



Fetch AI

Agentverse Launchpad

SMART CONTRACT AUDIT

05.05.2025

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 Call Graph.....	10
5.4 Source Lines & Risk.....	11
5.5 Capabilities	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 Signature Malleability Vulnerability in Multi-Sig Implementation	16
6.2.2 Arbitrary Pricing Divisor Vulnerability in FETAgentCoin Contract	18
6.2.3 Hardcoded Liquidity Target Vulnerability in FETAgentCoin	19
6.2.4 Unbounded ERC20 Approval Vulnerability in FETAgentCoin Constructor.....	20
6.3 Verify Claims	23
7. Executive Summary.....	24



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 FETCH.AI 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 (23.04.2025)	Layout
0.4 (28.04.2025)	Automated Security Testing Manual Security Testing
0.5 (29.04.2025)	Verify Claims
0.9 (02.05.2025)	Summary and Recommendation
1.0 (05.05.2025)	Final document



2. About the Project and Company

Company address:

FETCH.AI LIMITED
50-60 Station Road
Cambridgeshire, United Kingdom



Website (Agentverse): <https://agentverse.ai>

Website (Fetch AI): <https://fetch.ai>

Twitter (X): https://twitter.com/Fetch_ai

Telegram: https://t.me/fetch_ai

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

GitHub: <https://github.com/fetchai>

YouTube: <https://www.youtube.com/fetchai>

LinkedIn: <https://www.linkedin.com/company/fetch-ai/>

Reddit: https://www.reddit.com/r/FetchAI_Community/?rdt=52803



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

2.1 Project Overview

Fetch.ai is an open decentralized machine-to-machine ecosystem that combines AI, blockchain, and multi-agent systems to enable autonomous services across industries. It provides a foundational infrastructure for building intelligent agents capable of decentralized communication, negotiation, and transaction execution without human intervention.

At its core, Fetch.ai leverages open-source components including the Fetch.ai SDK and uAgents framework, allowing developers to easily create and manage AI agents. These agents are deployed and discovered across the network through a modular architecture consisting of the Universal Agent Gateway, Almanac AI Register, and Aname, all operating atop a decentralized blockchain layer. This structure ensures full transparency, resilience, and auditability in agent-to-agent interactions.

Fetch.ai offers both a free AI discovery tool and a commercial subscription layer for optimized AI interactions, making it accessible for experimentation while scalable for production-grade deployments.

AgentVerse is a decentralized development, management, and trading hub within the Fetch.ai ecosystem. It allows users to create, deploy, register, and trade agent-specific tokens (AgentCoins) through a smart contract-based launchpad, the FETCH AgentVerse.

This enables agents to become economically autonomous entities, each represented by its own token with defined utility, price dynamics, and liquidity management. Through its bonding curve mechanism and strict wallet/lifecycle controls, the AgentVerse ensures a fair launch environment for each agent.



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/fetchai/launchpadDAO>

Commit: 9552b3d052c68d288b98dad28e2fc447bc7a18b9

File	Fingerprint (MD5)
./smart_contracts/IERC20.sol	df36f7051335cd1e748b1b6463b7fdd3
./smart_contracts/ERC20.sol	3ec4ff687811242bc5dfc6d61e9cb1ff
./smart_contracts/FETAgentVerseDeployer.sol	18c809b0e432c5ad58db1090182d7a45
./smart_contracts/IUniswapV2Router.sol	cc6ae8df92c60a7043e4eab26cc80df2
./smart_contracts/FETAgentcoin.sol	5c159c1df0baacc7da38d4530b1edd0
./smart_contracts/ECDSA.sol	81de029d56aa803972be03c5d277cb6c
./smart_contracts/IERC20Metadata.sol	7d180a51f2ae3e249b0aaee46c94c996
./smart_contracts/IUniswapV2Factory.sol	a758bbf5d0ff8f11575fa8d2b6420d66
./smart_contracts/Context.sol	5f2c5c4b6af2dd4551027144797bc8be



