Amun

Basket & Safe Polygon

SMART CONTRACT AUDIT

20.10.2021

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer

The audit makes no statements or warrantees 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 Amun Limited. 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 (06.09.2021)	Layout
0.2 (09.09.2021)	Test Deployment
0.5 (14.09.2021)	Automated Security Testing Manual Security Testing
0.6 (15.09.2021)	Testing SWC Checks
0.7 (15.09.2021)	Verify Claims
0.9 (15.09.2021)	Summary and Recommendation
1.0 (17.09.2021)	Final document
1.1 (22.10.2021)	Added deployed contract addresses

2. About the Project and Company

Company address:

Amun Limited
Suite 202 2nd Floor
Eden Plaza, Eden Island
PO Box 1352, Mahe
Seychelles

Website: https://amun.com

Twitter: twitter.com/amuntokens

Medium: https://medium.com/amun-tokens

Telegram: https://t.me/AmunTokens

Discord: https://discord.gg/MVJmyCzahH

Github: https://github.com/amun/contracts

2.1 Project Overview

Amun DeFi is a set of Ethereum-based tools that allow users to maximize their returns across major DeFi platforms through lending and staking by helping them choose the best product at any given time. Amun DeFi supports integrations with popular protocols representing tens of billions of total locked value (TVL). From purchasing major cryptocurrencies quickly and with ease, to swapping between popular ERC-20 tokens, all the way to lending tokens at the best available rates, Amun DeFi makes it simple to monitor and interact with a user's holdings on one secure, easy-to-use platform.

Through gateways with various lending platforms and AMMs, Amun DeFi enables users to execute their desired investment strategy with relative ease. Because there are so many different platforms that users can use to achieve maximum returns, the average DeFi user might not know precisely where to begin. Further, many retail investors will be unable to implement their intended strategy due to a lack of knowledge or technical sophistication.

Amun's goal is to reduce the fragmentation of this space by offering access to all the top protocols in one platform. They intend to offer the following features to the DeFi community :

1. Dashboard that monitors your holdings and depicts your exposure across the various protocols
2. Ability to invest in the top liquidity pools and lending protocols
3. Offers our own products (autopilot strategies) that automatically moves your funds to the protocol offering the best yields
4. Seamlessly diversify your investments through our basket tokens

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
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.1-solc-0.7/contracts/access/Ownable.sol
@openzeppelin/contracts/math/SafeMath.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.1-solc-0.7/contracts/math/SafeMath.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.1-solc-0.7/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.1-solc-0.7/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.1-solc-0.7/contracts/token/ERC20/SafeERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.1-solc-0.7/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/utils/math/SafeMath.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.2.1-solc-0.7/contracts/utils/math/SafeMath.sol
@pangolindex/exchange-contracts/contracts/pangolin-periphery/interfaces/IPangolinRouter.sol	https://github.com/pangolindex/exchange-contracts/contracts/pangolin-periphery/interfaces/IPangolinRouter.sol
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	https://github.com/Uniswap/v2-periphery/blob/master/contracts/interfaces/IUniswapV2Router02.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

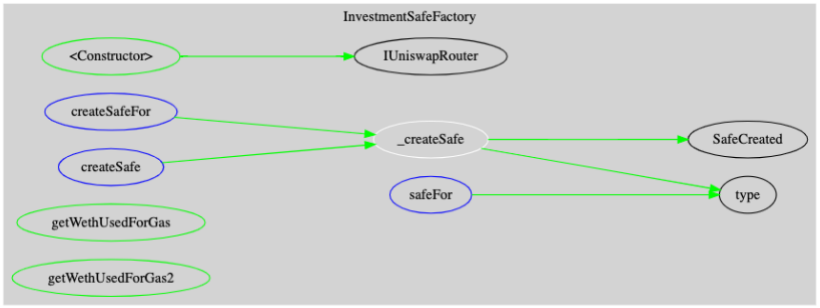
Basket

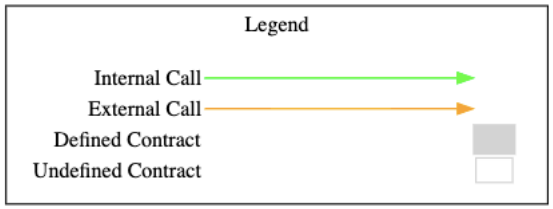
File	Fingerprint (MD5)
./EthSingleTokenJoinV2.sol	c976abb86f270a79096061ce6fdb9a1e
./SingleNativeTokenExitV2.sol	249e0a7252af2dc44c5dd7fa57dbc831
./SingleTokenJoinV2.sol	5bb36da3a3257eef46da8afede14c806
./callManager/RebalanceManagerV3.sol	02eacf1d9e7d78e5b48199837a44dbd7

