



Full Audit Report

Patwars Security Assessment

Real Cybersecurity
Protecting digital assets



Made in Thailand

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




FULL AUDIT REPORT

Table of Contents	1
▪ Report Information	2
▪ Disclaimer	3
▪ Executive Summary	4
NVD CVSS Scoring	
Audit Result	
▪ Project Introduction	5
Scope Information	
Audit Information	
Audit Version History	
▪ Initial Audit Scope	6-7
▪ Security Assessment Procedure	8
▪ Risk Rating	9
▪ Vulnerability Severity Summary	10
▪ Vulnerability Findings	11-12
SWC & SEC & Non-severity level	
▪ SWC Findings	13-15
▪ Visibility, Mutability, Modifier function testing	16-17
Component, Exposed Function	
StateVariables, Capabilities, Contract Descripton Table	
▪ Inheritate Function Relation Graph	18
▪ UML Diagram	19
▪ About Securi	20

FULL AUDIT REPORT

Report Information

About Report	Patwars Security Assessment
Version	v1.0
Client	Patwars
Language	Solidity
Confidentiality	Public
Contract File	PAWS.sol SHA-1: 52270e6c6dede473b38b6012eb74f8d91bad8343
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)

FULL AUDIT REPORT

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 LAB~~ 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.

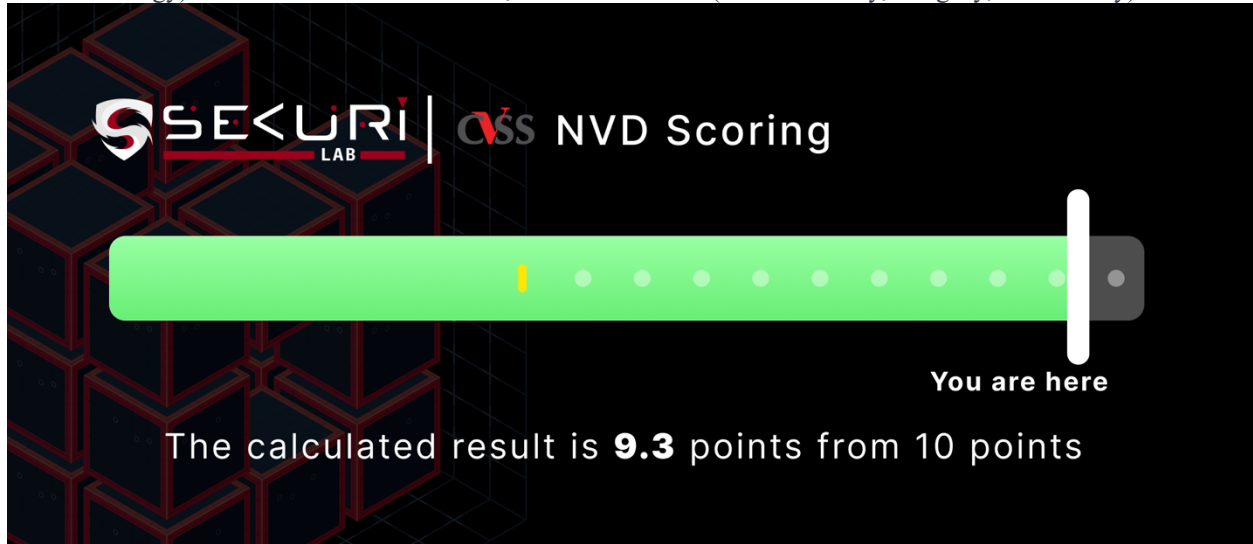
FULL AUDIT REPORT

Executive Summary

For this security assessment, SECURI LAB received a request from Patwars on Thursday, April 20, 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 : 2**

Critical	High	Medium	Low	Very Low	Informational
0	0	0	0	0	0

SECURI LAB has assessed the security of this smart contract.

The results of the security assessment revealed

No Vulnerabilities.

Full Audit Report by SECURI LAB on Apr 20, 2023

FULL AUDIT REPORT

Project Introduction

Scope Information:

Project Name	Patwars
Website	https://www.apebrigade.io
Chain	-
Language	Solidity

Audit Information:

Request Date	Thursday, April 20, 2023
Audit Date	Thursday, April 20, 2023
Re-assessment Date	-

Audit Version History:









Version	Date	Description
1.0	Thursday, April 20, 2023	Preliminary Report

LAB

FULL AUDIT REPORT

Initial Audit Scope:

Smart Contract File	PAWS.sol SHA-1: 52270e6c6dede473b38b6012eb74f8d91bad8343
Compiler Version	v0.8.18

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
  	contracts/ PAWS.sol	2	2	179	146	95	25	79	 Σ
  	Totals	2	2	179	146	95	25	79	 Σ



- **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, ...)

