



AllUnity

Stablecoin Framework

SMART CONTRACT AUDIT

15.05.2025

Made in Germany by Softstack.io



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

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 |
| 5. Metrics | 8 |
| 5.1 Tested Contract Files | 8 |
| 5.2 Inheritance Graph..... | 9 |
| 5.3 Source Lines & Risk..... | 10 |
| 5.4 Capabilities | 11 |
| 5.5 Dependencies / External Imports..... | 12 |
| 5.6 Source Unites in Scope | 13 |
| 6. Scope of Work..... | 14 |
| 6.1 Findings Overview | 15 |
| 6.2 Manual and Automated Vulnerability Test..... | 16 |
| 6.2.1 Conditional Allowance Bypass in burnFrom for Blacklisted Accounts..... | 16 |
| 6.2.2 Missing Storage Gap in AllUnity.sol | 17 |
| 6.2.3 Missing Zero-Address Checks in initialize() | 19 |
| 6.2.4 Gas Limit Exhaustion in BlacklistManagement.batchAddBlacklist..... | 21 |
| 6.2.5 Missing batchRemoveBlacklist Function in BlacklistManagement.sol..... | 22 |
| 6.3 Verify Claims | 23 |



| | |
|----------------------------|----|
| 7. Executive Summary..... | 25 |
| 8. About the Auditor | 26 |

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 AllUnity 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 (07.05.2025) | Layout |
| 0.4 (08.05.2025) | Automated Security Testing Manual Security Testing |
| 0.5 (12.05.2025) | Verify Claims |
| 0.9 (13.05.2025) | Summary and Recommendation |
| 1.0 (15.05.2025) | Final document |



2. About the Project and Company

Company address:

AllUnity GmbH
Sandweg 94
Frankfurt 60316
Germany



Website: <https://allunity.com/>

Twitter (X): <https://x.com/AllUnityStable>

GitHub: <https://github.com/All-Unity>

LinkedIn: <https://www.linkedin.com/company/allunity/>



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

2.1 Project Overview

AllUnity is a MiCAR-aligned, institutional-grade infrastructure provider for secure and compliant on-chain euro payments. Backed by leading financial and Web3 institutions DWS, Flow Traders, and Galaxy. AllUnity delivers a fully collateralized EUR stablecoin designed for regulated, real-time value transfer across tokenized markets.

At its core, AllUnity operates as a regulated e-money institution under BaFin supervision, anchoring its stablecoin issuance in full compliance with the EU's Markets in Crypto-Assets Regulation (MiCAR). Its infrastructure is engineered to meet the needs of financial institutions, exchanges, and DeFi platforms seeking predictable, low-latency euro liquidity across digital rails.

AllUnity's architecture includes an interoperable issuance engine, fiat on/off-ramp integrations, and real-time auditing and compliance tooling, ensuring stablecoin transparency and trust. The stablecoin itself is fully collateralized in euro, held with EU-based financial institutions, and subject to regular reporting and attestation.

Use cases range from intra-bank settlements, institutional DeFi, and tokenized asset markets to high-speed cross-border payments. AllUnity also supports automated escrow and smart contract-based clearing functions, reducing counterparty risk and operational complexity.

To facilitate adoption, AllUnity offers API-first access, streamlined onboarding, and partner modules for compliance, analytics, and liquidity provisioning. Its roadmap includes expanding to additional EEA markets, exploring programmable finance use cases, and integrating with on-chain ESG frameworks for sustainable finance tracking.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

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 auditors 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 softstack 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 softstack 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.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 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.