hello@softstack.io
www.softstack.io

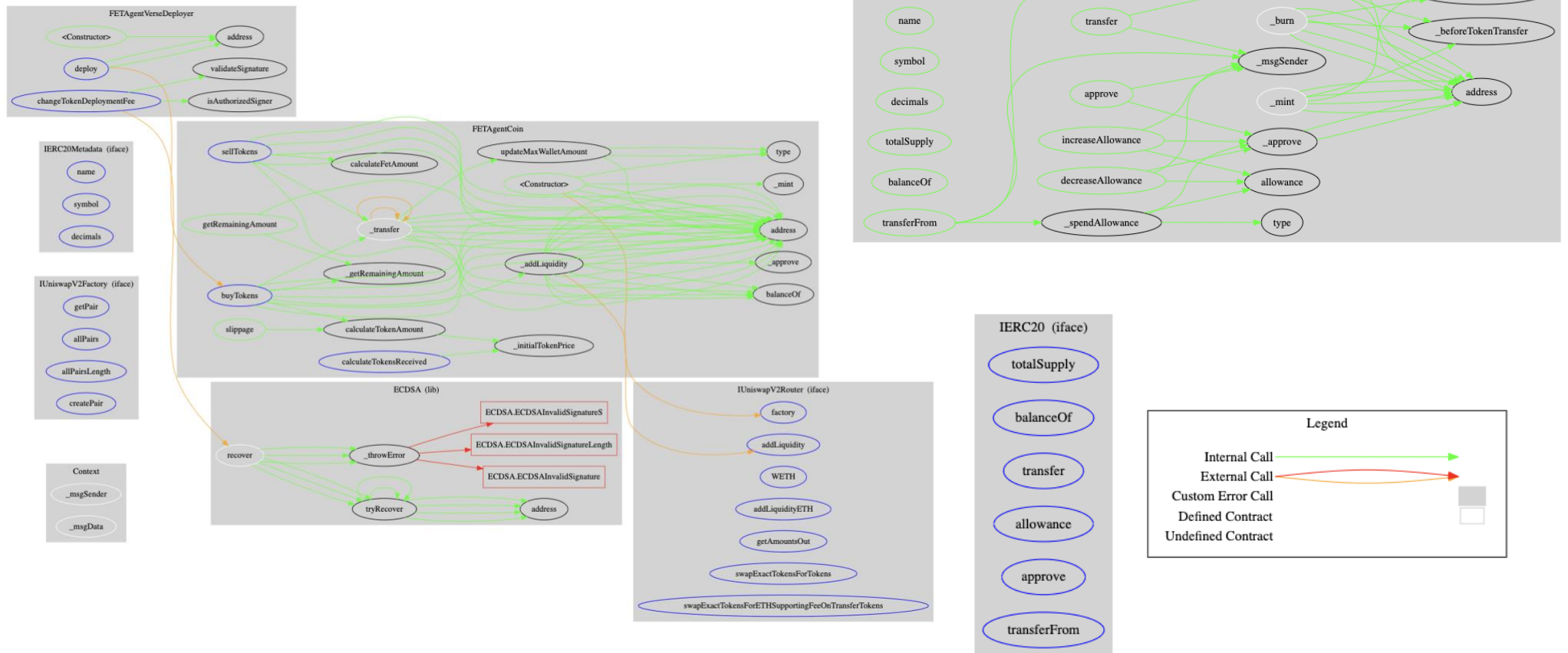
softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