FULL AUDIT REPORT

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 will not publish preliminary reports for safety.

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



FULL AUDIT REPORT

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,

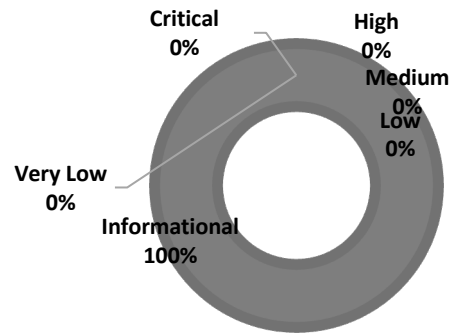
$Risk\ rating = impact * confidence$

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	0
Medium	0
Low	0
Very Low	0
Informational	0
Non-class/Optimization/Best-practices	2

Category information:

Centralization Centralization Risk is The risk incurred by a sole proprietor, such as the Owner being able to change something without permission	Economics Risk Economics Risk is Risks that may affect the economic mechanism system, such as the ability to increase Mint token	Logical Issue Logical Issue is that can cause errors to core processing, such as any prior operations that cause background processes to crash.	Authorization Authorization is Possible pitfalls from weak coding allows unrelated people to take any action to modify the values.	Mathematical Mathematical Any erroneous arithmetic operations affect the operation of the system or lead to erroneous values.	Naming Conventions Naming Conventions naming variables that may affect code understanding or naming inconsistencies
Security Risk Security Risk of loss or damage if it's no mitigate	Coding Style Coding Style is Tips coding for efficiency performance	Best Practices Best Practices is suggestions for improvement	Optimization Optimization is performance improvement	Gas Optimization Gas Optimization is increase performance to avoid expensive gas	Dead Code 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	Use Custom Errors	-	Gas Optimization	-
SEC-02	Long revert strings	-	Gas Optimization	-



FULL AUDIT REPORT

SEC-01: Use Custom Errors

Vulnerability Detail	Severity	Location	Category	Status
Use Custom Errors	-	Check on finding	Gas Optimization	-

Finding:

```

127:         require(currentAllowance >= subtractedValue, "ERC20: decreased allowance
below zero");

142:         require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");

158:         require(owner != address(0), "ERC20: approve from the zero address");

159:         require(spender != address(0), "ERC20: approve to the zero address");

172:         require(currentAllowance >= amount, "ERC20: insufficient allowance");

```

Scenario:

-

Recommendation:

Instead of using error strings, to reduce deployment and runtime cost, you should use Custom Errors. This would save both deployment and runtime cost.

<https://blog.soliditylang.org/2021/04/21/custom-errors/>

Alleviation:

-

FULL AUDIT REPORT

SEC-02: Long revert strings

Vulnerability Detail	Severity	Location	Category	Status
Long revert strings	-	Check on finding	Gas Optimization	-

Finding:

```

127:         require(currentAllowance >= subtractedValue, "ERC20: decreased allowance
below zero");

142:         require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");

158:         require(owner != address(0), "ERC20: approve from the zero address");

159:         require(spender != address(0), "ERC20: approve to the zero address");

```

Scenario:

-

Recommendation:

Long revert strings can indeed increase gas costs for transactions because they are stored as part of the contract bytecode. To optimize gas usage

Alleviation:

-



FULL AUDIT REPORT

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

FULL AUDIT REPORT

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
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Complete	No risk

FULL AUDIT REPORT

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



FULL AUDIT REPORT



Visibility, Mutability, Modifier function testing

Components


 Contracts	 Libraries	 Interfaces	 Abstract
1	0	2	1

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			
20	0			
External	Internal	Private	Pure	View
9	17	0	0	14

StateVariables

Total	 Public
2	0



Capabilities

Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts	
0.8.18					
 Transfers ETH	 Low-Level Calls	 DelegateC all	 Uses Hash Functions	 ECRecover	 New/Create/Create2
 TryCatch	Σ Unchecked				
	yes				

FULL AUDIT REPORT

Contracts Description Table

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
L	totalSupply	External !		NO !
L	balanceOf	External !		NO !
L	transfer	External !	●	NO !
L	allowance	External !		NO !
L	approve	External !	●	NO !
L	transferFrom	External !	●	NO !
IERC20Metadata	Interface	IERC20		
L	name	External !		NO !
L	symbol	External !		NO !
L	decimals	External !		NO !
Context	Implementation			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
PAWS	Implementation	Context, IERC20Metadata		
L		Public !	●	NO !
L	name	Public !		NO !
L	symbol	Public !		NO !

