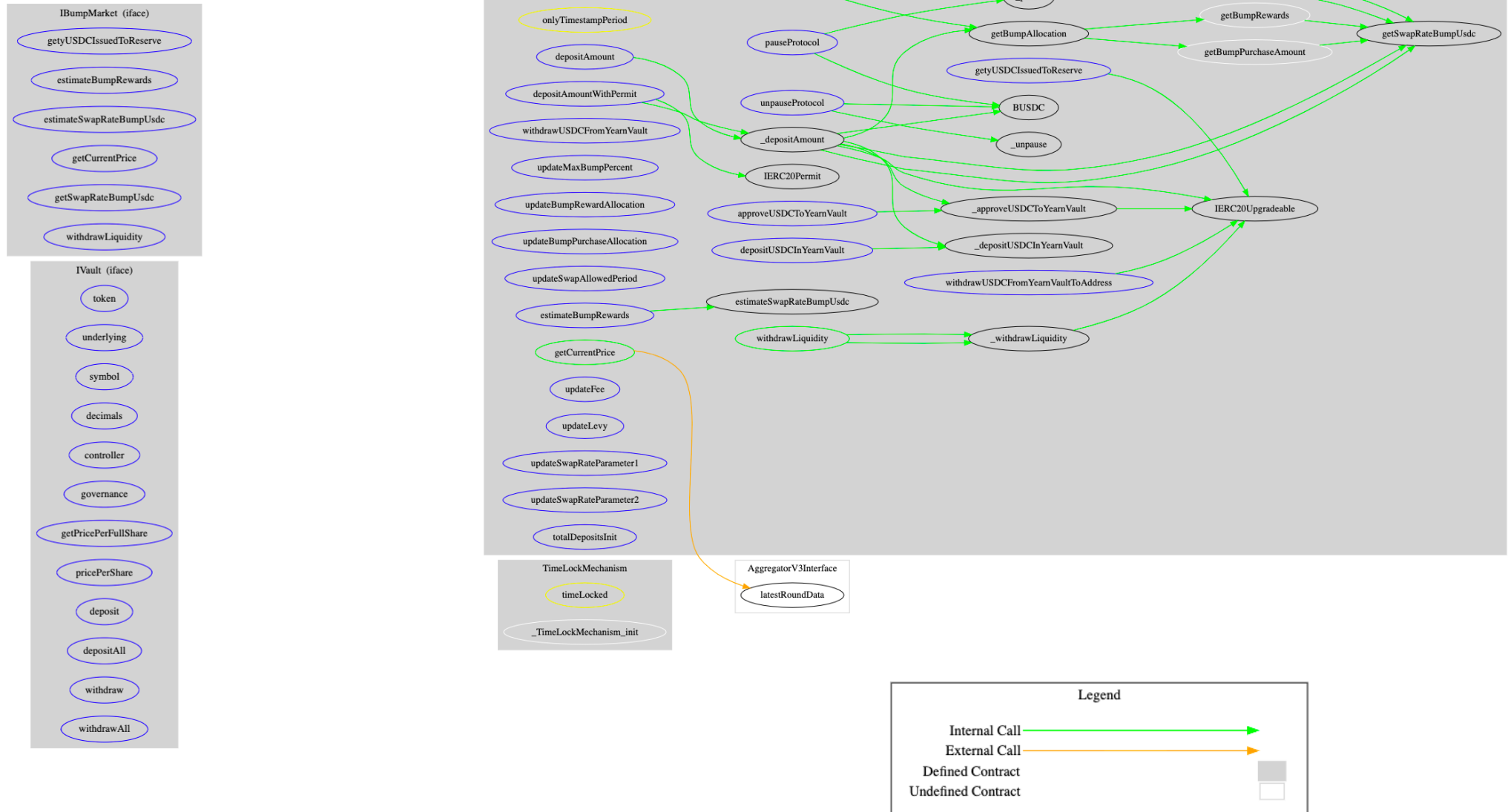
Dependency / Import Path	Source
@chainlink/contracts/src/v0.6/interfaces/AggregatorV3Interface.sol	https://github.com/smartcontractkit/chainlink/tree/develop/contracts/src/v0.6/interfaces/AggregatorV3Interface.sol
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.1.0/contracts/access/OwnableUpgradeable.sol
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.1.0/contracts/proxy/utils/Initializable.sol
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.1.0/contracts/security/PausableUpgradeable.sol
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.1.0/contracts/security/ReentrancyGuardUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.1.0/contracts/token/ERC20/IERC20Upgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC20PausableUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.1.0/contracts/token/ERC20/extensions/ERC20PausableUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.1.0/contracts/token/ERC20/utils/SafeERC20Upgradeable.sol
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.1.0/contracts/utils/ContextUpgradeable.sol
@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.0.0/contracts/token/ERC20/extensions/draft-IERC20Permit.sol