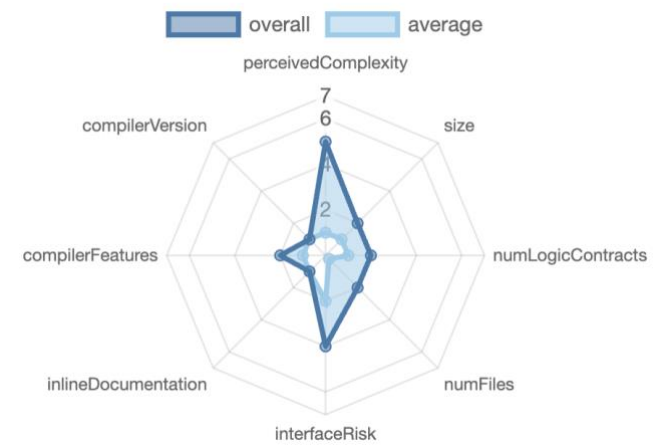
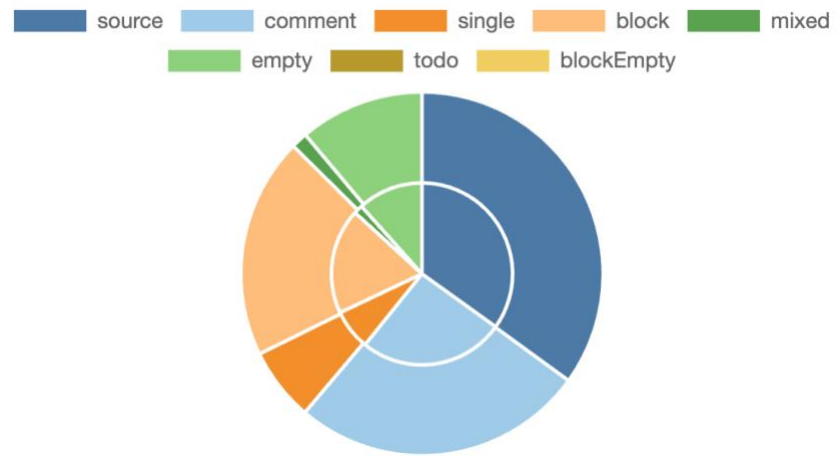
5.2 Inheritance Graph



5.3 Call Graph



5.4 Source Lines & Risk



hello@softstack.io
www.softstack.io



softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984


5.5 Capabilities

Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts	
<div>^0.8.0</div> <div>^0.8.20</div> <div>0.8.26</div>				<div>yes</div>		<div>yes</div> <div>(2 asm blocks)</div>			
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECTrecover	
<div>yes</div>						<div>yes</div>		<div>yes</div> <div>→ NewContract:FETAgentCoin</div>	

Exposed Functions

 Public	 Payable			
39	1			
External	Internal	Private	Pure	View
25	52	1	11	25

StateVariables

Total	 Public
35	25



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

5.6 Source Unites in Scope

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines
smart_contracts/Context.sol	1		24	24	9	12
smart_contracts/IUniswapV2Factory.sol		1	10	6	3	1
smart_contracts/IERC20Metadata.sol		1	28	17	4	16
smart_contracts/ECDSA.sol	1		180	168	70	85
smart_contracts/FETAgentcoin.sol	1		347	347	214	64
smart_contracts/IUniswapV2Router.sol		1	42	5	3	1
smart_contracts/FETAgentVerseDeployer.sol	1		204	196	140	24
smart_contracts/ERC20.sol	1		366	366	115	212
smart_contracts/IERC20.sol		1	78	38	16	58
Totals	5	4	1279	1167	574	473

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



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

6. Scope of Work

The Fetch.ai team has provided the full set of smart contracts for the Agentverse platform. The audit is focused on verifying the security, efficiency, and correctness of the token creation, bonding curve sale, and automated listing mechanics, ensuring they are robust against potential vulnerabilities.

