Unibright

UBT Payment Splitter

SMART CONTRACT AUDIT

22.02.2022

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 Tested Contract Files 8
   4.3 Used Code from other Frameworks/Smart Contracts 8
   4.4 Metrics / CallGraph 9
   4.5 Metrics / Source Lines & Risk 10
   4.6 Metrics / Capabilities 11
   4.7 Metrics / Source Unites in Scope 12
5. Scope of Work 13
   5.1 Manual and Automated Vulnerability Test 14
   5.1.1 A floating pragma is set. 14
   5.2. SWC Attacks 15
6. Executive Summary 19
7. Deployed Smart Contract 19

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 Unibright IT GmbH. 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 (17.02.2022)	Layout
0.4 (21.02.2022)	Automated Security Testing Manual Security Testing
0.5 (22.02.2022)	Verify Claims and Test Deployment
0.6 (22.02.2022)	Testing SWC Checks
0.9 (22.02.2022)	Summary and Recommendation
1.0 (22.02.2022)	Final document
1.1 (TBA)	Added deployed contract

2. About the Project and Company

Company address:

Unibright IT GmbH
Nahestr. 28
55411 Bingen am Rhein
Germany

Website: https://unibright.io

Twitter: https://twitter.com/UnibrightIO

Telegram: https://t.me/unibright_io

Medium: https://medium.com/unibrightio

Facebook: https://www.facebook.com/Unibright.IO

LinkedIn: https://www.linkedin.com/company/unibright

Coingecko: https://www.coingecko.com/en/coins/unibright

CoinMarketCap: https://coinmarketcap.com/currencies/unibright/

Bitcointalk: https://bitcointalk.org/index.php?topic=2892915.0

Reddit: https://reddit.com/r/Unibright

Etherscan: https://etherscan.io/token/0x8400d94a5cb0fa0d041a3788e395285d61c9ee5e

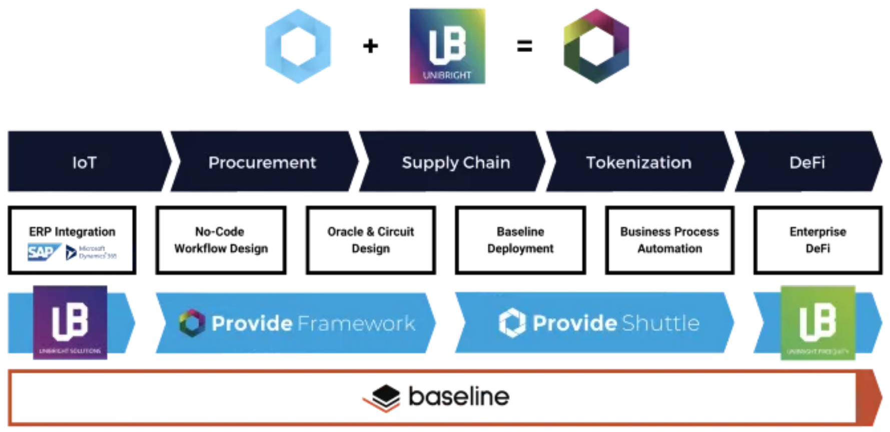
2.1 Project Overview

Unibright was launched in 2016 and is backed by a team of experts in the field of blockchain, as well as architects, developers and consultants that have over 20 years of experience in business processes and integration. Unibright offers a unified framework with the goal of bringing blockchain technology and contracts to mainstream users. Unibright offers enterprise-level blockchain solutions, integration platforms and an ecosystem centered around tokenized assets.

Unibright also plays an important role in the development of the Baseline Protocol. This protocol is designed to connect traditional systems to each other and notarize states on a public mainnet, such as Ethereum.

Baseline Protocol is an open-source initiative. Acting as middleware, it combines advances in cryptography and messaging with blockchain in order to deliver secure and private business processes at a low cost.

Baseline Protocol can automate B2B agreements without the need to create new blockchain silos and integrate new relationships with existing ones in a flexible manner and without the loss of system integrity. Furthermore, Baseline Protocol can enforce consistency between different parties' records without the need to move the data from legacy systems. It can also enforce a multiparty workflow.



