



SOLIDProof

Bring trust into your projects

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY

USDFI / STABLE

AUDIT

SECURITY ASSESSMENT

06. July, 2023

FOR



SolidProof_io



@solidproof_io

Introduction	3
Disclaimer	3
Project Overview	4
Summary	4
Social Medias	4
Audit Summary	5
File Overview	6
Imported packages	7
Audit Information	8
Vulnerability & Risk Level	8
Auditing Strategy and Techniques Applied	9
Methodology	9
Overall Security	10
Medium or higher issues	10
Upgradeability	11
Ownership	12
Ownership Privileges	13
Minting tokens	13
Burning tokens	14
Blacklist addresses	16
Fees and Tax	17
Lock User Funds	18
Components	19
Exposed Functions	19
Capabilities	20
Inheritance Graph	21
Centralization Privileges	22
Audit Results	24

Introduction

[SolidProof.io](#) is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.

Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

[SolidProof.io](#) reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	USDFi
Website	https://usdfi.com/
About the project	USDFi is crypto's first and only Universal Banking Protocol and DeFi's multichain native banking layer.
Chain	BSC
Language	Solidity
Codebase Link	
USDFI	https://bscscan.com/address/0x11a38e06699b238d6d9a0c7a01f3ac63a07ad318#code
STABLE	https://bscscan.com/address/0xa3870fbBeb730BA99e4107051612af3465CA9F5e#code
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/USDFI
Twitter	https://twitter.com/stable_usdfi
Facebook	N/A
Instagram	https://www.instagram.com/stable.usdfi/
Github	https://github.com/USDFI
Reddit	N/A
Medium	https://medium.com/@usdfi
Discord	https://discord.gg/MjvpF8UwB4
Youtube	https://www.youtube.com/@usdfi
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	03. July 2023	<ul style="list-style-type: none"> • Layout Project • Automated- /Manual-Security Testing • Summary

Note - This Audit report consists of a security analysis of the **USDFI / STABLE** smart contract. This analysis did not include functional testing (or unit testing) of the contract's logic. Stable is the same contract as USDFI.



File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/BlacklistRole.sol	de76f018d198f8d58b05e8ce3d9dda16ddc0433e
contracts/MinterRole.sol	967566ee4e4e97753e126e8b02e317feedc1f0c3
contracts/ManagerRole.sol	f764a364482bea8ac3e6eed8988fefe456b0bde3
contracts/Context.sol	74efdd28e3015f10d42a48996e00482e8b64e6f6
contracts/Manager.sol	ebfc918e902c4d9f14b9065ddfeb7f6728d5c907
contracts/Blacklist.sol	3b4f275e44d053dbd80cc64a969e6d4e9d84de39
contracts/SafeMath.sol	778e0c9037dbb03319f8723953e48a6ec47a0744
contracts/Ownable.sol	489ef4082cb177f2d1918d0831e6955f315a0ab1
contracts/Pausable.sol	039ac89372ceec8e5611fc08bfa931bb29fa0016
contracts/usdfi.sol	17996a4a683ab4e93f44ef6a4b79734a26d957de
contracts/ERC20.sol	80821c98273f6ef59c5566c011a24e4465d0374b
contracts/IERC20.sol	5b458789c5cfa979b09326d39883c4a54096bcfe
contracts/Roles.sol	4d7aea2cc0c839775c3c1c987eaf6bf2a38bceff

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.



Imported packages

Used code from other Frameworks/Smart Contracts (direct imports).

```
./Context.sol  
./IERC20.sol  
./SafeMath.sol  
./Manager.sol  
./Blacklist.sol
```

Note for Investors: We only Audited a token contract for **USDFi**. However, If the project has other contracts (for example, a Presale contract etc), and they were not provided to us in the audit scope, then we cannot comment on its security and are not responsible for it in any way.

Audit Information

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. The 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

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers and smart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - a. Reviewing the specifications, sources, and instructions provided to SolidProof to ensure we understand the size, scope, and functionality of the smart contract.
 - b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
 - c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.
2. Testing and automated analysis that includes the following:
 - a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
 - b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.
3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.
4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Medium or higher issues

No critical Issues found

 **Contract is already deployed**

Description

The contract does not contain issues of high or medium criticality. This means that no known vulnerabilities were found in the source code.

Comment

The owner of the contract will not redeploy the contract.



Upgradeability

Contract is not an upgradeable



Deployer cannot update the contract with new functionalities

Description

The contract is not an upgradeable contract. The deployer is not able to change or add any functionalities to the contract after deploying.

Comment

N/A





Ownership

The ownership is not renounced

✗ The owner is not renounce

Description

The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including:

- Centralizations
- The owner has significant control over contract's operations

Example

The owner can manually blacklist an address and then drain funds from that wallet to any arbitrary address.

