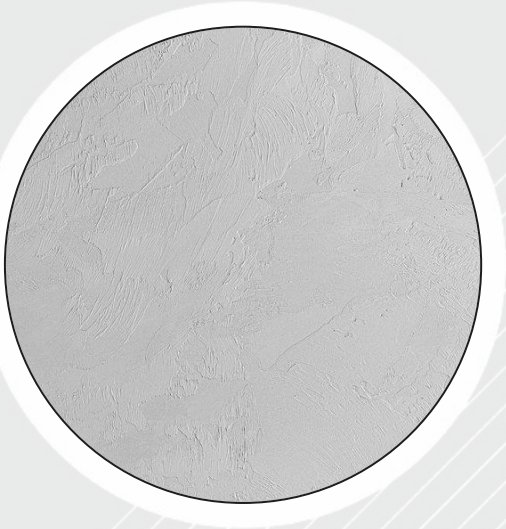
SPYWOLF

Security Audit Report



Audit prepared for

Signum

Completed on

November 26, 2024

OVERVIEW

This goal of this report is to review the main aspects of the project to help investors make an informative decision during their research process.

You will find a a summarized review of the following key points:

- ✔ Contract's source code
- ✔ Owners' wallets
- ✔ Tokenomics
- ✔ Team transparency and goals
- ✔ Website's age, code, security and UX
- ✔ Whitepaper and roadmap
- ✔ Social media & online presence

The results of this audit are purely based on the team’s evaluation and does not guarantee nor reflect the projects outcome and goal

- SPYWOLF Team -

TABLE OF CONTENTS

Autopay Contract

Token Name
Unavailable

Symbol
Unavailable

Contract Address
Unavailable

Network
Unavailable

Language
Solidity

Deployment Date
Not deployed yet

Contract Type
Tipping/Rewards

Total Supply
Unavailable

Decimals
Unavailable

TAXES

Buy Tax
0%

Sell Tax
0%

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

VULNERABILITY ANALYSIS

ID	Title	
SWC-100	Function Default Visibility	**Passed**
SWC-101	Integer Overflow and Underflow	**Passed**
SWC-102	Outdated Compiler Version	**Passed**
SWC-103	Floating Pragma	**Passed**
SWC-104	Unchecked Call Return Value	**Passed**
SWC-105	Unprotected Ether Withdrawal	**Passed**
SWC-106	Unprotected SELFDESTRUCT Instruction	**Passed**
SWC-107	Reentrancy	**Passed**
SWC-108	State Variable Default Visibility	**Passed**
SWC-109	Uninitialized Storage Pointer	**Passed**
SWC-110	Assert Violation	**Passed**
SWC-111	Use of Deprecated Solidity Functions	**Passed**
SWC-112	Delegatecall to Untrusted Callee	**Passed**
SWC-113	DoS with Failed Call	**Passed**
SWC-114	Transaction Order Dependence	**Passed**
SWC-115	Authorization through tx.origin	**Passed**
SWC-116	Block values as a proxy for time	**Passed**
SWC-117	Signature Malleability	**Passed**
SWC-118	Incorrect Constructor Name	**Passed**

VULNERABILITY ANALYSIS

ID	Title	
SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-135	Code With No Effects	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed

VULNERABILITY ANALYSIS

NO ERRORS FOUND

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

Governance Contract

Token Name
Unavailable

Symbol
Unavailable

Contract Address
Unavailable

Network
Unavailable

Language
Solidity

Deployment Date
Not deployed yet

Contract Type
Governance

Total Supply
Unavailable

Decimals
Unavailable

TAXES

Buy Tax
0%

Sell Tax
0%

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

'teamMultisig' address can change the 'privateFactoryAddress'.

```
function updatePrivateFactoryAddress(address _privateFactoryAddress) external {
    require(msg.sender == teamMultisig, "only the team can update the factory deploy address");
    privateFactoryAddress = _privateFactoryAddress;

    emit PrivateFactoryUpdated(_privateFactoryAddress);
}
```

