Full Audit Report

JETI-NFT-Trudopes Security Assessment

Real Cybersecurity
Protecting digital assets

Made in Thailand

Audit

SECURI LAB
(THAILAND) contact@securi-lab.com

FULL AUDIT REPORT

Table of Contents 1

FULL AUDIT REPORT

Report Information

About Report	**JETI-NFT-Trudopes Security Assessment**
Version	**v1.0**
Client	**Jeti Service**
Language	**Solidity**
Confidentiality	**Public**
Contract File	**Trudopes.sol** SHA-1: a29da6a5b2b3bf920b8ae4cb87486d130865221a **Marketplace.sol** SHA-1: f0cc672709680e5049721d8fb98f3eef1f0fb923 **Payments.sol** SHA-1: cbe51e33ff442221f94ef032dc7e7c44f3750a7e **PaymentFactory.sol** SHA-1: a044475dbdd032ad6df4f93e572c1d60544b75f8 **Vendor.sol** SHA-1: 333e332ca46c1fcd3ff32a54e4c681fdacac019e **This audit uses the file as the client submitted. Please check with a differential checker after the smart contract code has been deployed and verified.**
Audit Method	**Whitebox**
Security Assessment Author	**Auditor** **Mark K. [Security Researcher | Redteam]** **Kevin N. [Security Researcher | Web3 Dev]** **Yusheng T. [Security Researcher | Incident Response]** **Approve Document** **Ronny C. CTO & Head of Security Researcher** **Chinnakit J. CEO & Founder**

*Audit Method

Whitebox: SECURI LAB Team receives all source code from the client to provide the assessment.
Blackbox: SECURI LAB Team receives only bytecode from the client to provide the assessment.

Digital Sign (Only Full Audit Report)

Digitally signed by Chinnakit J.
DN: cn=Chinnakit J., o=SECURI LAB (Thailand), ou=CEO & Founder, email=contact@securi-lab.com, c=TH
Date: 2023.06.14 20:02:33 +07'00'

Disclaimer

Regarding this security assessment, there are no guarantees about the security of the program instruction received from the client is hereinafter referred to as "**Source code**".
And **SECURI Lab** hereinafter referred to as "**Service Provider**", the **Service Provider** will not be held liable for any legal liability arising from errors in the security assessment. The responsibility will be the responsibility of the **Client**, hereinafter referred to as "**Service User**" and the **Service User** agrees not to be held liable to the **service provider** in any case. By contract **Service Provider** to conduct security assessments with integrity with professional ethics, and transparency to deliver security assessments to users The **Service Provider** has the right to postpone the delivery of the security assessment. If the security assessment is delayed whether caused by any reason and is not responsible for any delayed security assessments.
If **the service provider** finds a vulnerability The **service provider** will notify the **service user** via the Preliminary Report, which will be kept confidential for security. The **service provider** disclaims responsibility in the event of any attacks occurring whether before conducting a security assessment. Or happened later All responsibility shall be sole with the **service user**.

Security Assessment Not Financial/Investment Advice Any loss arising from any investment in any project is the responsibility of the investor.

SECURI LAB disclaims any liability incurred. Whether it's Rugpull, Abandonment, Soft Rugpull

The SECURI LAB team has conducted a comprehensive security assessment of the vulnerabilities.
This assessment is tested with an expert assessment. Using the following test requirements

1. Smart Contract Testing with Expert Analysis By testing the most common and uncommon vulnerabilities.
2. Automated program testing It includes a sample vulnerability test and a sample of the potential vulnerabilities being used for the most frequent attacks.
3. Manual Testing with AST/WAS/ASE/SMT and reviewed code line by line
4. Visibility, Mutability, Modifier function testing, such as whether a function can be seen in general, or whether a function can be changed and if so, who can change it.
5. Function association test It will be displayed through the association graph.
6. This safety assessment is cross-checked prior to the delivery of the assessment results.