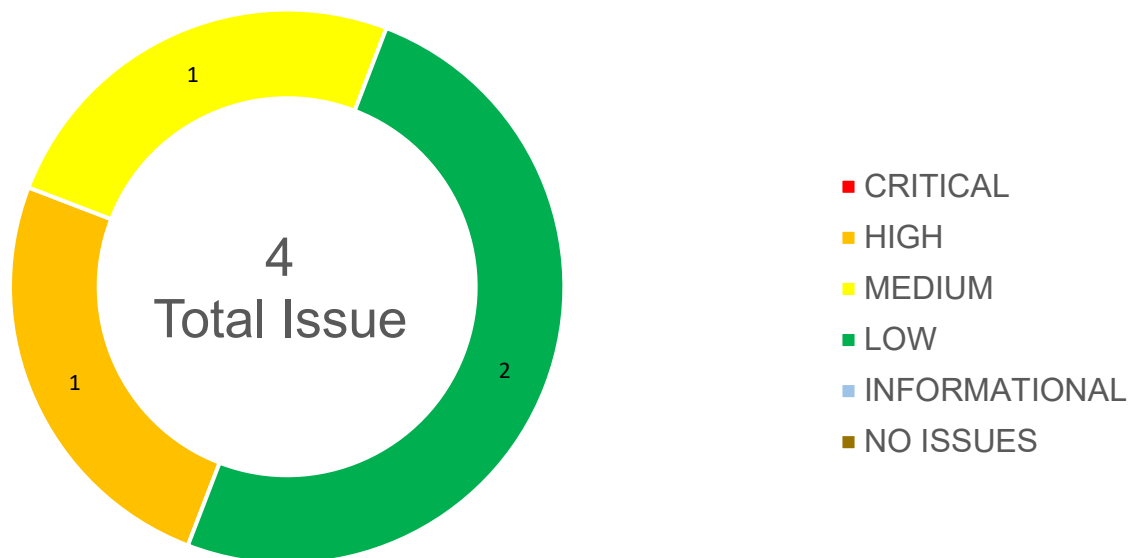
The team has outlined the following security and functionality assumptions for the smart contracts:

1. **Token Sale Integrity and Liquidity Provisioning**
The audit should confirm that AgentCoin tokens are deployed with accurate bonding curve pricing and that pre-listing purchase and transfer logic is strictly enforced.
2. **Secure Fund Flow and Fee Management**
The contract must securely handle FET tokens used for purchases, correctly deducting and routing fees to the revenue account. The audit will verify that fee calculations are precise and that no reentrancy or fund-drain vulnerabilities exist during token buys, sells, or liquidity provisioning.
3. **Robust Access Control and Governance**
The audit should verify that only authorized multisig signers can modify deployment parameters (such as deployment fees), and that no privileged actions can interfere with user funds or token logic.
4. **Gas Efficiency and Economic Precision**
The bonding curve pricing formula, token amount calculations, and max wallet enforcement must be evaluated for computational efficiency.
5. **Edge Case Resilience and Attack Mitigation**
The contract should be resilient to non-standard inputs, including extreme token or FET values, repeated buy/sell cycles near listing threshold, or exploit attempts via LP manipulation.

The primary goal of this audit will be to validate these assumptions and ensure that the Fetch.ai Agentverse 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	Signature Malleability Vulnerability in Multi-Sig Implementation	HIGH	FIXED
6.2.2	Arbitrary Pricing Divisor Vulnerability in FETAgentCoin Contract	MEDIUM	FIXED
6.2.3	Hardcoded Liquidity Target Vulnerability in FETAgentCoin	LOW	ACKNOWLEDGED
6.2.4	Unbounded ERC20 Approval Vulnerability in FETAgentCoin Constructor	LOW	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 **one High issue** in the code of the smart contract.