'privateFactoryAddress' can add addresses to the 'privateOracleAddresses' mapping

```
function addPrivateOracle(address _privateOracleAddress) external {
    require(msg.sender == privateFactoryAddress, "only the factory can add oracles to be governed");
    privateOracleAddresses[_privateOracleAddress] = true;

    emit PrivateOracleAdded(_privateOracleAddress);
}
```

FOUND THREATS

Informational

Private disputes can be opened only by addresses mapped as ’privateOracleAddresses’.

```
function beginDisputePrivate(IOracle _oracle, bytes32 _queryId, uint256 _timestamp) external {
    require(privateOracleAddresses[address(_oracle)], "Oracle address must be a private signum oracle.");
    // Ensure value actually exists
    address _reporter = _oracle.getReporterByTimestamp(_queryId, _timestamp);
    require(_reporter != address(0), "no value exists at given timestamp");
    bytes32 _hash = keccak256(abi.encodePacked(_queryId, _timestamp));
    // Push new vote round
    uint256 _disputeId = voteCount + 1;
    uint256[] storage _voteRounds = voteRounds[_hash];
    _voteRounds.push(_disputeId);

    // Create new vote and dispute
    Vote storage _thisVote = voteInfo[_disputeId];
    Dispute storage _thisDispute = disputeInfo[_disputeId];

    // Initialize dispute information - query ID, timestamp, value, etc.
    _thisDispute.queryId = _queryId;
    _thisDispute.timestamp = _timestamp;
    _thisDispute.disputedReporter = _reporter;
    // Initialize vote information - hash, initiator, block number, etc.
    _thisVote.identifierHash = _hash;
    _thisVote.initiator = msg.sender;
    _thisVote.blockNumber = block.number;
    _thisVote.startDate = block.timestamp;
    _thisVote.voteRound = _voteRounds.length;
    disputeIdsByReporter[_reporter].push(_disputeId);
    ..................................
}
```

SignumFlex PrivateFactory

Token Name
Unavailable

Symbol
Unavailable

Contract Address
Unavailable

Network
Unavailable

Language
Solidity

Deployment Date
Not deployed yet

Contract Type
Factory

Total Supply
Unavailable

Decimals
Unavailable

TAXES

Buy Tax
0%

Sell Tax
0%

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Owner can set creation fee and fee receiver.

```
function updateFeeSettings(uint256 _creationFee, address _feeReceiver) external onlyOwner {
    creationFee = _creationFee;
    feeReceiver = _feeReceiver;
}
```

Owner can set 'stakeAmount' up to 75,000 tokens and 'stakeLockTime' up to 10 years.
These variables are read and used by the SignumFlexPrivate contracts.

```
function updateStakeRequirements(uint256 _stakeAmount, uint256 _stakeLockTime) external onlyOwner {
    stakeAmount = _stakeAmount;
    stakeLockTime = _stakeLockTime;
    require(stakeAmount <= 75000e18, "75K Max");
    require(stakeLockTime <= 315360000, "10 Year max.");
    emit StakeRequirementsUpdated(_stakeAmount, _stakeLockTime);
}
```

FOUND THREATS

Informational

Users can deploy new SignumFlexPrivate contracts. SignumFlexPrivate contracts are added in 'privateOracleAddresses' mapping in the governance contract, allowing them to use the 'beginDisputePrivate' functionality.

```
function deploySignumFlexPrivate() external payable returns (address newContractAddress) {
    // Transfer creationFee
    if (creationFee > 0) {
    IERC20(feeToken).transferFrom(msg.sender, feeReceiver, creationFee);
    }

    // Deploy a new instance of the SignumFlexPrivate contract
    SignumFlexPrivate newContract = new SignumFlexPrivate(address(this));

    // Store the address of the new contract in the deployedContracts array
    deployedContracts.push(address(newContract));
    isPrivateOracle[address(newContract)] = true;
    governance.addPrivateOracle(address(newContract));

    // Emit the ContractDeployed event
    emit PrivateOracleDeployed(address(newContract), msg.sender);

    // Return the address of the newly deployed contract
    return address(newContract);
}
```