Source: <https://github.com/All-Unity/smart-contracts>

Commit: 9d56d76054128c5b7fe5b5258fefa829abc21839 (re-check 15.05.2025)

| File | Fingerprint (MD5) |
|--|----------------------------------|
| ./ethereum-contracts/contracts/AllUnity.sol | 7ceb32e85b56878d80dbe6e7de810a77 |
| ./ethereum-contracts/contracts/BlacklistManagement.sol | 8fbc26ed5610c5cc6342f2f4fd7911b4 |
| ./ethereum-contracts/contracts/AutProxy.sol | 0faf2a570689e39d5c581af8cb88e075 |

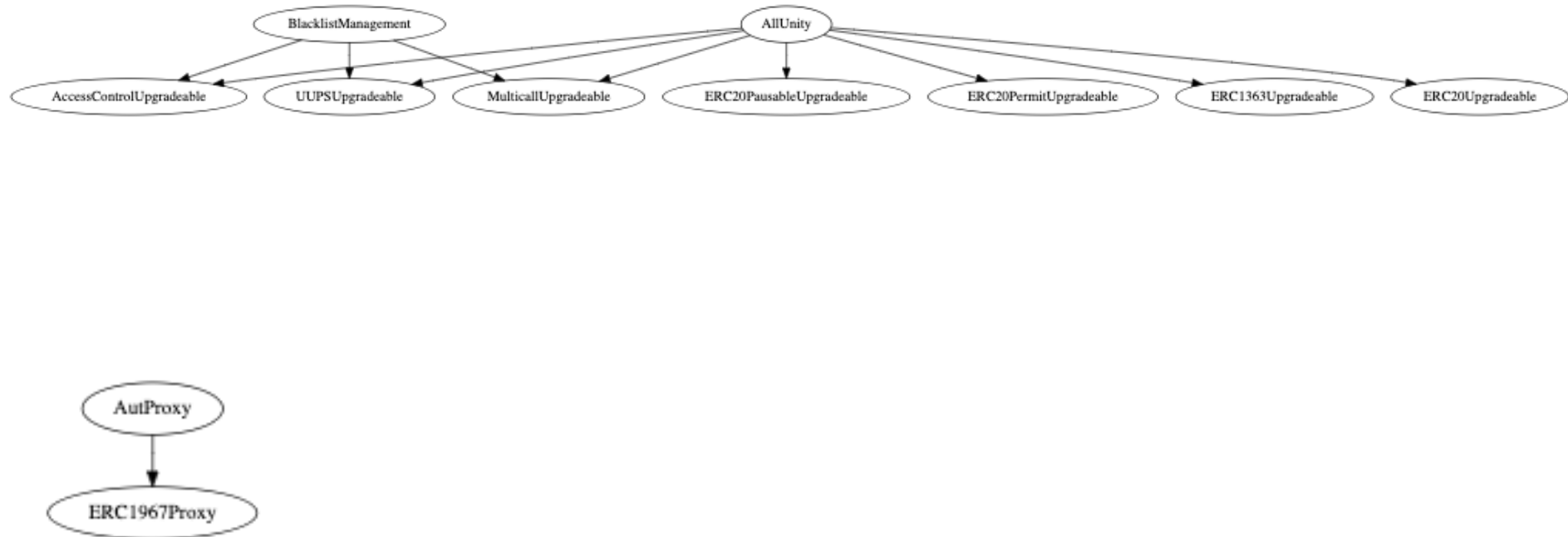


hello@softstack.io
www.softstack.io

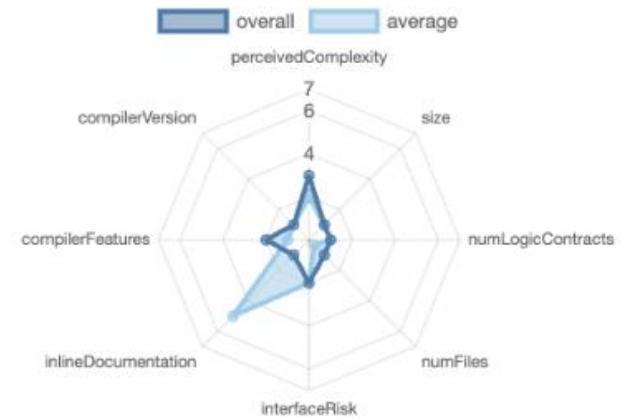
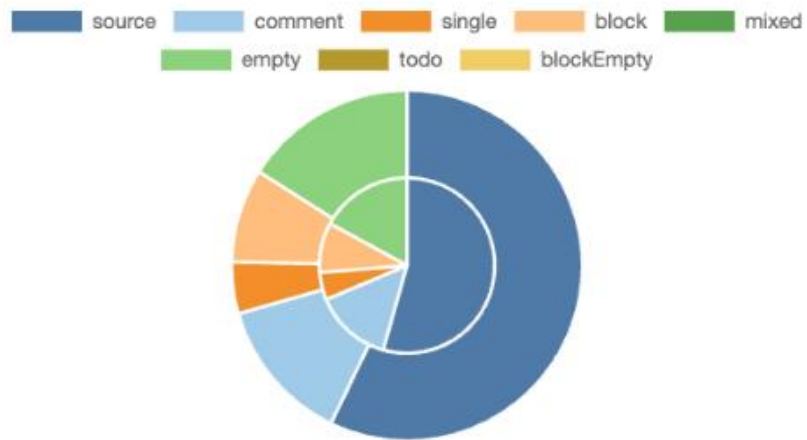
softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

5.2 Inheritance Graph



5.3 Source Lines & Risk





hello@softstack.io
www.softstack.io



softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984


5.4 Capabilities

| Solidity Versions observed | |  Experimental Features |  Can Receive Funds |  Uses Assembly |  Has Destroyable Contracts |
|---|---|---|---|---|---|
| <div>^0.8.22</div> | | | <div>yes</div> | | |
|  Transfers ETH |  Low-Level Calls |  DelegateCall |  Uses Hash Functions |  Ecrecover |  New/Create/Create2 |
| | | | <div>yes</div> | | |

Exposed Functions

| | | | | |
|--|---|---------|------|------|
|  Public |  Payable | | | |
| 14 | 1 | | | |
| External | Internal | Private | Pure | View |
| 0 | 18 | 0 | 2 | 2 |

StateVariables

| Total |  Public |
|-------|--|
| 11 | 8 |



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

5.5 Dependencies / External Imports

| Dependency / Import Path | Source |
|---|---|
| @openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v5.3.0/contracts/access/AccessControlUpgradeable.sol |
| @openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v5.3.0/contracts/proxy/utils/UUPSUpgradeable.sol |
| @openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v5.3.0/contracts/token/ERC20/ERC20Upgradeable.sol |
| @openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC1363Upgradeable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v5.3.0/contracts/token/ERC20/extensions/ERC1363Upgradeable.sol |