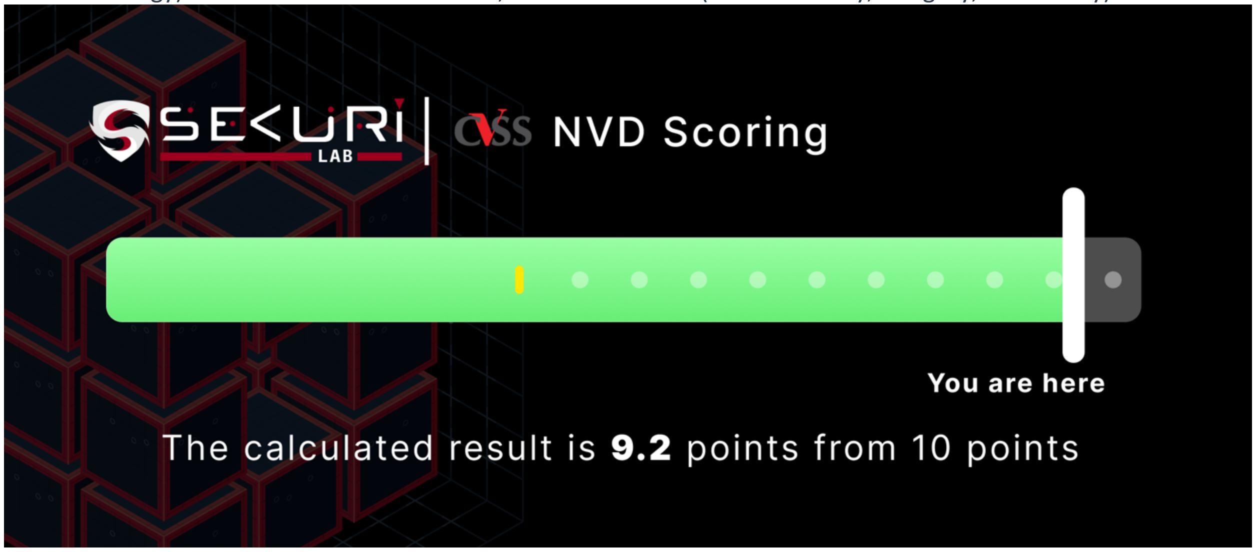
FULL AUDIT REPORT

Executive Summary

For this security assessment, SECURI LAB received a request from Jeti Services on Thursday, May 18, 2023.

NVD CVSS Scoring

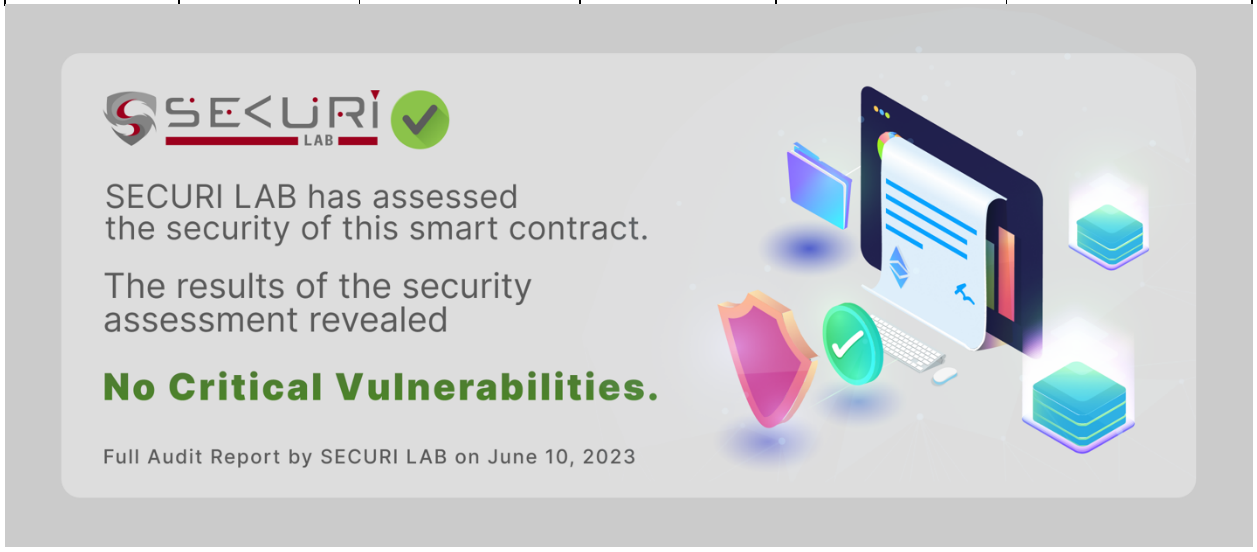
The score was calculated using the NVD (National Vulnerability Database) of NIST (National Institute of Standards and Technology) under the CVSS 3.1 standard, based on the CIA (Confidentiality, Integrity, Availability).



Audit Result

SECURI LAB evaluated the smart contract security of the project and found: **[Total : 3]**

Critical	High	Medium	Low	Very Low	Informational
0	1	0	0	0	2


SECURI LAB
SECURI LAB has assessed the security of this smart contract.
The results of the security assessment revealed
No Critical Vulnerabilities.
Full Audit Report by SECURI LAB on June 10, 2023

FULL AUDIT REPORT

Project Introduction

Scope Information:

Project Name	Jeti Services
Website	https://jeti.one/
Chain	-
Language	Solidity

Audit Information:

Request Date	Thursday, May 18, 2023
Audit Date	Sunday, June 7, 2023
Re-assessment Date	-

Audit Version History:

Version	Date	Description
1.0	Saturday, June 10, 2023	Preliminary Report
1.1	Wednesday, June 14, 2023	Full Audit Report

FULL AUDIT REPORT

Initial Audit Scope:

Smart Contract File	**Trudopes.sol** SHA-1: a29da6a5b2b3bf920b8ae4cb87486d130865221a **Marketplace.sol** SHA-1: f0cc672709680e5049721d8fb98f3eef1f0fb923 **Payments.sol** SHA-1: cbe51e33ff442221f94ef032dc7e7c44f3750a7e **PaymentFactory.sol** SHA-1: a044475dbdd032ad6df4f93e572c1d60544b75f8 **Vendor.sol** SHA-1: 333e332ca46c1fcd3ff32a54e4c681fdacac019e **This audit uses the file as the client submitted. Please check with a differential checker after the smart contract code has been deployed and verified.**
Compiler Version	**v0.8.17**

Source Units Analyzed: **5**
Source Units in Scope: **5** (**100%**)

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝🔍	contracts/Trudopes.sol	1	1	224	212	150	27	156	💰📤🧮
📝🔍	contracts/Marketplace.sol	1	3	312	302	229	1	187	💰🌀
📝🔍	contracts/Payments.sol	1	1	234	230	163	1	127	💰
📝🔍	contracts/PaymentFactory.sol	1	1	102	95	70	1	71	💰📤🌀
📝🔍	contracts/Vendor.sol	1	2	383	369	290	5	219	💰📤

FULL AUDIT REPORT

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝🔍	Totals	5	8	1255	1208	902	35	760	💰📤🧮🌀

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, ...)

Dependencies / External Imports

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	5
@openzeppelin/contracts/security/ReentrancyGuard.sol	5
@openzeppelin/contracts/token/ERC721/ERC721.sol	2
@openzeppelin/contracts/utils/Address.sol	2
@openzeppelin/contracts/utils/Counters.sol	2
@openzeppelin/contracts/utils/Strings.sol	1
@openzeppelin/contracts/utils/cryptography/MerkleProof.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	2

Description Report Files Description Table

File Name	SHA-1 Hash
contracts/Trudopes.sol	a29da6a5b2b3bf920b8ae4cb87486d130865221a
contracts/Marketplace.sol	f0cc672709680e5049721d8fb98f3eef1f0fb923
contracts/Payments.sol	cbe51e33ff442221f94ef032dc7e7c44f3750a7e
contracts/PaymentFactory.sol	a044475dbdd032ad6df4f93e572c1d60544b75f8
contracts/Vendor.sol	333e332ca46c1fcd3ff32a54e4c681fdacac019e

Security Assessment Procedure

Securi has the following procedures and regulations for conducting security assessments:

1.Request Audit Client submits a form request through the Securi channel. After receiving the request, Securi will discuss a security assessment. And drafting a contract and agreeing to sign a contract together with the Client

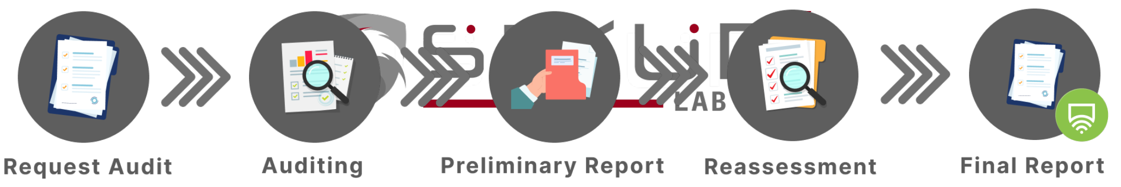
2.Auditing Securi performs security assessments of smart contracts obtained through automated analysis and expert manual audits.

3.Preliminary Report At this stage, Securi will deliver an initial security assessment. To report on vulnerabilities and errors found under Audit Scope <u>will not publish preliminary reports for safety</u>.

4.Reassessment After Securi has delivered the Preliminary Report to the Client, Securi will track the status of the vulnerability or error, which will be published to the Final Report at a later date with the following statuses:

- **a.Acknowledge** The client has been informed about errors or vulnerabilities from the security assessment.
- **b.Resolved** The client has resolved the error or vulnerability. Resolved is probably just a commit, and Securi is unable to verify that the resolved has been implemented or not.
- **c.Decline** Client has rejected the results of the security assessment on the issue.