FOUND THREATS

Informational

Users can deploy new SignumFlexPrivate contracts. SignumFlexPrivate contracts are added in 'privateOracleAddresses' mapping in the governance contract, allowing them to use the 'beginDisputePrivate' functionality.

```
function deploySignumFlexPrivate() external payable returns (address newContractAddress) {
    // Transfer creationFee
    if (creationFee > 0) {
    IERC20(feeToken).transferFrom(msg.sender, feeReceiver, creationFee);
    }

    // Deploy a new instance of the SignumFlexPrivate contract
    SignumFlexPrivate newContract = new SignumFlexPrivate(address(this));

    // Store the address of the new contract in the deployedContracts array
    deployedContracts.push(address(newContract));
    isPrivateOracle[address(newContract)] = true;
    governance.addPrivateOracle(address(newContract));

    // Emit the ContractDeployed event
    emit PrivateOracleDeployed(address(newContract), msg.sender);

    // Return the address of the newly deployed contract
    return address(newContract);
}
```

SignumFlex Private

Token Name
Unavailable

Symbol
Unavailable

Contract Address
Unavailable

Network
Unavailable

Language
Solidity

Deployment Date
Not deployed yet

Contract Type
Factory

Total Supply
Unavailable

Decimals
Unavailable

TAXES

Buy Tax
0%

Sell Tax
0%

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ Medium Risk

If _lockedBalance is higher than IFactory(factory).stakeAmount() overflow will occur, causing the transaction to revert.

Overflow might also occur if factory's ‘stakeAmount’ state variable is changed to inappropriate number.

```
function slashReporter(address _reporter)
    external
    returns (uint256 _slashAmount)
{
.............................

uint256 factoryStakeAmount = IFactory(factory).stakeAmount();
else if (_lockedBalance + _stakedBalance >= factoryStakeAmount) {
    uint256 remainder = factoryStakeAmount - _lockedBalance;

    // Ensure there won't be an underflow
    require(remainder <= _stakedBalance, "invalid staked balance for slashing");

    _slashAmount = factoryStakeAmount;

    _updateStakeAndPayRewards(
        _reporter,
        _stakedBalance - remainder
    );
    toWithdraw -= _lockedBalance;
    _staker.lockedBalance = 0;
}
.............................
}
```

Fail scenario given that 3 contracts are deployed via the factory with default stakeAmount of 200:
In 2 of the contracts there is user which _lockedBalance is 100.
In 1 of the contracts there is user which _lockedBalance is 50.
Administrator of the factory contract changes the stakeAmount to 70.
Now tx will revert for the user which _lockedBalances is 100, because the equation will be 70 - 100 = -30, resulting in negative number, causing the uint256 to overflow (prevented by the compiler after solidity v0.8).
Tx will go through for the 3rd contract because the equation will be 70 - 50 = 20.
Require statement won’t be reached when overflow occurs.

- Recommendation:
  - When changing stakeAmount variable in factory contract, ensure that there are no users which _lockedBalances surpass the newly set value in the derived flexPrivate contracts to prevent overflow for these users.

FOUND THREATS

Informational

Owner can whitelist reporter addresses to submit specific data set. Eligible reporters can submit new data records to the contract. Once set, the data that whitelisted reporter can submit cannot be changed.

```
function updateWhitelist(address _reporter, bytes32 _queryId, bool _status) external {
    require(msg.sender == owner, "only owner can update whitelist");
    whitelist[_reporter] = _status;

    if (stakerDetails[_reporter].reporterQueryId == 0) {
    stakerDetails[_reporter].reporterQueryId = _queryId;
    }

    emit WhitelistUpdated(_reporter, _status);
}
```

FOUND THREATS

Informational

Governance address can remove reported data (nullify already created reports).

```
function removeValue(bytes32 _queryId, uint256 _timestamp) external {
    require(msg.sender == governance, "caller must be governance address");
    Report storage _report = reports[_queryId];
    require(!_report.isDisputed[_timestamp], "value already disputed");
    uint256 _index = _report.timestampIndex[_timestamp];
    require(_timestamp == _report.timestamps[_index], "invalid timestamp");
    _report.valueByTimestamp[_timestamp] = "";
    _report.isDisputed[_timestamp] = true;
    emit ValueRemoved(_queryId, _timestamp);
}
```