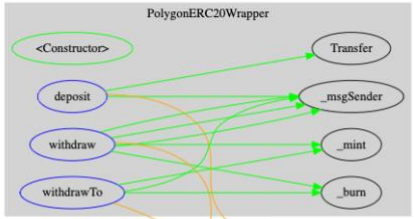
Safe

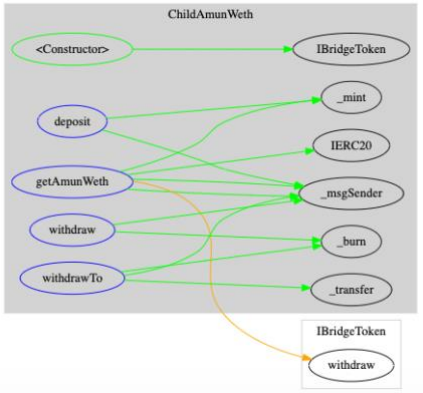
File	Fingerprint (MD5)
./ChildAmunWeth.sol	93efb1117e824064a55224e33488cb22
./InvestmentSafe.sol	52743fa2b9cb0ee98aa0acb91d7c24eb
./InvestmentSafeFactory.sol	bdbf684054d60db9871d7d931a01076b
./PolygonERC20Wrapper.sol	0cd41be9358853ac7e1e915f93e5fa61
./RootAmunWeth.sol	f29c20a8d780e23a380ae19431b9e4bc

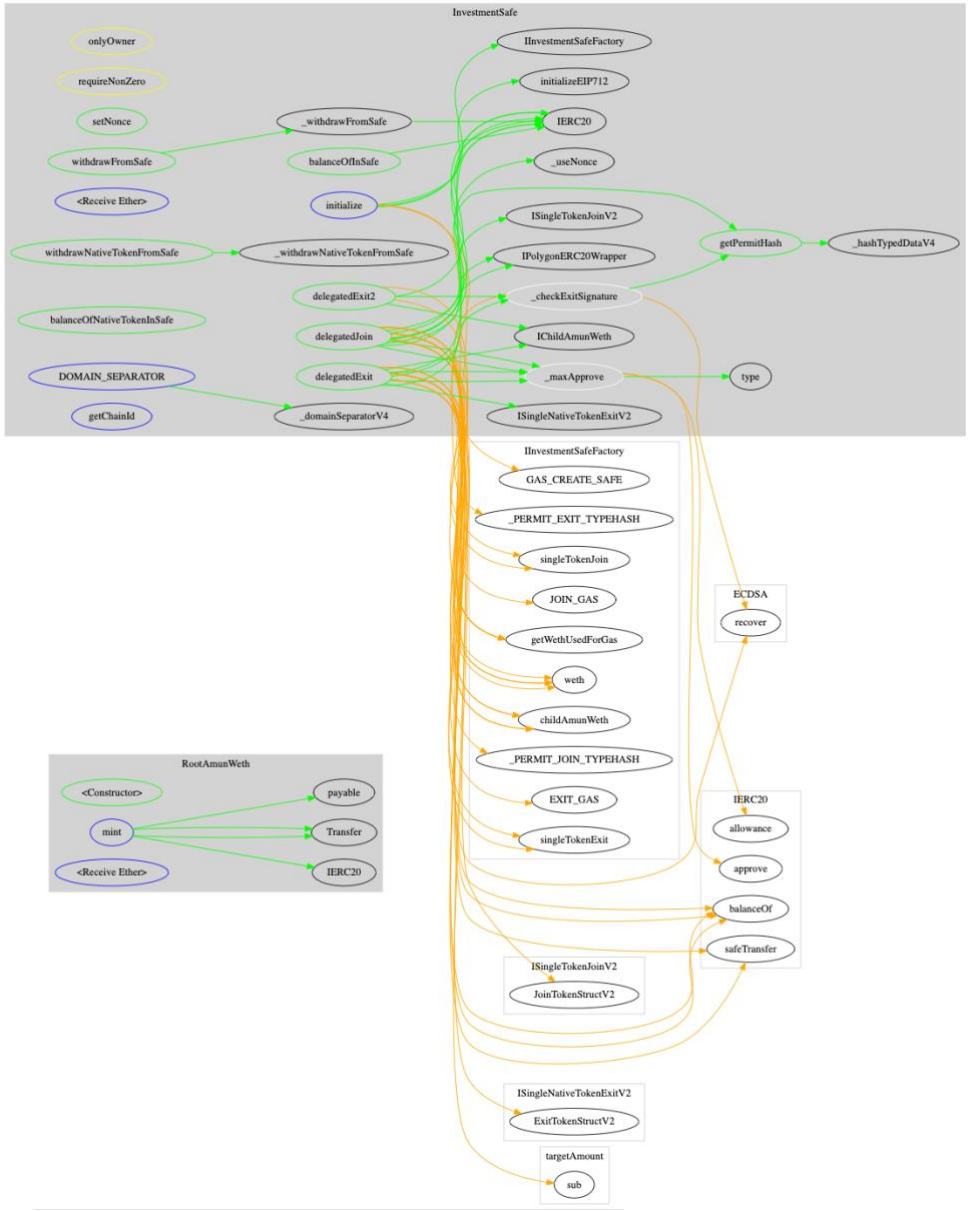
4.4 Metrics / CallGraph (Safe)