5.Final Report Securi providing full security assessment report and public



Risk Rating

Risk rating using this commonly defined: $Risk\ rating\ =\ impact\ *\ confidence$

Impact	The severity and potential impact of an attacker attack
Confidence	Ensuring that attackers expose and use this vulnerability

Both have a total of 3 levels: **High**, **Medium**, **Low**. By *Informational* will not be classified as a level

Confidence Impact [Likelihood]	Low	Medium	High
Low	**Very Low**	**Low**	**Medium**
Medium	**Low**	**Medium**	**High**
High	**Medium**	**High**	**Critical**

FULL AUDIT REPORT

Vulnerability Severity Summary

Severity is a risk assessment It is calculated from the Impact and Confidence values using the following calculation methods,

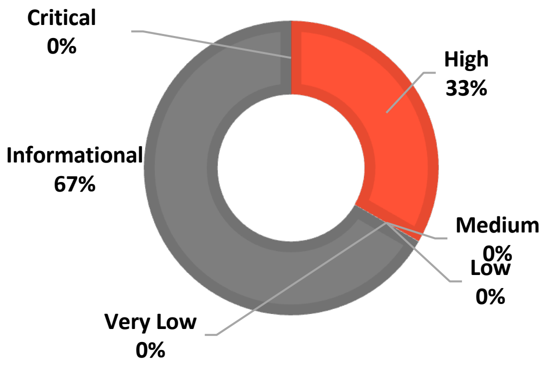
$Risk\ rating\ =\ impact\ *\ confidence$

It is categorized into

5 categories based on the **lowest severity**:

Very Low, Low, Medium, **High**, Critical.

For **Informational** & will **Non-class/Optimization/Best-practices will** not be counted as **severity**



Vulnerability Severity Level	Total
Critical	**0**
High	**1**
Medium	**0**
Low	**0**
Very Low	**0**
Informational	**2**
Non-class/Optimization/Best-practices	**0**

Category information:

Centralization	Economics Risk	Logical Issue	Authorization	Mathematical	Naming Conventions
Centralization Risk is The risk incurred by a sole proprietor, such as the Owner being able to change something without permission	**Economics Risk** is Risks that may affect the economic mechanism system, such as the ability to increase Mint token	**Logical Issue** is that can cause errors to core processing, such as any prior operations that cause background processes to crash.	**Authorization** is Possible pitfalls from weak coding allows unrelated people to take any action to modify the values.	**Mathematical** Any erroneous arithmetic operations affect the operation of the system or lead to erroneous values.	**Naming Conventions** naming variables that may affect code understanding or naming inconsistencies

Security Risk	Coding Style	Best Practices	Optimization	Gas Optimization	Dead Code
Security Risk of loss or damage if it's no mitigate	**Coding Style** is Tips coding for efficiency performance	**Best Practices** is suggestions for improvement	**Optimization** is performance improvement	**Gas Optimization** is increase performance to avoid expensive gas	**Dead Code** having unused code This may result in wasted resources and gas fees.

FULL AUDIT REPORT

Vulnerability Findings

ID	Vulnerability Detail	Severity	Category	Status
SEC-01	Centralization Risk	High	Centralization	**Mitigate**
SEC-02	Avoid using block timestamp	Informational	Best Practices	**Acknowledge**
SEC-03	\`abi.encodePacked()\` should not be used with dynamic types when passing the result to a hash function such as \`keccak256()\`	Informational	Best Practices	**Acknowledge**

FULL AUDIT REPORT

SEC-01: Centralization Risk

Vulnerability Detail	Severity	Location	Category	Status
Centralization Risk	High	Check on finding	Centralization	**Mitigate**

Finding:

```
File: Marketplace.sol

23: contract MarketPlace is ReentrancyGuard, Ownable {

304:     function withdraw() public payable onlyOwner {

```

```solidity
File: PaymentFactory.sol

14: contract PaymentFactory is Ownable, ReentrancyGuard {

42:     function setFeeTo(address feeReceivingAddress) external onlyOwner {

46:     function setFlatFee(uint256 fee) external onlyOwner {

57:     function newBlacklistContract(address _newBlacklist) public onlyOwner {

61:     function newAdmin(address _newAdmin) public onlyOwner {

```

```solidity
File: Payments.sol

14: contract Payments is Ownable, ReentrancyGuard {

113:     function changeWallets(address[] memory wallets_) public onlyOwner {

119:     function changePercentages(uint256[] memory percentages_) public onlyOwner {

133:     function changeWalletsAndPercentages(address[] memory wallets_, uint256[]
memory percentages_) public onlyOwner {

148:     function addOwner(address owner) onlyOwner public {
```

```
154:     function removeOwner(address owner) onlyOwner public {

170:     function withdrawFunds(uint256 amount) public onlyOwner {