| @openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC20PausableUpgradeable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v5.3.0/contracts/token/ERC20/extensions/ERC20PausableUpgradeable.sol |
| @openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC20PermitUpgradeable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v5.3.0/contracts/token/ERC20/extensions/ERC20PermitUpgradeable.sol |
| @openzeppelin/contracts-upgradeable/utils/MulticallUpgradeable.sol | https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v5.3.0/contracts/utils/MulticallUpgradeable.sol |
| @openzeppelin/contracts/access/IAccessControl.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v5.3.0/contracts/ |



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

| Dependency / Import Path | Source |
|---|---|
| @openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v5.3.0/contracts/proxy/ERC1967/ERC1967Proxy.sol |
| @openzeppelin/contracts/utils/introspection/ERC165Checker.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v5.3.0/contracts/utils/introspection/ERC165Checker.sol |

5.6 Source Unites in Scope

| File | Logic Contracts | Interfaces | Lines | nLines | nSLOC | Comment Lines |
|---|-----------------|------------|------------|------------|------------|---------------|
| ethereum/ethereum-contracts/contracts/AutProxy.sol | 1 | | 9 | 9 | 5 | 2 |
| ethereum/ethereum-contracts/contracts/BlacklistManagement.sol | 1 | | 58 | 56 | 36 | 8 |
| ethereum/ethereum-contracts/contracts/AllUnity.sol | 1 | | 133 | 121 | 77 | 21 |
| Totals | 3 | | 200 | 186 | 118 | 31 |

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



6. Scope of Work

The AllUnity team has provided the complete suite of smart contracts for their MiCAR-compliant stablecoin framework, including token logic, role management, blacklisting, and upgrade mechanisms. The audit is focused on validating security, compliance with institutional-grade requirements, and robustness across upgradeable multi-contract architecture.