6.2.1 Signature Malleability Vulnerability in Multi-Sig Implementation

Severity: HIGH

Status: FIXED

File(s) affected: FETAgentVerseDeployer.sol

Update: <https://github.com/fetchai/launchpadDAO/commit/6c5fb0327ebb678c397ce0d5cf22bea139747e0d>

Attack / Description	<p>The multi-signature authorization mechanism contains a critical flaw in signature verification that allows signature malleability attacks. This violates EIP-2 and EIP-2098 standards, enabling attackers to create valid signature duplicates for unauthorized parameter changes.</p> <p>The vulnerability stems from three key issues:</p> <ol style="list-style-type: none">1. Malleable Signature Components // Vulnerable pattern: ecrecover(hash, v, r, s); // Attackable through: - s-value > secp256k1n/2 (CVE-2020-14762) - v-value toggling between 27/282. Missing EIP-2098 Validation
-----------------------------	--



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

	<p>Compact signature format not enforced, allowing multiple valid representations of the same signature.</p> <p>3. Nonce Mismanagement Single nonce increment allows multiple valid signatures for same operation.</p>
Code	<p>Line 193 – 206 (FETAgentVerseDeployer.sol):</p> <pre>// FETAgentVerseDeployer.sol function recoverSigner(bytes32 hash, bytes memory sig) internal pure returns (address) { require(sig.length == 65, "Invalid signature length"); bytes32 r; bytes32 s; uint8 v; assembly { r := mload(add(sig, 0x20)) s := mload(add(sig, 0x40)) v := byte(0, mload(add(sig, 0x60))) } return ecrecover(hash, v, r, s); }</pre>
Result/Recommendation	<p>We recommend the following three changes:</p> <p>Implement ECDSA Library</p> <pre>import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol"; function recoverSigner(bytes32 hash, bytes memory sig) internal pure returns (address) { return ECDSA.recover(hash, sig); }</pre> <p>Add Signature Tracking</p>

```
mapping(bytes32 => bool) public usedSignatures;

function changeTokenDeploymentFee(...) external {
    bytes32 sigHash = keccak256(signatures[i]);
    require(!usedSignatures[sigHash], "Signature reused");
    usedSignatures[sigHash] = true;
}

Enforce Compact Signatures
function validateSignature(bytes memory sig) internal pure {
    require(sig.length == 64 || sig.length == 65, "Invalid signature");
    if (sig.length == 65) {
        require(uint8(sig[64]) == 27 || uint8(sig[64]) == 28, "Invalid v value");
    }
}
```

MEDIUM ISSUES

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

6.2.2 Arbitrary Pricing Divisor Vulnerability in FETAgentCoin Contract

Severity: MEDIUM

Status: FIXED

File(s) affected: FETAgentcoin.sol

Update: <https://github.com/fetchai/launchpadDAO/commit/9552b3d052c68d288b98dad28e2fc447bc7a18b9>

Attack / Description

The FETAgentCoin contract contains an undocumented hard-coded divisor (6) in its core pricing formula, creating an unvalidated economic model vulnerable to manipulation. This violates best practices for mathematical constants in financial smart contracts.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

Code	<p>Line 254 (FETAgentcoin.sol):</p> <pre> uint256 tokenPrice = initialTokenPrice + ((initialTokenPrice * BUY_PRICE_DIFFERENCE_PERCENT / 100) * (totalFETBought + (fetAmount / 6)) // <- Arbitrary divisor / TARGET_LIQUIDITY); </pre>
Result/Recommendation	<p>We recommend making the following two changes:</p> <p>Parameterization: uint256 public PRICE_SCALING_FACTOR = 6; // Make configurable</p> <p>Documentation: Improve code clarity by adding NatSpec-style inline documentation that explains the rationale behind this parameter:</p> <pre> /// @dev Scaling factor derived from [Reference to economic model] /// Ensures 1 FET spent increases price by X% at Y liquidity level </pre>

LOW ISSUES

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

6.2.3 Hardcoded Liquidity Target Vulnerability in FETAgentCoin

Severity: LOW

Status: ACKNOWLEDGED

File(s) affected: FETAgentCoin.sol

Update: The values are not intended to be changed in the future.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