```

```solidity
File: Trudopes.sol

20: contract Trudopes is Ownable, ERC721A, ReentrancyGuard, DefaultOperatorFilterer {

106:     function giftsFromLobbyists(address _to, uint _quantity) public onlyOwner {

112:     function emptyShelves(uint256 _newMaxSupply) public onlyOwner {

123:     function setMaxDopeInflationReductionTotal(uint256
_newMaxDopeInflationReductionTotal) public onlyOwner {

128:     function setMaxWalletDopeInflation(uint256 _newMaxWalletDopeInflation) public
onlyOwner {

133:     function setMaxWalletDopeInflationReduction(uint256
_newMaxWalletDopeInflationReduction) public onlyOwner {

138:     function dopeInflationRebate(uint256 _newDopeInflationReductionPrice) public
onlyOwner {

143:     function dopeInflationFee(uint256 _newDopeInflationPrice) public onlyOwner {

148:     function setBaseURI(string memory _newBaseURI) public onlyOwner {

153:     function setStep(uint8 _step) public onlyOwner {

166:     function setMerkleRootWL(bytes32 _newMerkleRootDopeInflationReduction)
external onlyOwner {

218:     function setRoyaltyInfo (address _receiver, uint96 royaltyFeesInBips_) public
onlyOwner {

```

```solidity
File: Vendor.sol

25: contract Vendor is ReentrancyGuard, Ownable {
```

```
375:     function withdraw() public payable onlyOwner {

```
```

Scenario:
Centralized risk refers to the potential security risks that arise when a smart contract is controlled by a central entity or a single point of failure. If the contract is controlled by a central authority, then the contract may be vulnerable to attacks that target the centralized entity.
Centralized risk that can lead to rug pulls typically arises from the centralization of control or ownership of a project's assets, particularly in decentralized finance (DeFi) projects built on blockchain platforms like Ethereum.

Recommendation:
In terms of timeframes, there are three categories: short-term, long-term, and permanent.

For short-term solutions, a combination of timelock and multi-signature (2/3 or 3/5) can be used to mitigate risk by delaying sensitive operations and avoiding a single point of failure in key management. This includes implementing a timelock with a reasonable latency, such as 48 hours, for privileged operations; assigning privileged roles to multi-signature wallets to prevent private key compromise; and sharing the timelock contract and multi-signer addresses with the public via a medium/blog link.

For long-term solutions, a combination of timelock and DAO can be used to apply decentralization and transparency to the system. This includes implementing a timelock with a reasonable latency, such as 48 hours, for privileged operations; introducing a DAO/governance/voting module to increase transparency and user involvement; and sharing the timelock contract, multi-signer addresses, and DAO information with the public via a medium/blog link.

Finally, permanent solutions should be implemented to ensure the ongoing security and protection of the system.

Alleviation:
Regarding this, we discussed and found a solution to the matter with Jeti One Team, because the contract needed a function to be suspended/change percentage/setting fee/blacklist—important settings to comply with the mechanism of the system. We deserve to see that such issues are addressed on Mitigate part and users are encouraged to follow and update platform announcements at all times.
```

FULL AUDIT REPORT

SEC-02: Avoid using block timestamp

Vulnerability Detail	Severity	Location	Category	Status
Avoid using block timestamp	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:

```
File: Vendor.sol

239:                 block.timestamp,

259:         listedItems[arrayId].cancelledDate = block.timestamp;

317:         listedItems[arrayId].soldDate = block.timestamp;
```

Recommendation:

Avoid relying on `block.timestamp`.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Exploit Scenario:

Dangerous usage of block.timestamp. block.timestamp can be manipulated by miners.

"Bob's contract relies on block.timestamp for its randomness. Eve is a miner and manipulates block.timestamp to exploit Bob's contract.

Alleviation:

Jeti Team has acknowledge this issue.

SEC-03: `abi.encodePacked()` should not be used with dynamic types when passing the result to a hash function such as `keccak256()`

Vulnerability Detail	Severity	Location	Category	Status
`abi.encodePacked()` should not be used with dynamic types when passing the result to a hash function such as `keccak256()`	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:

```
File: Trudopes.sol

175:          return keccak256(abi.encodePacked(_account));
```

Recommendation:

Use `abi.encode()` instead which will pad items to 32 bytes, which will [prevent hash collisions](https://docs.soliditylang.org/en/v0.8.13/abi-spec.html#non-standard-packed-mode) (e.g. `abi.encodePacked(0x123,0x456)` => `0x123456` => `abi.encodePacked(0x1,0x23456)`, but `abi.encode(0x123,0x456)` => `0x0...1230...456`). "Unless there is a compelling reason, `abi.encode` should be preferred". If there is only one argument to `abi.encodePacked()` it can often be cast to `bytes()` or `bytes32()` [instead](https://ethereum.stackexchange.com/questions/30912/how-to-compare-strings-in-solidity#answer-82739).
If all arguments are strings and or bytes, `bytes.concat()` should be used instead

Exploit Scenario:

-

Alleviation:

Jeti Team has acknowledge this issue.