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 Tested Contract Files

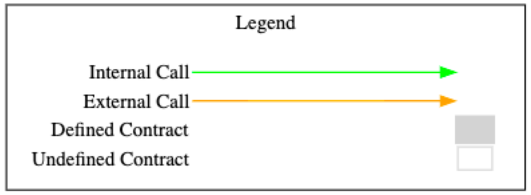
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability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

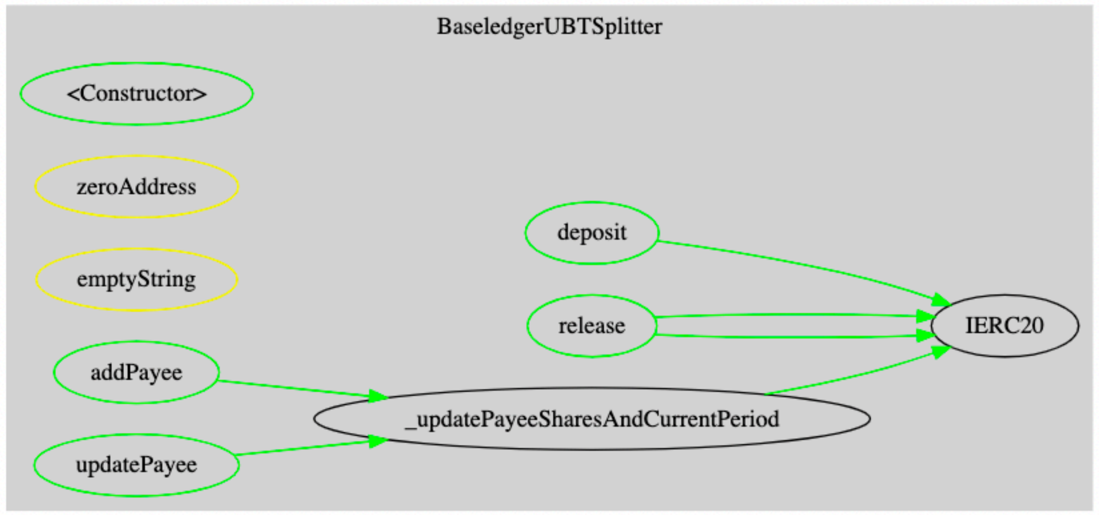
File	Fingerprint (MD5)
BaseledgerUBTSplitter.sol	52d16cb6d2511c8380632cf8f88d18e7

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

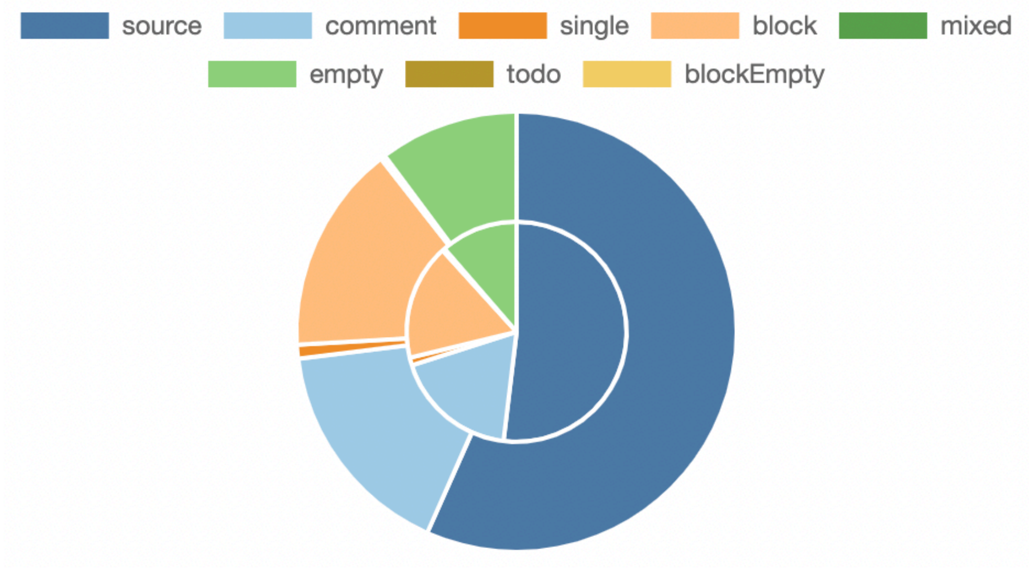
Dependency / Import Path	**Source**
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.1/contracts/access/Ownable.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.1/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/utils/Address.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.1/contracts/utils/Address.sol
@openzeppelin/contracts/utils/Context.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.1/contracts/utils/Context.sol