SignumPresale Contract

Token Name
Unavailable

Symbol
Unavailable

Contract Address
Unavailable

Network
Unavailable

Language
Solidity

Deployment Date
Not deployed yet

Contract Type
Factory

Total Supply
Unavailable

Decimals
Unavailable

TAXES

Buy Tax
0%

Sell Tax
0%

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ Low Risk

Owner can withdraw contract's native currency balances (ETH, BNB, etc.). Function may not be used if the owner is multi sig wallet which fallback/receive functionality requires more than 30,000 gas.

```
function withdraw() public onlyOwner {
    uint256 balance = address(this).balance;
    payable(msg.sender).call{value: balance, gas: 30000}("");
}
```

If multi sig wallet usage is desired ensure its receive/fallback won't consume more than 30,000 gas.

⚠ Low Risk

Owner can add purchase rounds.
Beware for token's decimals when _tokenCost is set, depending on each chain. This may cause undesired misspricing and/or revert in buy() functionality.

```
function addRound(uint256 _tokensAvailable, uint256 _tokenCost) public onlyOwner {
    rounds.push(Round(_tokensAvailable, _tokenCost));
}
```

Example: USDT stable coin is with 6 decimals in Ethereum mainnet but with 18 decimals in Binance smart chain mainnet.

FOUND THREATS

⚠ Low Risk

Owner can set payment token that will be used to purchase the presale tokens.
Beware for payment token's decimals. If token that have inappropriate decimals (different than 18), this may result in mispricing delivered from the ExchangeHelper contract.
Overflow may also occur if token's decimals are above 18 in ExchangeHelper's getRateFromDex() functionality.

```
function setPaymentToken(address _paymentToken, bool _allowed) public onlyOwner {
    paymentTokens[_paymentToken] = _allowed;
}

function getRateFromDex(address _tokenAddress) public view returns (uint256) {
    PriceRecord storage priceRecord = priceRecords[_tokenAddress];
    if (priceRecord.active) {
        uint256 rawTokenAmount = IERC20Extended(priceRecord.token).balanceOf(
            priceRecord.lpToken
        );
        uint256 tokenDecimalDelta = 18 -
            uint256(IERC20Extended(priceRecord.token).decimals());
        uint256 tokenAmount = rawTokenAmount.mul(10**tokenDecimalDelta);

        uint256 rawBaseTokenAmount = IERC20Extended(priceRecord.baseToken)
            .balanceOf(priceRecord.lpToken);
        uint256 baseTokenDecimalDelta = 18 -
            uint256(IERC20Extended(priceRecord.baseToken).decimals());
        uint256 baseTokenAmount = rawBaseTokenAmount.mul(
            10**baseTokenDecimalDelta
        );

        // Fetch the base token price using the getBaseTokenPrice function
        uint256 baseTokenPrice = getBaseTokenPrice();

        // Calculate the token price based on the fetched base token price
        uint256 tokenPrice = baseTokenPrice.mul(baseTokenAmount).div(tokenAmount);

        return tokenPrice;
    } else {
        return 0;
    }
}
```

FOUND THREATS

Informational

Owner can withdraw any ERC20 token from the contract.

```
function withdrawTokens(address _tokenAddress, uint256 _amount)
    public
    onlyOwner
{
    require(
        _amount <= IERC20(_tokenAddress).balanceOf(address(this)),
        "Insufficient token balance in contract."
    );

    IERC20(_tokenAddress).transfer(address(msg.sender), _amount);
}
```

Owner can set presale's start and end time.

```
function setStartAndStopTime(uint32 _startTime, uint32 _stopTime) public onlyOwner {
    require(_stopTime > _startTime, "Stop time must be after startTime.");
    startTime = _startTime;
    stopTime = _stopTime;
}
```

Owner can set minimum and maximum amount of tokens that can be purchased by user.

```
function setMinAndMaxAmount(uint256 _minAmount, uint256 _maxAmount) public onlyOwner {
    minAmount = _minAmount;
    maxAmount = _maxAmount;
}
```

FOUND THREATS

Informational: 2

Owner can set payment tokens receiver.

```
function setReceiver(address _newReceiver) public onlyOwner {
    receiver = _newReceiver;
}
```

General information:
When users participate in the presale with the buy() function, no actual tokens are distributed. Their buy share is saved into ‘totalBoughtByuser ’ mapping.