Comment

The contract is heavily centralized

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

<div>Contract owner can mint new tokens</div> <div>✗ The owner able to mint new tokens</div>	
Description	Owners who have the ability to mint new tokens can reward themselves or other stakeholders, who can then sell the newly minted tokens on a cryptocurrency exchange to raise funds. However, there is a risk that the owner may abuse this power, leading to a decrease in trust and credibility in the project or platform. If stakeholders perceive that the owner is using their power to mint new tokens unfairly or without transparency, it can result in decreased demand for the token and a reduction in its value.
Example	If investors drive up the token price, the owner may choose to mint new tokens and sell them on a cryptocurrency exchange to raise funds. If the owner is not transparent and honest about their actions, they may be attempting a rugpull, where they suddenly abandon the project after raising funds, leaving investors with worthless tokens. This can lead to a decrease in the value of existing tokens, potentially rendering them worthless, and causing investors to suffer losses. It is essential for investors to carefully research the project and its developers and exercise caution before investing in any cryptocurrency or DeFi project.
Comment	The owner can mint unlimited tokens. Read the alleviation of the team at the bottom of the pdf.

File, Line/s: 338

Codebase: ERC20.sol

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner can burn tokens

✗ The owner able to burn tokens

Description

In some cases, burning tokens can be used as a tactic by the owner or developers to manipulate the token's value. If the owner or developers burn a significant number of tokens without transparency or justification, it can cause stakeholders to lose trust in the project and lead to a decrease in demand for the token.

It is important for projects and platforms to be transparent about their burning practices and ensure that they are justified and in the best interest of their stakeholders. If stakeholders perceive that the owner or developers are burning tokens unfairly or without transparency, it can cause a decrease in demand for the token and a reduction in its value. As such, investors should always do their due diligence and carefully research the project and its burning practices before investing in any cryptocurrency or DeFi project.

Example

For example, suppose an owner burns a large number of tokens without any explanation or communication to stakeholders. In that case, it can create speculation and uncertainty among investors, potentially causing them to sell their tokens and decreasing the token's value. Additionally, if the owner has a significant number of tokens themselves and burns them, it can lead to an unfair advantage and concentration of power, as the remaining tokens may become more valuable.

Therefore, it is crucial for projects and platforms to be transparent about their burning practices and ensure that they are justified and in the best interest of their stakeholders. Investors should carefully research the project and its burning practices and exercise caution before investing in any cryptocurrency or DeFi project to avoid potential losses.

If the owner is able to burn tokens without any allowances then he's able to burn any tokens what you bought in the past.

**Contract owner can
burn tokens****✗ The owner able to burn tokens**

Comment

In this case, the owner is able to blacklist addresses and then manually burn their tokens or worse, the owner is also able to redeem the complete balance of a blacklisted user

File, Line/s: 276**Codebase:** ERC20.sol

Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner can blacklist addresses	✗ The owner able to blacklist addresses
Description	<p>If the owner or developers of the smart contract abuse their power to blacklist addresses without proper justification or transparency, it can lead to a decrease in trust and credibility in the project. For example, suppose an owner or developer blacklists an address without proper explanation or communication to stakeholders. In that case, it can create speculation and uncertainty among investors, potentially causing them to sell their tokens and decreasing the token's value.</p> <p>Furthermore, if the owner or developers have a significant number of tokens themselves and use their power to blacklist competitors or manipulate the market, it can lead to an unfair advantage and concentration of power, potentially harming the interests of other stakeholders.</p> <p>Therefore, it is important for projects and platforms to be transparent about their blacklisting practices and ensure that they are justified and in the best interest of their stakeholders. Investors should carefully research the project and its blacklisting practices and exercise caution before investing in any cryptocurrency or DeFi project to avoid potential losses.</p>
Example	<p>The owner is able to grant blacklister role to any arbitrary address, and this privilege will stay with the wallet even after the renouncement of the ownership. Moreover, this address can blacklist wallets.</p>
Comment	<p>The minter address is able to burn tokens from the blacklist addresses. However, it is also possible for the minter address to drain the balance of blacklisted address to any arbitrary wallet of their choosing</p>



Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort in developing and maintaining the contract.

Contract owner cannot set fees more than 25%



The owner cannot levy unfair taxes

Description

The owner is not able to set the fees above 25%

Comment

There is no fee functionality



Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When tokens or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Contract owner can lock the user funds

✗ The owner is able to lock the contract

Description	Locking the contract means that the owner is able to lock any funds of addresses that they are not able to transfer bought tokens anymore.
Example	An example of locking is by pausing the contract or blacklisting any addresses. That causes that the blacklisted address is not able to transfer (buy/sell) anymore.
Comment	The manager can pause the transfer function manually and lock the transfer of tokens. Read the alleviation of the team at the bottom of the pdf.