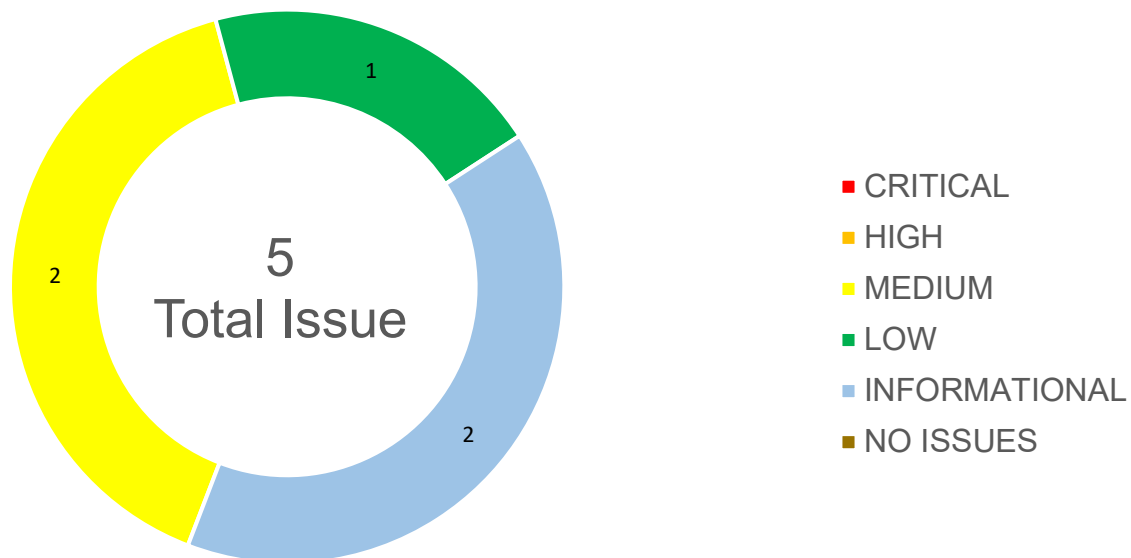
The team has outlined the following critical security and functionality assumptions for the AllUnity smart contracts:

1. **Blacklist Enforcement and Allowance Control**
The audit must confirm that blacklisted users are strictly prevented from sending, receiving, or minting tokens. The AllowanceManagement contract should be examined for secure and correctly enforced access control, ensuring that only authorized roles can assign or revoke blacklist status. Interactions with the AllUnity token must correctly respect blacklisting logic at all stages.
2. **Upgradeability and Proxy Security**
The contracts leverage UUPSUpgradeable and ERC1967Proxy standards. The audit will ensure upgrade permissions are strictly limited to authorized roles and that re-initialization and logic override vulnerabilities are prevented.
3. **Role-Based Access & Governance Separation**
The audit should verify that all roles ADMIN, PAUSER, MINTER, BLACKLISTER are separated according to least privilege principles. Role escalation and admin reassignment logic must not introduce unintended control paths or centralized power that could compromise token integrity or user protections.
4. **Token Lifecycle Operations and Security**
Token minting, burning, and pausing operations must be checked for proper enforcement of business logic and reentrancy safety. Transfers must halt during pause states, and minting must never occur to blacklisted addresses.
5. **Gas Optimization and Denial-of-Service Resilience**
The audit will assess the contracts for unnecessary gas overhead and patterns that could lead to DoS (e.g., overuse of loops in role changes or upgrade operations). Special attention will be given to edge-case handling, such as multiple rapid role updates, mass blacklisting, or proxy misconfigurations under load.