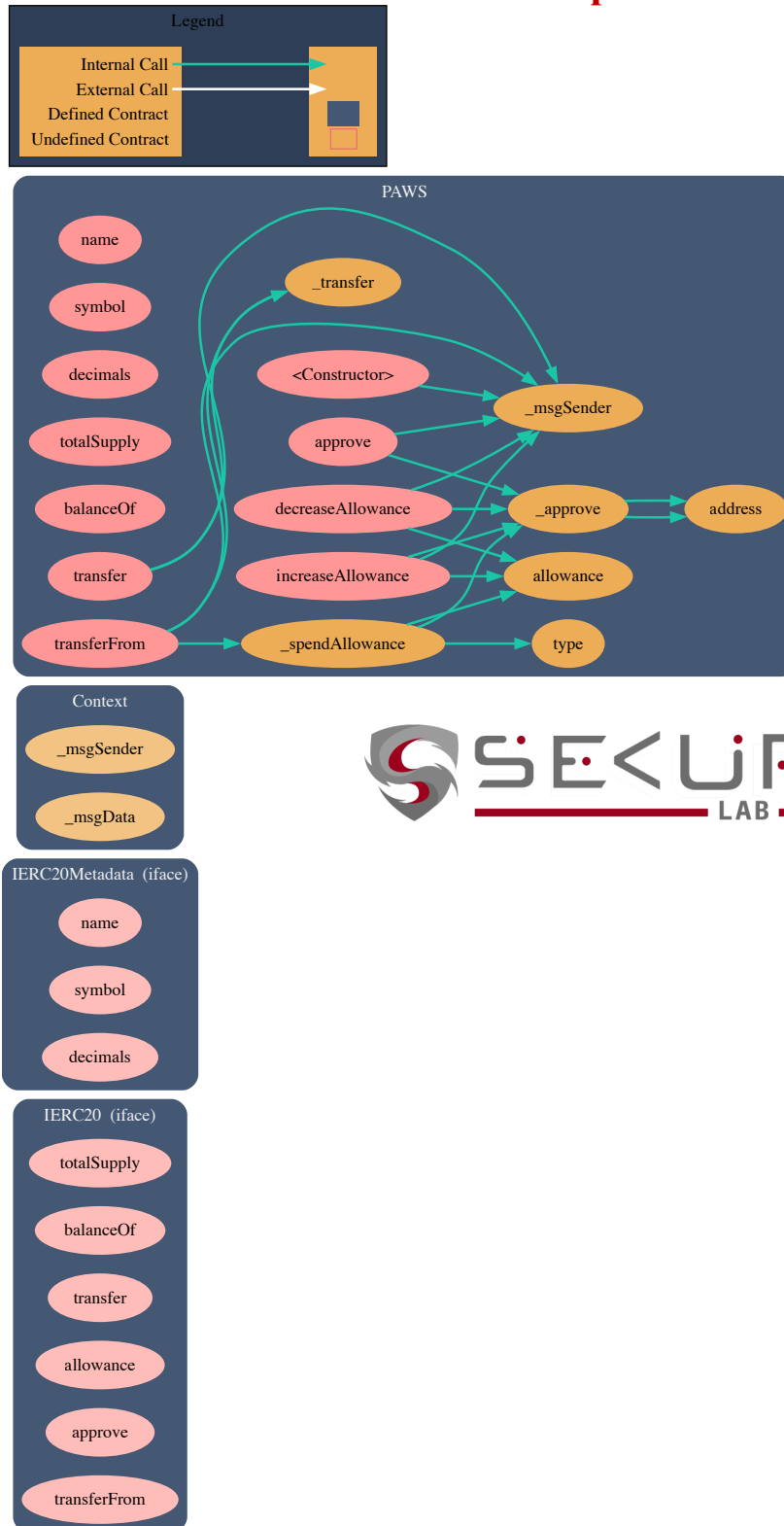
FULL AUDIT REPORT

Contract	Type	Bases		
L	decimals	Public !		NO !
L	totalSupply	Public !		NO !
L	balanceOf	Public !		NO !
L	transfer	Public !	🔴	NO !
L	allowance	Public !		NO !
L	approve	Public !	🔴	NO !
L	transferFrom	Public !	🔴	NO !
L	increaseAllowance	Public !	🔴	NO !
L	decreaseAllowance	Public !	🔴	NO !
L	_transfer	Internal 🔒	🔴	
L	_approve	Internal 🔒	🔴	
L	_spendAllowance	Internal 🔒	🔴	