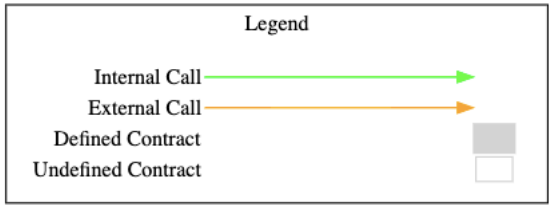


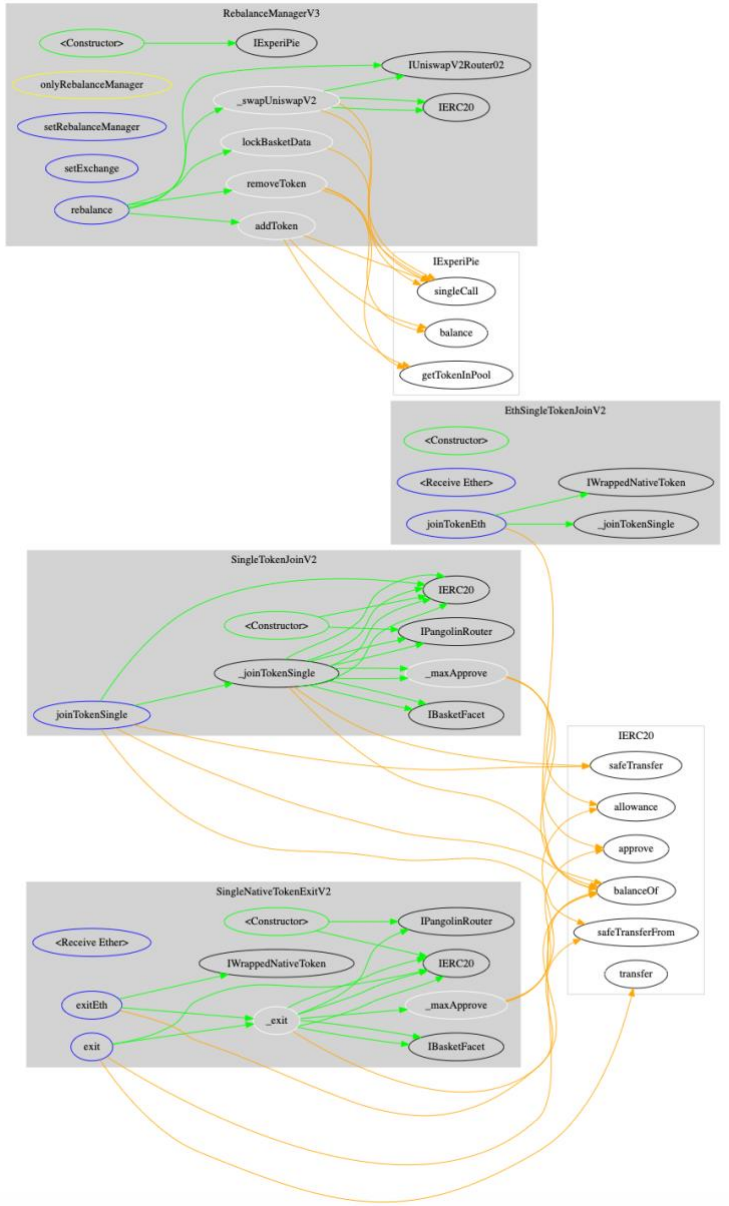




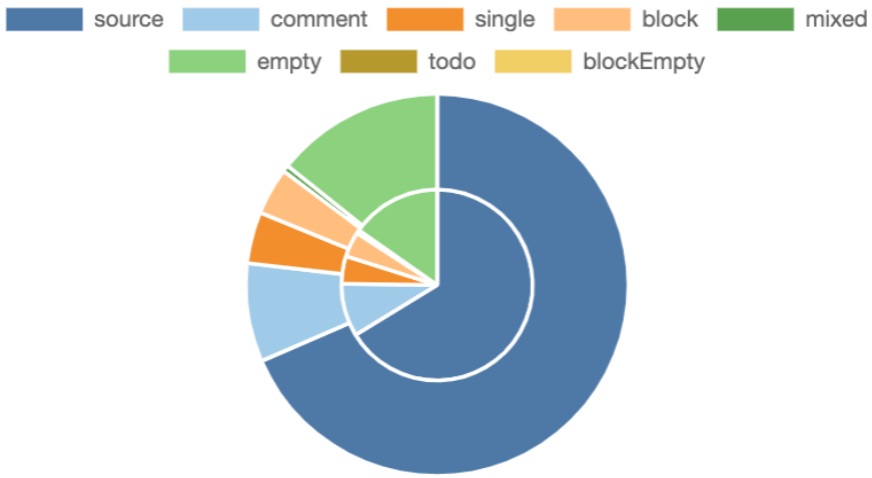


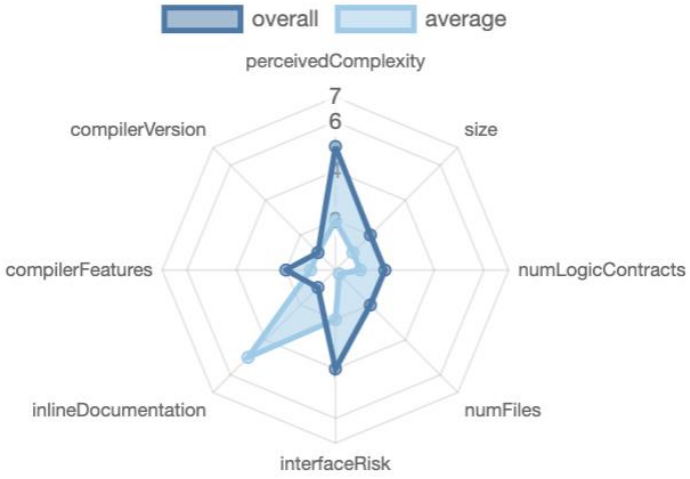
4.5 Metrics / CallGraph (Basket)





4.6 Metrics / Source Lines & Risk





4.7 Metrics / Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`^0.7.5` `^0.8.6`	`ABIEncoderV2`	`yes`	`yes` (1 asm blocks)	

Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
`yes`			`yes`		`yes` → `AssemblyCall:Name:create2`

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
37	5

External	Internal	Private	Pure	View
26	60	0	0	8

StateVariables

Total	Public
30	30

4.8 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
□	bridge-safe-sc-main/contracts/InvestmentSafeFactory.sol	1		115	99	74	6	80	□□□
□	bridge-safe-sc-main/contracts/RootAmunWeth.sol	1		39	39	29	5	23	□□
□	bridge-safe-sc-main/contracts/InvestmentSafe.sol	1		381	340	271	19	156	□□□
□	bridge-safe-sc-main/contracts/ChildAmunWeth.sol	1		77	77	41	28	34	
□□	bridge-safe-sc-main/contracts/PolygonERC20Wrapper.sol	1	1	60	55	29	19	35	
□□	**Totals**	**5**	**1**	**672**	**610**	**444**	**77**	**328**	□□□□□□

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
□	LimaVaults-LIMA-Matic-add-v2-of-single-token-join/contracts/EthSingleTokenJoinV2.sol	1		34	32	22	2	27	□□□

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
□	LimaVaults-LIMA-Matic-add-v2-of-single-token-join/contracts/SingleTokenJoinV2.sol	1		139	135	103	8	77	□
□	LimaVaults-LIMA-Matic-add-v2-of-single-token-join/contracts/callManagers/RebalanceManagerV3.sol	1		195	175	133	15	87	□
□	LimaVaults-LIMA-Matic-add-v2-of-single-token-join/contracts/SingleNativeTokenExitV2.sol	1		121	121	89	5	67	□□□
□	**Totals**	**4**		**489**	**463**	**347**	**30**	**258**	□□□

Legend: [十

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized 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, ...)