The primary goal of this audit will be to validate these assumptions and ensure that the contracts are secure, performant, and reliable. At the client's request, the audit team will also provide feedback on specific areas such as token economics or ecosystem integration.



6.1 Findings Overview



| No | Title | Severity | Status |
|-------|---|---------------|--------|
| 6.2.1 | Conditional Allowance Bypass in burnFrom for Blacklisted Accounts | MEDIUM | FIXED |
| 6.2.2 | Missing Storage Gap in AllUnity.sol | MEDIUM | FIXED |
| 6.2.3 | Missing Zero-Address Checks in initialize() | LOW | FIXED |
| 6.2.4 | Gas Limit Exhaustion in BlacklistManagement.batchAddBlacklist | INFORMATIONAL | FIXED |
| 6.2.5 | Missing batchRemoveBlacklist Function in BlacklistManagement.sol | INFORMATIONAL | FIXED |



6.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES

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

HIGH ISSUES

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

MEDIUM ISSUES

During the audit, softstack's experts found **two Medium issues** in the code of the smart contract.

6.2.1 Conditional Allowance Bypass in burnFrom for Blacklisted Accounts

Severity: MEDIUM

Status: FIXED

File(s) affected: AllUnity.sol

Update: <https://github.com/All-Unity/smart-contracts/pull/39/files>

We have a centralized governance system by design. We can see centralisation both in management of administrative operations as well as the burn mechanism around blacklisted funds. For us this is both a feature and a necessity since we are a regulated financial entity. For better segregation we decided to split both the roles and functions for two different burn mechanisms: burning with allowance (BURNER_ROLE, burnFrom), burning because blacklisted (BLACKLISTED_BURNER_FROM, burnFromBlacklistedAddress)

| | |
|-----------------------------|---|
| Attack / Description | <p>The burnFrom function, restricted to BURNER_ROLE, skips the ERC20 allowance check (<code>_spendAllowance</code>) if the from address is blacklisted. This allows the BURNER_ROLE to burn tokens from blacklisted accounts without their prior explicit approval via <code>approve()</code>.</p> <p>Impact: This is a powerful administrative feature. If intended for scenarios like seizing illicit funds from sanctioned accounts, it functions as designed. However, if the BURNER_ROLE is</p> |
|-----------------------------|---|



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

| | |
|------------------------------|--|
| | compromised or acts maliciously, or if an account is incorrectly blacklisted, this feature can be used to destroy user funds without their consent (via allowance). The <code>_update</code> hook correctly allows burning <i>from</i> a blacklisted address by <code>BURNER_ROLE</code> to <code>address(0)</code> . |
| Code | <p>Line 78 - 83 (AllUnity.sol):</p> <pre> function burnFrom(address from, uint256 amount) public onlyRole(BURNER_ROLE) { if (!blacklist.hasRole(BLACKLISTED_ROLE, from)) { _spendAllowance(from, _msgSender(), amount); } _burn(from, amount); } </pre> |
| Result/Recommendation | <p>Recommendation:</p> <ol style="list-style-type: none"> 1. Clearly and publicly document this behaviour as an intended administrative capability and its specific use cases. 2. Ensure the <code>BURNER_ROLE</code> is assigned to a highly secure entity (e.g., multi-sig with strict operational controls and clear accountability). 3. If this direct burn capability without allowance is not desired for all blacklisting scenarios, the if condition should be removed, forcing <code>_spendAllowance</code> to always be called. 4. Consider a multi-step approval process or specific timelocks for burns from blacklisted accounts to add oversight. |