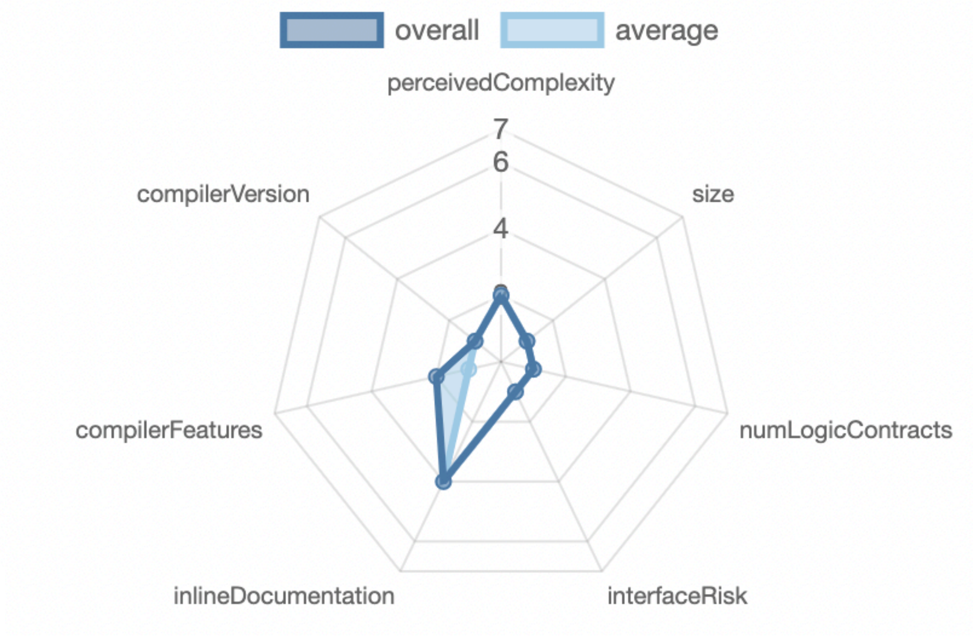
4.4 Metrics / CallGraph





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`				

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
`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
4	0

External	Internal	Private	Pure	View
0	6	1	0	0

StateVariables

Total	🌐Public
12	12

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	BaseledgerUBTSplitter.sol	1		243	214	137	48	60	📤
📝	**Totals**	**1**		**243**	**214**	**137**	**48**	**60**	📤

Legend: [➖]

- **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 Unibright Team provided us with the file that needs to be tested. The scope of the audit is the UBT Splitter contract.

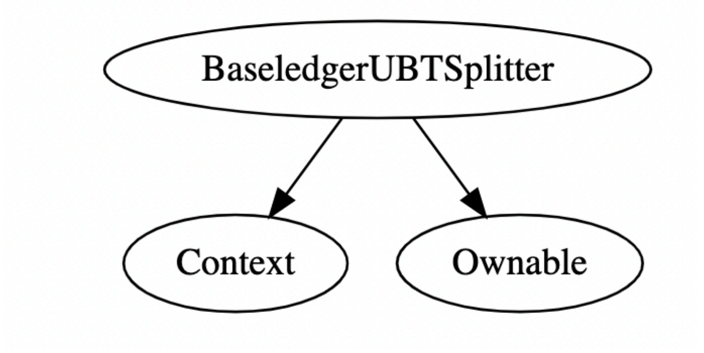
Following contract with the direct imports has been tested:

- BaseledgerUBTSplitter.sol

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

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

INFORMATIONAL ISSUES

5.1.1 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 "^0.8.0". It is recommended to specify a	Line 1: `pragma solidity ^0.8.0;`	It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did

fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.		not foresee. i.e. Pragma solidity 0.8.0

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	✓

ID	Title	Relationships	Test Result
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	✅
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	✅

ID	Title	Relationships	Test Result
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')	✓
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	✓

ID	Title	Relationships	Test Result
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	✓
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	X
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 main goal of the audit was to verify the claims regarding the security of the smart contract and the functions. During the audit, no critical issues were found, after the manual and automated security testing. Only one informational issue was found. Overall, everything worked as it was supposed to be, we have been satisfied with the code quality and security measures, that has been taken.

7. Deployed Smart Contract

PENDING