4.3 Tested Contract Files

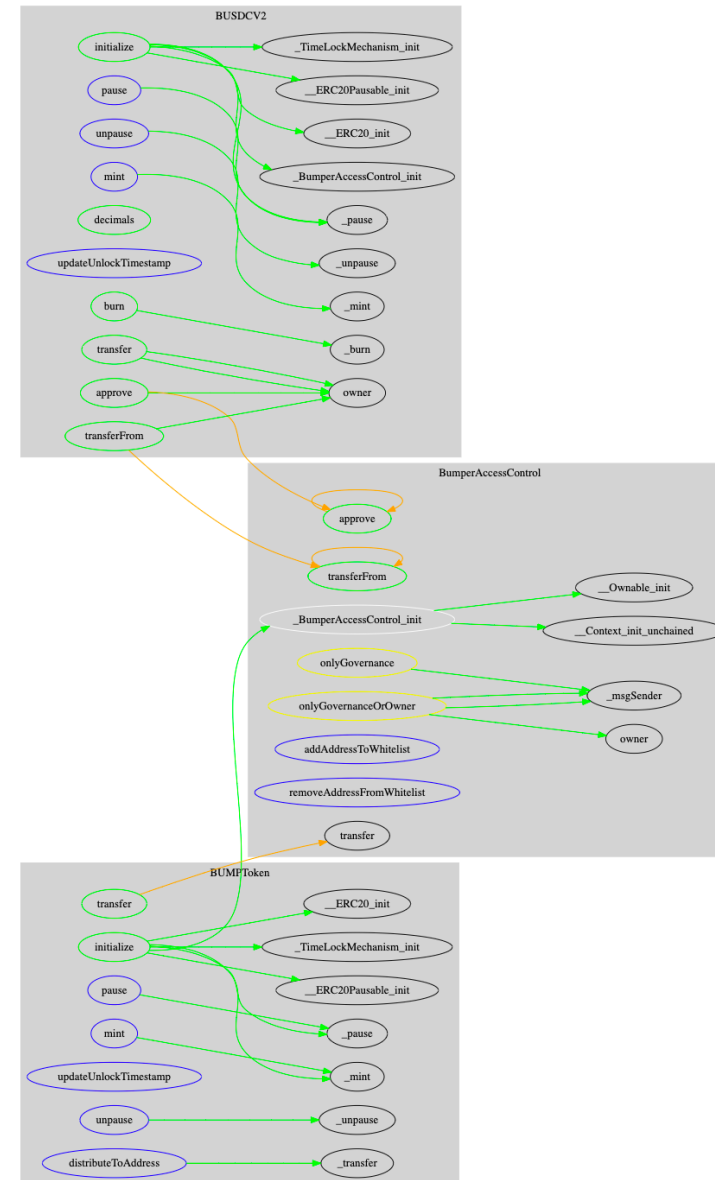
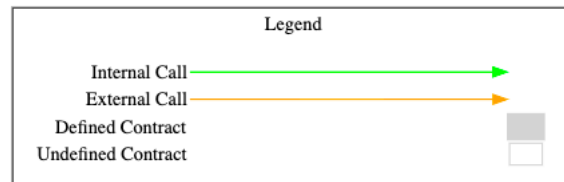
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
./contracts/BumpMarketV2.sol	a0f90e68d11e53f48200a7615047f791
./contracts/BUMPToken.sol	e9ff45f9576b7c8db8001465de7b3135
./contracts/BumperAccessControl.sol	0b902476cb924243db53e69545e3d7b7
./contracts/BUSDCV2.sol	ec4950d0949711f022c9df471eff0507
./contracts/TimeLockMechanism.sol	e2e0282b72845e648633b2f31a344dc0
./contracts/interfaces/IVault.sol	d0f71c8df51336b27cd4519ca73d5093
./contracts/interfaces/IBumpMarket.sol	acfd72cd1afc1dfbd8be9059f3913bd3