SWC Findings

ID	Title	Scanning	Result
SWC-100	Function Default Visibility	Complete	No risk
SWC-101	Integer Overflow and Underflow	Complete	No risk
SWC-102	Outdated Compiler Version	Complete	No risk
SWC-103	Floating Pragma	Complete	No risk
SWC-104	Unchecked Call Return Value	Complete	No risk
SWC-105	Unprotected Ether Withdrawal	Complete	No risk
SWC-106	Unprotected SELFDESTRUCT Instruction	Complete	No risk
SWC-107	Reentrancy	Complete	No risk
SWC-108	State Variable Default Visibility	Complete	No risk
SWC-109	Uninitialized Storage Pointer	Complete	No risk
SWC-110	Assert Violation	Complete	No risk
SWC-111	Use of Deprecated Solidity Functions	Complete	No risk
SWC-112	Delegatecall to Untrusted Callee	Complete	No risk
SWC-113	DoS with Failed Call	Complete	No risk
SWC-114	Transaction Order Dependence	Complete	No risk
SWC-115	Authorization through tx.origin	Complete	No risk

SWC-116	Block values as a proxy for time	Complete	No risk
SWC-117	Signature Malleability	Complete	No risk
SWC-118	Incorrect Constructor Name	Complete	No risk
SWC-119	Shadowing State Variables	Complete	No risk
SWC-120	Weak Sources of Randomness from Chain Attributes	Complete	No risk
SWC-121	Missing Protection against Signature Replay Attacks	Complete	No risk
SWC-122	Lack of Proper Signature Verification	Complete	No risk
SWC-123	Requirement Violation	Complete	No risk
SWC-124	Write to Arbitrary Storage Location	Complete	No risk
SWC-125	Incorrect Inheritance Order	Complete	No risk
SWC-126	Insufficient Gas Griefing	Complete	No risk
SWC-127	Arbitrary Jump with Function Type Variable	Complete	No risk
SWC-128	DoS With Block Gas Limit	Complete	No risk
SWC-129	Typographical Error	Complete	No risk
SWC-130	Right-To-Left-Override control character (U+202E)	Complete	No risk
SWC-131	Presence of unused variables	Complete	No risk
SWC-132	Unexpected Ether balance	Complete	No risk

FULL AUDIT REPORT

SWC-133	Hash Collisions With Multiple Variable Length Arguments	Complete	No risk
SWC-134	Message call with hardcoded gas amount	Complete	No risk
SWC-135	Code With No Effects	Complete	No risk
SWC-136	Unencrypted Private Data On-Chain	Complete	No risk

Visibility, Mutability, Modifier function testing

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
5	0	8	0

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
90	9

External	Internal	Private	Pure	View
25	63	2	1	34

StateVariables

Total	🌐Public
66	33

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`=0.8.17` `0.8.17`		`yes`		

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
`yes`			`yes`		`yes` → `NewContract:Ve`

FULL AUDIT REPORT

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
					`ndor` → `NewContract:Payments`

♻️ TryCatch	Σ Unchecked

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
iBlacklist	Interface			
└	getBlacklist	External ❗		NO❗
Trudopes	Implementation	Ownable, ERC721A, ReentrancyGuard, DefaultOperatorFilterer		
└		Public ❗	🛑	ERC721A
└	dopeInflationReductionMint	Public ❗	💵	nonReentrant callerIsUser
└	dopeInflationMint	Public ❗	💵	nonReentrant callerIsUser
└	giftsFromLobbyists	Public ❗	🛑	onlyOwner
└	emptyShelves	Public ❗	🛑	onlyOwner
└	setMaxDopeInflationTotal	Public ❗	🛑	onlyOwner
└	setMaxDopeInflationReductionTotal	Public ❗	🛑	onlyOwner
└	setMaxWalletDopeInflation	Public ❗	🛑	onlyOwner
└	setMaxWalletDopeInflationReduction	Public ❗	🛑	onlyOwner
└	dopeInflationRebate	Public ❗	🛑	onlyOwner
└	dopeInflationFee	Public ❗	🛑	onlyOwner
└	setBaseURI	Public ❗	🛑	onlyOwner

FULL AUDIT REPORT

Contract	Type	Bases		
└	setStep	Public ❗️	🛑	onlyOwner
└	tokenURI	Public ❗️		NO❗️
└	setMerkleRootWL	External ❗️	🛑	onlyOwner
└	isDopeInflationReduction	Internal 🔒		
└	leaf	Internal 🔒		
└	_verifyDopeInflationReduction	Internal 🔒		
└	setApprovalForAll	Public ❗️	🛑	onlyAllowedOperatorApproval
└	approve	Public ❗️	🛑	onlyAllowedOperatorApproval
└	transferFrom	Public ❗️	🛑	onlyAllowedOperator
└	safeTransferFrom	Public ❗️	🛑	onlyAllowedOperator
└	safeTransferFrom	Public ❗️	🛑	onlyAllowedOperator
└	royaltyInfo	External ❗️		NO❗️
└	calculateRoyalty	Public ❗️		NO❗️
└	setRoyaltyInfo	Public ❗️	🛑	onlyOwner
IBlacklist	Interface			
└	getBlacklist	External ❗️		NO❗️
INFTContract	Interface			
└	owner	External ❗️		NO❗️
└	royaltyInfo	External ❗️		NO❗️