Symbol	Meaning
🔴	Function can modify state
🔒	Function is payable

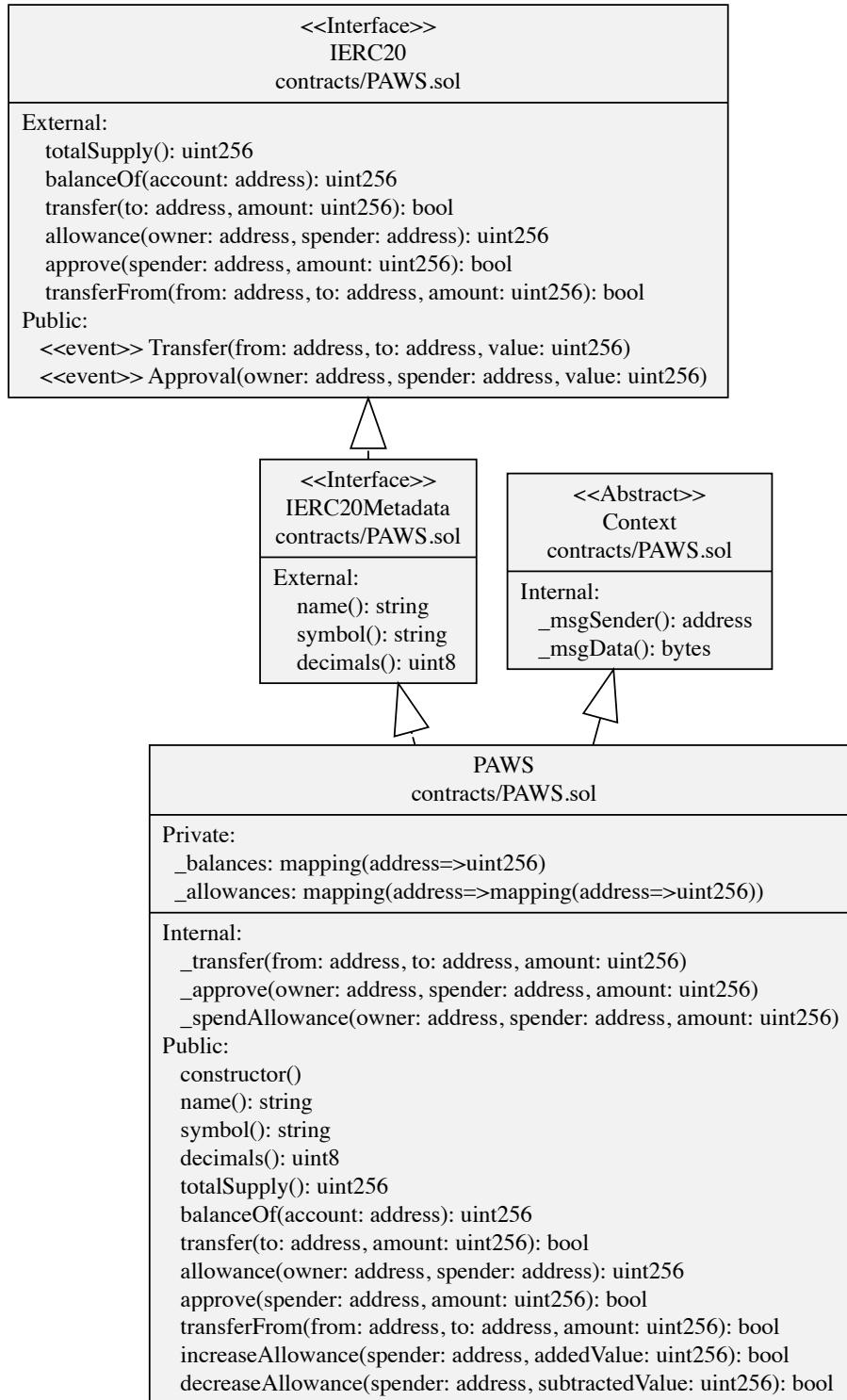
FULL AUDIT REPORT

Inheritate Function Relation Graph



FULL AUDIT REPORT

UML Class Diagram



FULL AUDIT REPORT

About SECURI LAB

Enhance the security and legitimacy of your blockchain project with our professional Audit & KYC services. Our experienced team provides reliable, cost-effective, and secure verification processes.



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