ExchangeHelper Contract

Token Name
Unavailable

Symbol
Unavailable

Contract Address
Unavailable

Network
Unavailable

Language
Solidity

Deployment Date
Not deployed yet

Contract Type
Factory

Total Supply
Unavailable

Decimals
Unavailable

TAXES

Buy Tax
0%

Sell Tax
0%

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ Low Risk

Owner can set new record for a token.

Beware for the newly set token's **decimals** as overflow will occur in 'getRateFromDex()' if the token have more than 18 decimals.

```
function setDexPriceRecord(
    address _token,
    address _baseToken,
    address _lpToken
) external onlyAdmin {
    PriceRecord storage priceRecord = priceRecords[_token];
    priceRecord.token = _token;
    priceRecord.baseToken = _baseToken;
    priceRecord.lpToken = _lpToken;
    priceRecord.active = true;
    emit PriceRecordUpdated(_token, _baseToken, _lpToken);
}

function getRateFromDex(address _tokenAddress) public view returns (uint256) {
    PriceRecord storage priceRecord = priceRecords[_tokenAddress];
    if (priceRecord.active) {
        uint256 rawTokenAmount = IERC20Extended(priceRecord.token).balanceOf(
            priceRecord.lpToken
        );
        uint256 tokenDecimalDelta = 18 -
            uint256(IERC20Extended(priceRecord.token).decimals());
        uint256 tokenAmount = rawTokenAmount.mul(10**tokenDecimalDelta);

        uint256 rawBaseTokenAmount = IERC20Extended(priceRecord.baseToken)
            .balanceOf(priceRecord.lpToken);
        uint256 baseTokenDecimalDelta = 18 -
            uint256(IERC20Extended(priceRecord.baseToken).decimals());
        uint256 baseTokenAmount = rawBaseTokenAmount.mul(
            10**baseTokenDecimalDelta
        );

        // Fetch the base token price using the getBaseTokenPrice function
        uint256 baseTokenPrice = getBaseTokenPrice();

        // Calculate the token price based on the fetched base token price
        uint256 tokenPrice = baseTokenPrice.mul(baseTokenAmount).div(tokenAmount);

        return tokenPrice;
    } else {
        return 0;
    }
}
```

FOUND THREATS

Informational

Admin can set exchange rate price for a token.

```
function setDirectPrice(address _token, uint256 _price) external onlyAdmin {
    emit DirectPriceUpdated(_token, assetPrices[_token], _price);
    assetPrices[_token] = _price;
}
```

Admin can transfer authorization to another address.

```
function setAdmin(address newAdmin) external onlyAdmin {
    address oldAdmin = admin;
    admin = newAdmin;

    emit NewAdmin(oldAdmin, newAdmin);
}
```

Signum TestToken

Token Name
Signum Test Token

Symbol
STT

Contract Address
Unavailable

Network
Unavailable

Language
Solidity

Deployment Date
Not deployed yet

Contract Type
Standard token

Total Supply
Assigned at deployment

Decimals
18

TAXES

Buy Tax
0%

Sell Tax
0%

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ Low Risk

Owner can self destruct the contract.
Depending on the EVM version, this may render the contract unusable/unreadable.

```
function selfDestruct() external onlyOwner {
    selfdestruct(payable(owner));
}
```

FOUND THREATS

Informational

'_EIP_SLOT' variable does not match the hash from keccak256(abi.encodePacked(_EIP_SLOT)) value.

```
bytes32 constant _EIP_SLOT =
    0x7050c9e0f4ca769c69bd3a8ef740bc37934f8e2c036e5a723fd8ee048ed3f8c3;
```

WEBSITE

Website URL:
Unavailable

Domain Registry
https://www.godaddy.com

Domain Expiration
Unavailable

Technical SEO Test
Passed

Security Test
Passed. SSL certificate present

Design
Unavailable

Content
Unavailable

Whitepaper
Unavailable

Roadmap

Unavailable

Mobile-friendly?