FULL AUDIT REPORT

Contract	Type	Bases		
IVendor	Interface			
└	setApproval	External ❗	🛑	NO❗
MarketPlace	Implementation	ReentrancyGuard, Ownable		
└		External ❗	💵	NO❗
└		Public ❗	🛑	NO❗
└	setAuthRequired	Public ❗	🛑	isAdmin
└	setSalePercentage	Public ❗	🛑	isAdmin
└	createVendor	Public ❗	🛑	NO❗
└	approvePendingVendor	Public ❗	🛑	isAdmin
└	addApproved	Public ❗	🛑	isAdjustor isSetContract
└	addHeld	Public ❗	🛑	isAdjustor isSetContract
└	addDenied	Public ❗	🛑	isAdjustor isSetContract
└	removeAddress	Public ❗	🛑	isAdjustor isSetContract
└	getVendorAddress	Public ❗		NO❗
└	getPendingVendors	Public ❗		NO❗
└	getApprovedVendors	Public ❗		NO❗
└	getHeldVendors	Public ❗		NO❗
└	getDeniedVendors	Public ❗		NO❗
└	_getBlacklist	Public ❗		NO❗
└	_getAdmin	Public ❗		NO❗

FULL AUDIT REPORT

Contract	Type	Bases		
└	_getSalePercentage	Public ❗️		NO❗️
└	_getPayment	Public ❗️		NO❗️
└	withdraw	Public ❗️	💵	onlyOwner
iPaymentFactory	Interface			
└	_getAdmin	External ❗️		NO❗️
Payments	Implementation	Ownable, ReentrancyGuard		
└		External ❗️	💵	NO❗️
└		Public ❗️	🛑	NO❗️
└	changeWallets	Public ❗️	🛑	onlyOwner
└	changePercentages	Public ❗️	🛑	onlyOwner
└	changeWalletsAndPercentages	Public ❗️	🛑	onlyOwner
└	addOwner	Public ❗️	🛑	onlyOwner
└	removeOwner	Public ❗️	🛑	onlyOwner
└	toggleAdminSigner	Public ❗️	🛑	isAdmin
└	availableFunds	Public ❗️		validOwner
└	withdrawFunds	Public ❗️	🛑	onlyOwner
└	createTransaction	Private 🔐	🛑	notOpen
└	signTransaction	Public ❗️	🛑	validOwner txExists notExecuted notConfirmed
└	_withdrawFunds	Private 🔐	🛑	

FULL AUDIT REPORT

Contract	**Type**	**Bases**		
iBlacklist	Interface			
└	getBlacklist	External ❗		NO❗
PaymentFactory	Implementation	Ownable, ReentrancyGuard		
└		Public ❗	🛑	NO❗
└	setFeeTo	External ❗	🛑	onlyOwner
└	setFlatFee	External ❗	🛑	onlyOwner
└	refundExcessiveFee	Internal 🔒	🛑	
└	newBlacklistContract	Public ❗	🛑	onlyOwner
└	newAdmin	Public ❗	🛑	onlyOwner
└	create	External ❗	💵	enoughFee nonReentrant
└	_getBlacklist	Public ❗		NO❗
└	_getAdmin	Public ❗		NO❗
INFTContract	Interface			
└	owner	External ❗		NO❗
└	royaltyInfo	External ❗		NO❗
IMarketPlace	Interface			
└	addApproved	External ❗	🛑	NO❗
└	addHeld	External ❗	🛑	NO❗
└	addDenied	External ❗	🛑	NO❗