Attack / Description	The contract uses a hardcoded value for liquidity target calculations, creating protocol incompatibility when parameters change. This violates maintainability best practices and could introduce system miscalculations.
Code	<p>Line 262 - 264 (FETAgentCoin.sol):</p> <pre>// FETAgentCoin.sol function calculateTokensReceived(uint256 fetAmount) external view returns (uint256) { uint256 target_fet = 30612244897959183673469; // Hardcoded value require(totalFETBought + fetAmount <= target_fet, "Liquidity target exceeded"); // ... }</pre>
Result/Recommendation	<p>We recommend to implement dynamic calculation:</p> <pre>function calculateTokensReceived(uint256 fetAmount) external view returns (uint256) { uint256 target_fet = TARGET_LIQUIDITY * 100 / (100 - uint256(FEE_PERCENTAGE)); require(totalFETBought + fetAmount <= target_fet, "Liquidity target exceeded"); // ... }</pre>

6.2.4 Unbounded ERC20 Approval Vulnerability in FETAgentCoin Constructor

Severity: LOW

Status: FIXED

File(s) affected: FETAgentCoin.sol

Update: The contract grants infinite ERC20 token allowance to the Uniswap router during initialization would be revised to the `neededSum`.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984


Attack / Description	The contract grants infinite ERC20 token allowance to the Uniswap router during initialization, creating a persistent attack vector if the router becomes compromised. This violates the principle of least privilege in smart contract security.
Code	<p>Line 83 (FETAgentCoin.sol):</p> <pre>// FETAgentCoin.sol constructor(...) { _approve(address(this), address(uniswapRouter), type(uint256).max); }</pre>
Result/Recommendation	<p>We recommend to implement temporary approvals with renewal system:</p> <p>Time-Bound Approvals</p> <pre>uint256 public approvalExpiry; function refreshApproval() internal { if (block.timestamp > approvalExpiry) { _approve(address(this), router, TOTAL_BUY_TOKENS); approvalExpiry = block.timestamp + 1 days; } }</pre> <p>Exact Approval Pattern</p> <pre>function _addLiquidity() internal { uint256 needed = balanceOf(address(this)); _approve(address(this), router, needed); // Reset to zero after use _approve(address(this), router, 0); }</pre> <p>Permit Integration</p>

```
function initializeWithPermit(  
    bytes memory permitSig,  
    uint256 deadline  
) external {  
    IERC20Permit(address(this)).permit(  
        address(this),  
        router,  
        type(uint256).max,  
        deadline,  
        permitSig  
    );  
}
```



6.3 Verify Claims


6.3.1 Token Sale Integrity and Liquidity Provisioning The audit should confirm that AgentCoin tokens are deployed with accurate bonding curve pricing and that pre-listing purchase and transfer logic is strictly enforced.

Status: tested and verified 


6.3.2 Secure Fund Flow and Fee Management The contract must securely handle FET tokens used for purchases, correctly deducting and routing fees to the revenue account. The audit will verify that fee calculations are precise and that no reentrancy or fund-drain vulnerabilities exist during token buys, sells, or liquidity provisioning.

Status: tested and verified 


6.3.3 Robust Access Control and Governance The audit should verify that only authorized multisig signers can modify deployment parameters (such as deployment fees), and that no privileged actions can interfere with user funds or token logic.

Status: tested and verified 

6.3.4 Gas Efficiency and Economic Precision The bonding curve pricing formula, token amount calculations, and max wallet enforcement must be evaluated for computational efficiency.

Status: tested and verified 

6.3.5 Edge Case Resilience and Attack Mitigation The contract should be resilient to non-standard inputs, including extreme token or FET values, repeated buy/sell cycles near listing threshold, or exploit attempts via LP manipulation.

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 Fetch AI 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.
- 1 high severity issues were found.
- 1 medium severity issues were found.
- 2 low severity issues were discovered
- No 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 that the Fetch AI team review the suggestions.

Update (03.05.2025): The Fetch AI team has successfully addressed all identified issues from the audit. All high, medium, and low-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