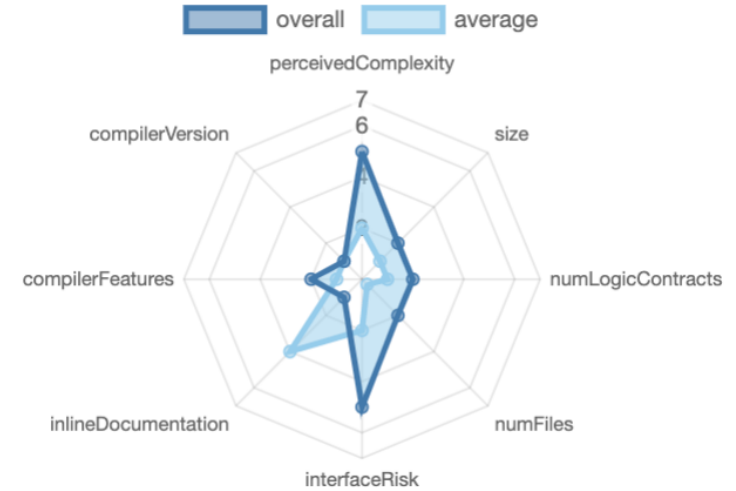
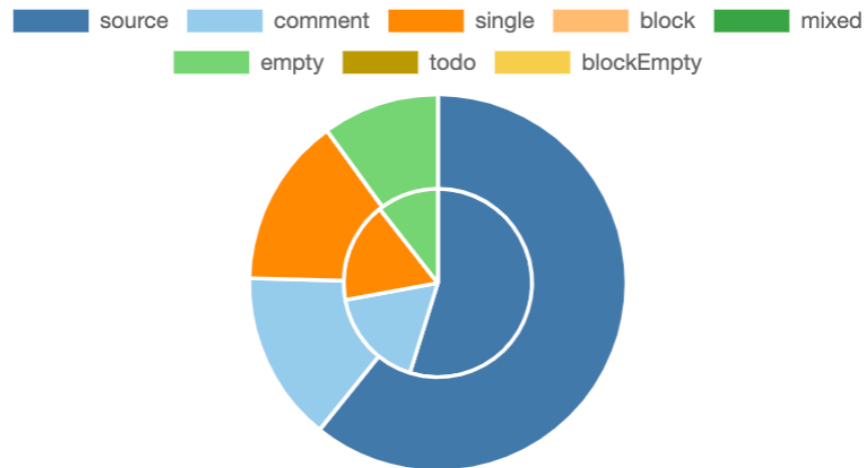
4.4 Metrics / CallGraph



4.4.1 Metrics / CallGraph



4.5 Metrics / Source Lines & Risk





4.6 Metrics / Capabilities


Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts			
0.8.0		ABIEncoderV2				**** (0 asm blocks)					
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECTrecover		 New/Create/Create2	
yes						yes					

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				
64	0				
External	Internal	Private	Pure	View	
50	61	1	0	20	

StateVariables

Total	 Public
23	21

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSL OC	Comment Lines	Complex. Score	Capabilities
	packages/web3/contracts/BUSDCV2.sol	1	_____	117	92	60	20	61	_____
	packages/web3/contracts/BumperAccessControl.sol	1	_____	61	58	40	9	30	
	packages/web3/contracts/TimeLockMechanism.sol	1	_____	33	30	21	3	6	
	packages/web3/contracts/BUMPToken.sol	1	_____	117	87	46	30	55	 
	packages/web3/contracts/BumpMarketV2.sol	1	_____	862	736	500	149	236	 
	packages/web3/contracts/interfaces/IVault.sol	_____	1	29	6	3	1	25	_____
	packages/web3/contracts/interfaces/IBumpMarket.sol	_____	1	27	8	4	1	13	_____
 	Totals	5	2	1246	1017	674	213	426	  