6.2.2 Missing Storage Gap in AllUnity.sol

Severity: MEDIUM

Status: FIXED

File(s) affected: AllUnity.sol



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

Update: <https://github.com/All-Unity/smart-contracts/commit/4957b34c9d31e6ee17ecb9dcd780469f4cdad65e>

| | |
|------------------------------|--|
| Attack / Description | <p>Upgradeable contracts require reserved storage space (“storage gap”) to safely add new state variables in future implementations without shifting the existing storage layout. The AllUnity contract lacks such a reserved gap.</p> <p>Potential Impact: On upgrading to a new implementation that introduces additional state variables, storage slots may overlap with inherited variables, leading to inadvertent overwriting of critical state. This can corrupt token balances, role assignments, or pause state, potentially rendering the contract unusable or hijackable.</p> <p>Proof of Concept: 1. Deploy AllUnity and initialize. 2. Upgrade to a new version that adds, e.g., uint256 public newCounter; at the end of the contract. 3. Call functions interacting with storage (e.g., mint) and observe that newCounter shares slots with existing variables, causing unpredictable state changes.</p> <p>Likelihood of Exploitation: High if an upgrade is performed without careful manual slot management. Even if developers manage slots manually now, future contributors may overlook it, leading to severe breakage.</p> |
| Code | NA |
| Result/Recommendation | <p>We recommend to insert a reserved gap of at least 50 slots after the last declared variable, for example:</p> <p>// Reserved storage space to allow for layout changes in the future.</p> <p>uint256[50] private __gap;</p> |



| | |
|--|---|
| | } |
|--|---|

LOW ISSUES

During the audit, softstack's experts found **one Low issue** in the code of the smart contract

6.2.3 Missing Zero-Address Checks in initialize()

Severity: LOW

Status: FIXED

File(s) affected: AllUnity.sol, BlacklistManagement.sol

Update:

<https://github.com/All-Unity/smart-contracts/commit/b0d43d573345e64aa6b7c9718b5aabe7cca4139e>

<https://github.com/All-Unity/smart-contracts/commit/2f76d00539abc6fc2afa92042df3a804829a0334>

<https://github.com/All-Unity/smart-contracts/commit/fcbf6e014d6eccc0a8c032b612b4c76c3709e316>

| Attack / Description | <p>In the AllUnity.sol initialize() and the BlacklistManagement.sol initialize() functions the parameters _adminAddress and _minterAddress aswell as adminAddress and blacklisterAddress are used to grant critical roles without validating that they are non-zero addresses.</p> <p>Potential Impact Granting roles to address(0) can lock administrators or minters out of the system (denial of service). If wrongly initialized by an attacker before legitimate initialization, roles could be improperly assigned, leading to lost control over pausing, minting, or burning.</p> <p>Proof of Concept</p> <ol style="list-style-type: none"> 1. Attacker deploys proxy but passes zero addresses 2. DEFAULT_ADMIN_ROLE, MINTER_ROLE and BLACKLISTER_ROLE now assigned to zero address. 3. No real account can pause, upgrade, or mint—contract becomes unusable. |
|----------------------|--|
|----------------------|--|



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

| | |
|------|--|
| | <p>4. Blacklisting address(0) could break any logic that assumes the zero address is never blacklisted (e.g. burnFrom).</p> <p>Likelihood of Exploitation</p> <p>Low, while initialization is a one-time operation, misconfiguration mistakes can happen, especially in complex deployment processes. An attacker observing the uninitialized proxy could front-run with zero inputs if the deployer forgets checks.</p> |
| Code | <p>Line 31 (AllUnity.sol):</p> <pre>function initialize(string memory _name, string memory _symbol, uint8 _decimals, address _adminAddress, address _minterAddress</pre> <p>Line 19 (BlacklistManagement.sol):</p> <pre>function initialize(address adminAddress, address blacklisterAddress) initializer public { __AccessControl_init(); __UUPSUpgradeable_init(); __Multicall_init(); _grantRole(DEFAULT_ADMIN_ROLE, adminAddress); _grantRole(BLACKLISTER_ROLE, blacklisterAddress); _setRoleAdmin(BLACKLISTED_ROLE, BLACKLISTER_ROLE);</pre> |

| | |
|------------------------------|--|
| | } |
| Result/Recommendation | <p>We recommend to add require statements to enforce non-zero addresses at the top of the initialize() functions:</p> <pre>require(_adminAddress != address(0), "AllUnity: admin address is zero"); require(_minterAddress != address(0), "AllUnity: minter address is zero"); require(adminAddress != address(0), "BlacklistManagement: admin address is zero"); require(blacklisterAddress != address(0), "BlacklistManagement: blacklister address is zero");</pre> |