Unavailable

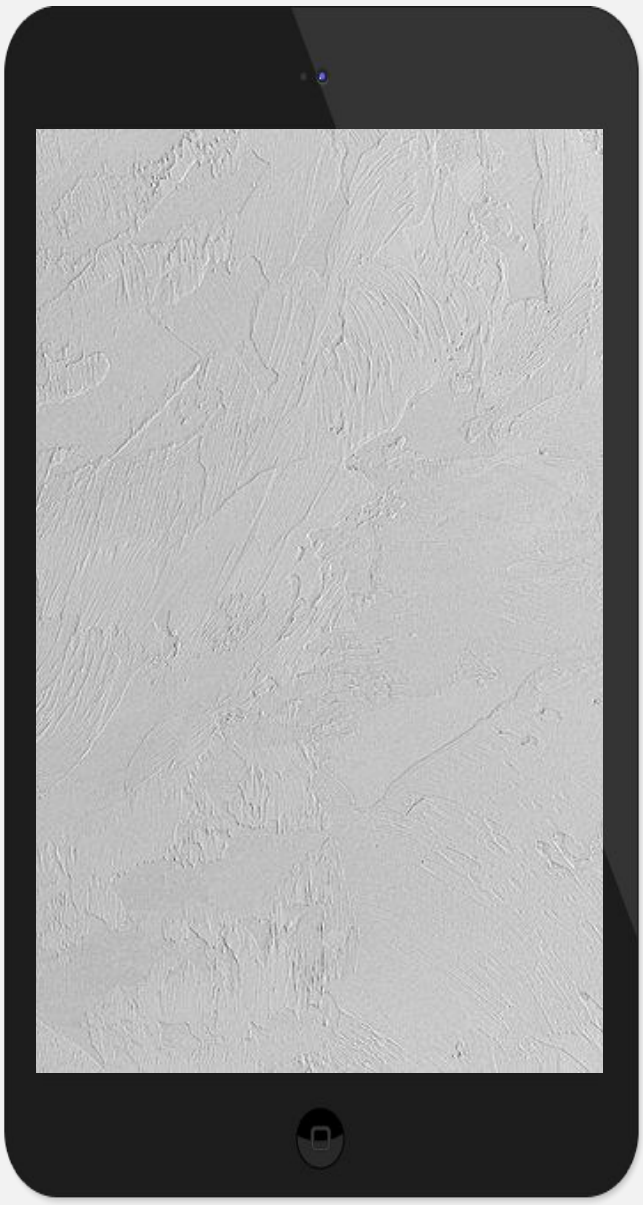
Under Construction

SOCIAL MEDIA

Social Score: 0%

ANALYSIS

Unavailable



Twitter:

Unavailable

Discord

Unavailable

Telegram:

Unavailable

Medium

Unavailable

SPYWOLF

CRYPTO SECURITY

Audits | KYCs | dApps
Contract Development

ABOUT US

We are a growing crypto security agency offering audits, KYCs and consulting services for some of the top names in the crypto industry.

- **OVER 700 SUCCESSFUL CLIENTS**
- **MORE THAN 1000 SCAMS EXPOSED**
- **MILLIONS SAVED IN POTENTIAL FRAUD**
- **PARTNERSHIPS WITH TOP LAUNCHPADS, INFLUENCERS AND CRYPTO PROJECTS**
- **CONSTANTLY BUILDING TOOLS TO HELP INVESTORS DO BETTER RESEARCH**

To hire us, reach out to
contact@spywolf.co or
t.me/joe_SpyWolf

FIND US ONLINE

SPYWOLF.CO

@SPYWOLFNETWORK

@SPYWOLFNETWORK

Disclaimer

This report shows findings based on our limited project analysis, following good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall social media and website presence and team transparency details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report.
While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER:
By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.
No one shall have any right to rely on the report or its contents, and SpyWolf and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (SpyWolf) owe no duty of care towards you or any other person, nor does SpyWolf make any warranty or representation to any person on the accuracy or completeness of the report.
The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SpyWolf hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SpyWolf hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SpyWolf, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts, website, social media and team.
No applications were reviewed for security. No product code has been reviewed.