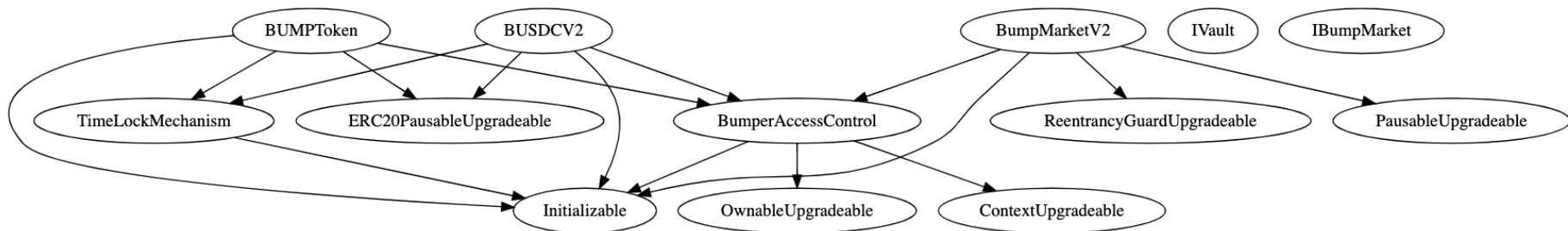
5. Scope of Work

The Bumper Finance Team provided us with the files that needs to be tested. The scope of the audit are the bumper protocol contracts.

The team put forward the following assumptions regarding the security, usage of the contracts:

- The smart contract is coded according to the newest standards and in a secure way

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

5.1.1 Upgrade chainlink AggregatorV3Interface to the newest version

Severity: LOW

Status: ACKNOWLEDGED

Code: NA

File(s) affected: BumpMarketV2.sol

Attack / Description	Code Snippet	Result/Recommendation
Old libraries can have bugs or vulnerabilities.	<pre>import "@chainlink/contracts/src/v0.6/interfaces/ AggregatorV3Interface.sol";</pre>	<p>It is recommended to upgrade the chainlink AggregatorV3Interface to the newest version, which is supporting 0.8.0.</p> <p>https://github.com/smartcontractkit/chainlink/blob/develop/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol</p>

INFORMATIONAL ISSUES

5.1.2 Fix Spelling and Grammatical Errors

Severity: INFORMATIONAL

Status: FIXED

Code: SWC-129

File(s) affected: BumpMarketV2.sol

Attack / Description	Code Snippet	Result/Recommendation
Spelling / Grammatical mistakes were identified in the codebase. Fixing these mistakes can help improve the end-user experience by providing clear information on errors encountered and improve the maintainability and auditability of the codebase.	BumpMarketV2.sol Line: 233 - 235 <pre>uint256 curentTime = block.timestamp; require(curentTime >= period.start && curentTime <= period.end,</pre>	Find and replace curentTime with currentTime or use currTime, as used with similar name for currBalance in line 253, to achieve consistency.

5.1.3 ABIEncoder v2

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: BumperAccessControl.sol, BumpMarketV2.sol, BUMPToken.sol, TimeLockMechanism.sol

Attack / Description	Code Snippet	Result/Recommendation
Solidity 0.8.0 has the ABI Encoder V2 activated by default. You can activate the old coder using pragma abicoder v1, or explicitly select v2 using pragma abicoder v2 - which has the same effect as pragma experimental ABIEncoderV2 had. ABI coder v2 is more complex than v1 but also performs additional checks on the input and supports a larger set of types than v1.	Line 3: <code>pragma experimental ABIEncoderV2;</code>	ABIEncoderV2 is activated by default since 0.8.0 and can be removed. https://blog.soliditylang.org/2020/12/16/solidity-v0.8.0-release-announcement/

5.2 SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✓
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✓
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✗
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✓
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✓
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✓
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✓
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✓

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✓
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✓
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✓
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✓
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✗
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✓
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✓

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✓
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✓
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✓
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓

6. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the October 06, 2021.

The main goal of the audit was to verify the claims regarding the security of the smart contract. During the audit, no critical issues were found, after the manual and automated security testing and the claim have been successfully verified.

7. Deployed Smart Contract

VERIFIED

Bump Market V2: <https://etherscan.io/address/0x2f57883d43db6118d9e5d65059baf7fe13a9c349#code>

Bump Market Proxy: <https://etherscan.io/address/0xBabeE6d5F6EDD301B5Fae591a0D61AB702b359d0#code>

bUSDC Proxy: <https://etherscan.io/address/0xf64f8958d2D44EAbae7517d013284c385A4CC2A3#code>

bUSDCV2: <https://etherscan.io/address/0xb85629f107ad634a95b1bfc853071d72e89a1e03#code>

Bump Token Proxy: <https://etherscan.io/address/0x785c34312dfA6B74F6f1829f79ADe39042222168#code>

Bump Token: <https://etherscan.io/address/0x46242e446aa798a62fb1e45bec1bf2c5e62ca0ea#code>