INFORMATIONAL ISSUES

During the audit, softstack's experts found **two Informational issues** in the code of the smart contract

6.2.4 Gas Limit Exhaustion in BlacklistManagement.batchAddBlacklist

Severity: INFORMATIONAL

Status: FIXED

File(s) affected: BlacklistManagement.sol

Update: <https://github.com/All-Unity/smart-contracts/commit/bbffb0d86a3066dd9d2a40a300da25f87dff6051>

| | |
|-----------------------------|--|
| Attack / Description | <p>The batchAddBlacklist function iterates through the accounts array, calling <code>_grantRole</code> for each. Each <code>_grantRole</code> is a state-changing operation involving storage writes and event emissions, making it gas-intensive. If the accounts array is very large, the transaction's cumulative gas cost can exceed the block gas limit, causing it to fail.</p> <p>Impact: Prevents the blacklisting of a large number of addresses in a single transaction, diminishing the utility of the batch function for its intended purpose in such scenarios and potentially requiring manual splitting into smaller transactions during time-sensitive situations.</p> |
|-----------------------------|--|



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

| | |
|------------------------------|---|
| Code | <p>Line 43 - 45 (BlacklistManagement.sol):</p> <pre>function batchAddBlacklist(address[] calldata accounts) public onlyRole(BLACKLISTER_ROLE) { for (uint256 i = 0; i < accounts.length; ++i) _grantRole(BLACKLISTED_ROLE, accounts[i]); }</pre> |
| Result/Recommendation | <p>We recommend the following action items:</p> <ol style="list-style-type: none"> 1. Define a public constant MAX_BATCH_SIZE (e.g., 50-100, to be determined by gas profiling). 2. Add a require(accounts.length <= MAX_BATCH_SIZE, "Batch size exceeds maximum limit"); at the beginning of the function. 3. Clearly document this limitation and advise users to split larger lists or use off-chain scripts for batching. |

6.2.5 Missing batchRemoveBlacklist Function in BlacklistManagement.sol

Severity: INFORMATIONAL

Status: FIXED

File(s) affected: BlacklistManagement.sol

Update: <https://github.com/All-Unity/smart-contracts/pull/41>

| | |
|-----------------------------|--|
| Attack / Description | <p>The contract includes batchAddBlacklist for adding multiple addresses to the blacklist efficiently but lacks a corresponding batchRemoveBlacklist function.</p> |
|-----------------------------|--|



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984


| | |
|------------------------------|---|
| | Impact: Removing multiple addresses from the blacklist requires individual transactions, leading to operational inefficiency and higher gas costs compared to a batch operation. |
| Code | <p>Line 43 - 45 (BlacklistManagement.sol):</p> <pre>function batchAddBlacklist(address[] calldata accounts) public onlyRole(BLACKLISTER_ROLE) { for (uint256 i = 0; i < accounts.length; ++i) _grantRole(BLACKLISTED_ROLE, accounts[i]); }</pre> |
| Result/Recommendation | <p>We recommend to implement a batchRemoveBlacklist(address[] calldata accounts) function, restricted to BLACKLISTER_ROLE, that iterates through the array and calls _revokeRole(BLACKLISTED_ROLE, account). This function should also incorporate gas limit considerations similar to batchAddBlacklist.</p> |



6.3 Verify Claims


6.3.1 Blacklist Enforcement and Allowance Control

The audit must confirm that blacklisted users are strictly prevented from sending, receiving, or minting tokens. The AllowanceManagement contract should be examined for secure and correctly enforced access control, ensuring that only authorized roles can assign or revoke blacklist status. Interactions with the AllUnity token must correctly respect blacklisting logic at all stages.