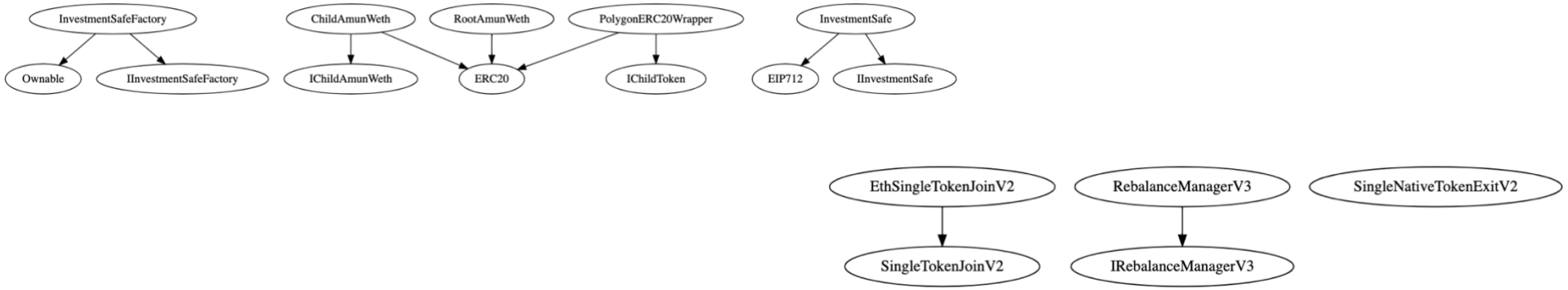
5. Scope of Work

The Amun Team provided us with the files that needs to be tested. The scope of the audit are the Basket and Safe contracts for Polygon network.

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 SPDX license identifier not provided
Severity: LOW
Status: ACKNOWLEDGED
Code: NA
File(s) affected: IChildAmunWeth.sol, IInvestmentSafe.sol, IPolygonERC20Wrapper.sol, ISingleNativeTokenExitV2.sol, IsingleTokenJoinV2.sol,

Attack / Description	Code Snippet	Result/Recommendation
SPDX license identifier not provided in source file.	-	Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.

5.1.2 Missing natspec documentation
Severity: LOW
Status: ACKNOWLEDGED
Code: CWE-1056
File(s) affected: InvestmentSafe.sol, InvestmentSafeFactory.sol, SingleTokenJoinV2.sol, SingleNativeTokenExitV2.sol, EthSingleTokenJoinV2.sol

Attack / Description	Code Snippet	Result/Recommendation
Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables and more. This special form is named the Ethereum Natural Language Specification Format (NatSpec).	NA	It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract. There are already in-line comments inside the codebase, but it can be increased.

INFORMATIONAL ISSUES

5.1.3 Fix Spelling and Grammatical Errors

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: SWC-129
File(s) affected: InvestmentSafe.sol

Attack / Description	Code Snippet	Result/Recommendation
Language 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.	InvestmentSafe.sol Line: 60 `address _createor`	Find and replace createor with creator

5.1.4 A floating pragma is set

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: SWC-103
File(s) affected: ALL

Attack / Description	Code Snippet	Result/Recommendation
The current pragma Solidity directive is "^0.7.5". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary	Line 1: `pragma solidity ^0.7.5;`	It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. i.e. Pragma solidity 0.7.5

between builds. This is especially important if you rely on bytecode-level verification of the code.		