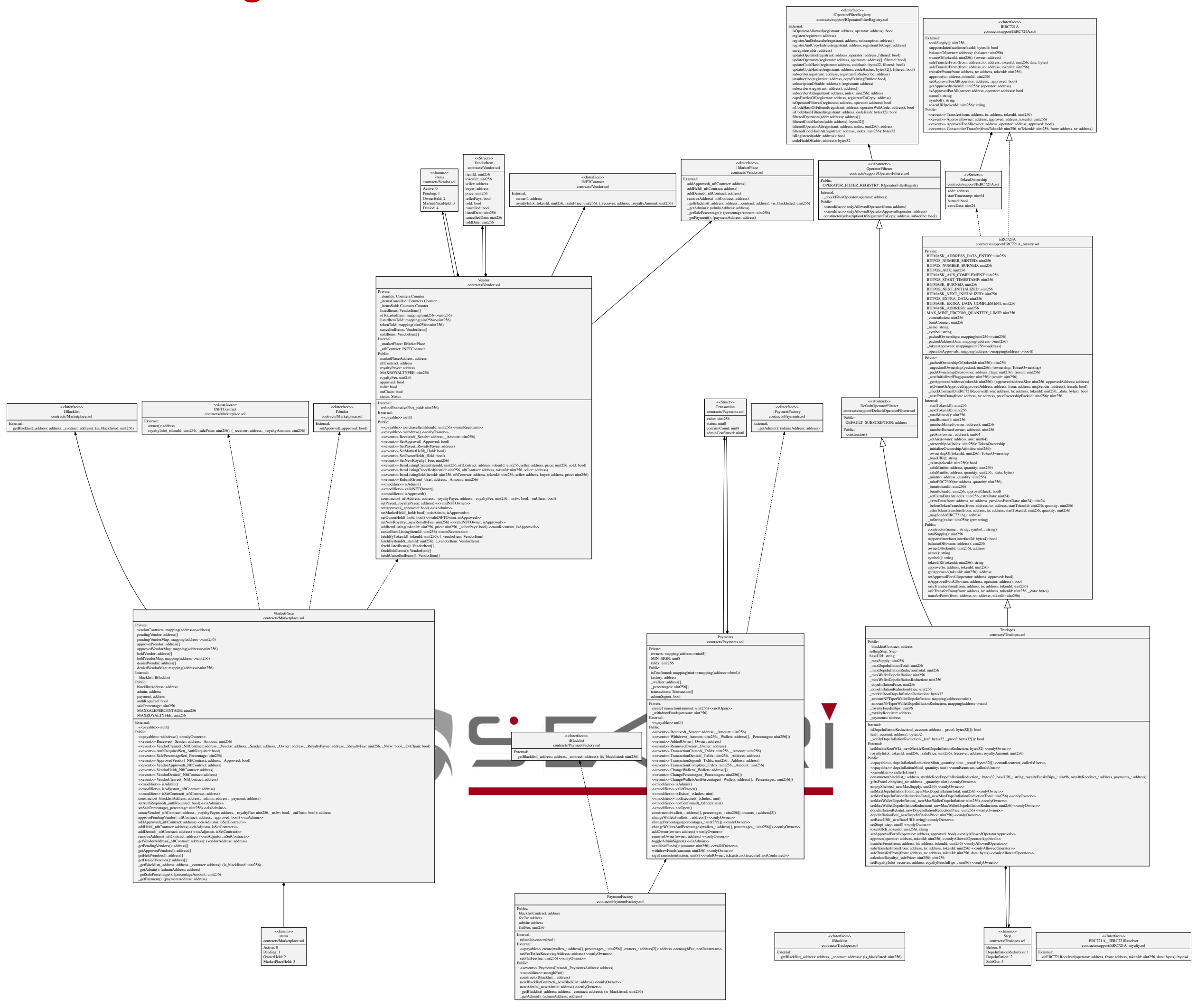
Contract	Type	Bases		
└	removeAddress	External ❗	🛑	NO❗
└	_getBlacklist	External ❗		NO❗
└	_getAdmin	External ❗		NO❗
└	_getSalePercentage	External ❗		NO❗
└	_getPayment	External ❗		NO❗
Vendor	Implementation	ReentrancyGuard, Ownable		
└		External ❗	💵	NO❗
└		Public ❗	🛑	NO❗
└	setPayee	Public ❗	🛑	validNFTOwner
└	setApproval	Public ❗	🛑	isAdmin
└	setMarketHeld	Public ❗	🛑	isAdmin isApproved
└	setOwnerHeld	Public ❗	🛑	validNFTOwner isApproved
└	setNewRoyalty	Public ❗	🛑	validNFTOwner isApproved
└	addItemListing	Public ❗	🛑	nonReentrant isApproved
└	cancelItemListing	Public ❗	🛑	nonReentrant
└	purchaseItem	Public ❗	💵	nonReentrant
└	refundExcessiveFee	Internal 🔒	🛑	
└	fetchByTokenId	Public ❗		NO❗
└	fetchByItemId	Public ❗		NO❗
└	fetchListedItems	Public ❗		NO❗

Contract	Type	Bases		
└	fetchSoldItems	Public ❗️		NO❗️
└	fetchCancelledItems	Public ❗️		NO❗️
└	withdraw	Public ❗️	💵	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

UML Class Diagram



About SECURI LAB

SECURI LAB is a group of cyber security experts providing cyber security consulting, smart contract security audits, and KYC services.

Follow Us On:

Website	https://securi-lab.com/
Twitter	https://twitter.com/SECURI_LAB
Telegram	https://t.me/securi_lab
Medium	https://medium.com/@securi