Status: tested and verified 


6.3.2 Upgradeability and Proxy Security

The contracts leverage UUPSUpgradeable and ERC1967Proxy standards. The audit will ensure upgrade permissions are strictly limited to authorized roles and that re-initialization and logic override vulnerabilities are prevented.

Status: tested and verified 


6.3.3 Role-Based Access & Governance Separation

The audit should verify that all roles ADMIN, PAUSER, MINTER, BLACKLISTER are separated according to least privilege principles. Role escalation and admin reassignment logic must not introduce unintended control paths or centralized power that could compromise token integrity or user protections.

Status: tested and verified 


6.3.4 Token Lifecycle Operations and Security

Token minting, burning, and pausing operations must be checked for proper enforcement of business logic and reentrancy safety. Transfers must halt during pause states, and minting must never occur to blacklisted addresses.

Status: tested and verified 

6.3.5 Gas Optimization and Denial-of-Service Resilience

The audit will assess the contracts for unnecessary gas overhead and patterns that could lead to DoS (e.g., overuse of loops in role changes or upgrade operations). Special attention will be given to edge-case handling, such as multiple rapid role updates, mass blacklisting, or proxy misconfigurations under load.

Status: tested and verified 



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

7. Executive Summary

Two independent softstack experts performed an unbiased and isolated audit of the smart contract provided by the AllUnity team. The main objective of the audit was to verify the security and functionality claims of the smart contract. The audit process involved a thorough manual code review and automated security testing.

Overall, the audit identified a total of one issue, classified as follows:

- No critical issues were found.
- No high severity issues were found.
- 2 medium severity issues were found.
- 1 low severity issues were discovered
- 2 informational issues were identified

The audit report provides detailed descriptions of each identified issue, including severity levels, proof of concepts and recommendations for mitigation. It also includes code snippets, where applicable, to demonstrate the issues and suggest possible fixes. We recommend the AllUnity team to review the suggestions.

Update (15.05.2025): The AllUnity team has successfully addressed all identified issues from the audit. All medium, low and informational-severity vulnerabilities have been mitigated based on the recommendations provided in the report. A follow-up review confirms that the fixes have been implemented effectively, ensuring the security and functionality of the smart contract. The updated codebase reflects the necessary improvements, and no further critical concerns remain.



8. About the Auditor

Established in 2017 under the name Chainsulting, and rebranded as softstack GmbH in 2023, softstack has been a trusted name in Web3 Security space. Within the rapidly growing Web3 industry, softstack provides a comprehensive range of offerings that include software development, cybersecurity, and consulting services. Softstack's competency extends across the security landscape of prominent blockchains like Solana, Tezos, TON, Ethereum and Polygon. The company is widely recognized for conducting thorough code audits aimed at mitigating risk and promoting transparency.

The firm's proficiency lies particularly in assessing and fortifying smart contracts of leading DeFi projects, a testament to their commitment to maintaining the integrity of these innovative financial platforms. To date, softstack plays a crucial role in safeguarding over \$100 billion worth of user funds in various DeFi protocols.

Underpinned by a team of industry veterans possessing robust technical knowledge in the Web3 domain, softstack offers industry-leading smart contract audit services. Committed to evolving with their clients' ever-changing business needs, softstack's approach is as dynamic and innovative as the industry it serves.

Check our website for further information: <https://softstack.io>

How We Work



1 -----

PREPARATION

Supply our team with audit ready code and additional materials



2 -----

COMMUNICATION

We setup a real-time communication tool of your choice or communicate via e-mails.



3 -----

AUDIT

We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.



4 -----

FIXES

Your development team applies fixes while consulting with our auditors on their safety.



5 -----

REPORT

We check the applied fixes and deliver a full report on all steps done.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984