5.1.5 ABIEncoder v2

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: NA
File(s) affected: InvestmentSafe.sol

Attack / Description	Code Snippet	Result/Recommendation
The second change since solidity 0.8.0 that is very visible is that the ABI coder v2 is 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: pragma experimental ABIEncoderV2;	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/

5.1.6 Public functions could be external

Severity: INFORMATIONAL
Status: ACKNOWLEDGED

Code: SWC-103
File(s) affected: InvestmentSafe.sol, InvestmentSafeFactory.sol

Attack / Description	Code Snippet	Result/Recommendation
In the current implementation several functions are declared as public where they could be external. For public functions Solidity immediately copies array arguments to memory, while external functions can read directly from calldata. Because memory allocation is expensive, the gas consumption of public functions is higher.	setNonce(address,uint256) should be declared external: InvestmentSafe.setNonce(address,uint256) (contracts/InvestmentSafe.sol#76-79) withdrawFromSafe(address,uint256) should be declared external: InvestmentSafe.withdrawFromSafe(address,uint256) (contracts/InvestmentSafe.sol#84-86) balanceOfInSafe(address) should be declared external: InvestmentSafe.balanceOfInSafe(address) (contracts/InvestmentSafe.sol#99-101) withdrawNativeTokenFromSafe(uint256) should be declared external: InvestmentSafe.withdrawNativeTokenFromSafe(uint256) (contracts/InvestmentSafe.sol#108-110) balanceOfNativeTokenInSafe() should be declared external: InvestmentSafe.balanceOfNativeTokenInSafe() (contracts/InvestmentSafe.sol#121-123)	We recommend declaring functions as external if they are not used internally. This leads to lower gas consumption and better code readability.

	delegatedJoin(IInvestmentSafe.DelegateJoinData,uint256,ISingleTokenJoinV2.UnderlyingTrade[]) should be declared external: InvestmentSafe.delegatedJoin(IInvestmentSafe.DelegateJoinData,uint256,ISingleTokenJoinV2.UnderlyingTrade[]) (contracts/InvestmentSafe.sol#162-242) delegatedExit(IInvestmentSafe.DelegateExitData,uint256,ISingleNativeTokenExitV2.ExitUnderlyingTrade[]) should be declared external: InvestmentSafe.delegatedExit(IInvestmentSafe.DelegateExitData,uint256,ISingleNativeTokenExitV2.ExitUnderlyingTrade[]) (contracts/InvestmentSafe.sol#268-329) delegatedExit2(IInvestmentSafe.DelegateExitData) should be declared external: InvestmentSafe.delegatedExit2(IInvestmentSafe.DelegateExitData) (contracts/InvestmentSafe.sol#332-359) getWethUsedForGas(uint256) should be declared external: InvestmentSafeFactory.getWethUsedForGas(uint256) (contracts/InvestmentSafeFactory.sol#108-110) getWethUsedForGas2(uint256,uint256) should be declared external:	

	InvestmentSafeFactory.getWethUsedForGas2(uint256,uint256) (contracts/InvestmentSafeFactory.sol#112-114)	

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 September 19, 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. Considering the complexity of building a token bridge smart contract, the approach that the Amun Team has taken, has decreased the attack surface significantly.

7. Deployed Smart Contract

VERIFIED

SingleTokenJoinV2
https://polygonscan.com/address/0x642BDFc54d52396F90Ec67eC9617F469a697135D#code

SingleNativeTokenExitV2
https://polygonscan.com/address/0xc89ccd53473881a5c0c78da2d570fb2e661c8ce9#code

ChildAmunWeth
https://polygonscan.com/address/0xe43fd6f70e55c0192a0b9cfe2959093d61606975#code

ChildAmunWeth (Proxy)
https://polygonscan.com/address/0x96A73E3b9E7A12CAeBC72838d9c8a90c02556dA8#code

DummyMintableERC20
https://etherscan.io/address/0x8ac89b31fc3094b7f4649bac56950935ff96d5df#code

DummyMintableERC20 (Proxy)
https://etherscan.io/address/0x9d3EE6B64e69Ebe12a4bF0b01D031CB80F556eE4#code

RootAmunWeth
https://etherscan.io/address/0xad8b19a32817b866b6b1e63c246f35f2ce48dd2c#code

RootAmunWeth (Proxy)
https://etherscan.io/address/0x96A73E3b9E7A12CAeBC72838d9c8a90c02556dA8#code

PolygonERC20Wrapper
https://polygonscan.com/address/0x8ac89b31fc3094b7f4649bac56950935ff96d5df#code

PolygonERC20Wrapper (Proxy)
https://polygonscan.com/address/0x13607B1ca022368c81f2f2709b181ee8e0f42dD6#code