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract's state. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be defined with a visibility modifier, such as public, private, or internal, which determines the access level of the variable.

Components

 Contracts	 Libraries	 Interfaces	 Abstract
8	2	1	2


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
42	0





External	Internal	Private	Pure	View
7	74	0	8	18

StateVariables

Total	 Public
14	2



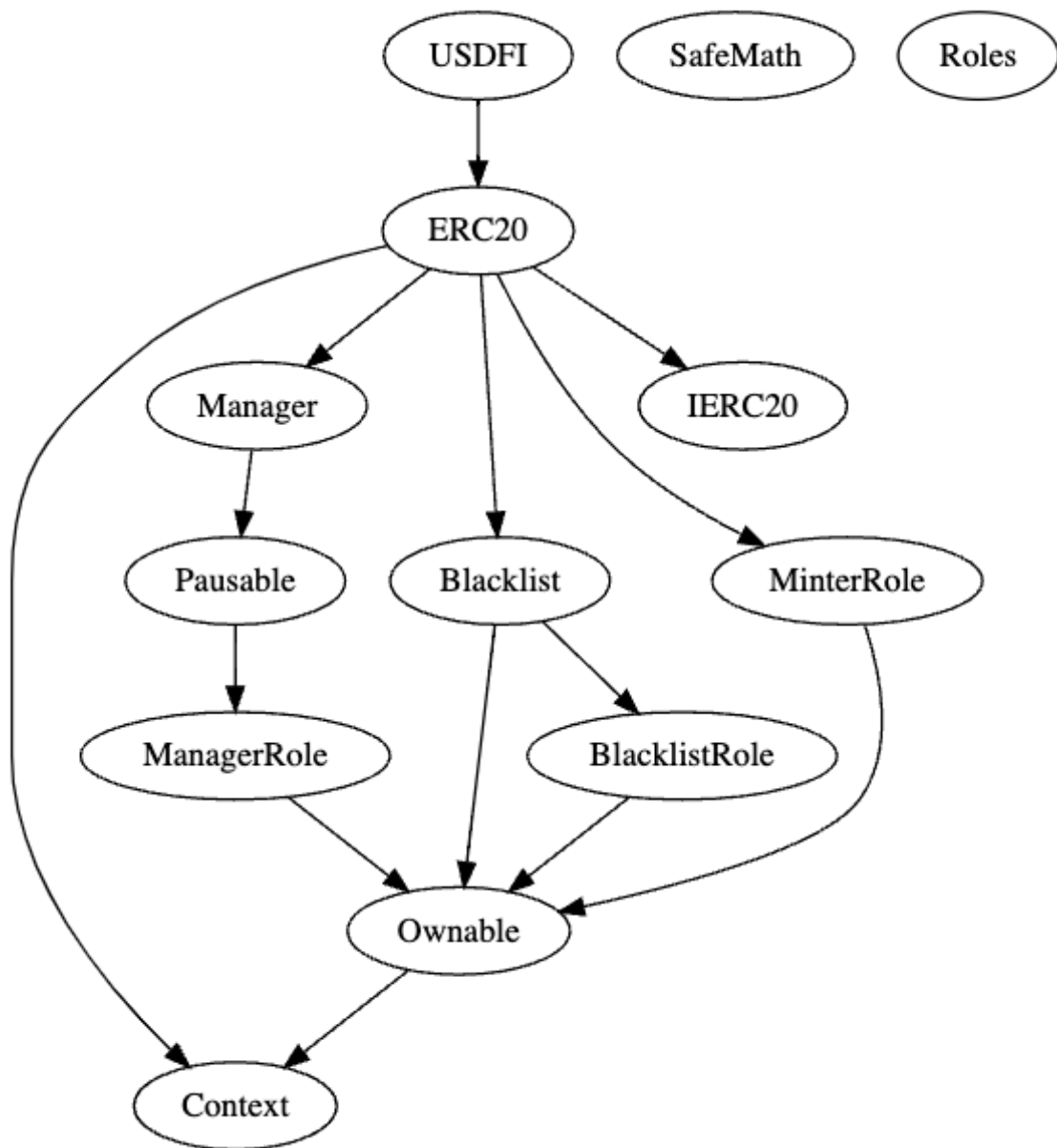
Capabilities

Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
0.6.12	-----	----	----	-----



Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methods from another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example, if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
1. ERC20.sol	<ul style="list-style-type: none"> ❖ onlyMinter <ul style="list-style-type: none"> - Mint Unlimited tokens - Burn tokens from blacklisted addresses - Transfer tokens from the blacklisted wallets to any other wallet
2. Blacklist.sol	<ul style="list-style-type: none"> ❖ onlyBlacklister <ul style="list-style-type: none"> - This address can add/remove wallets from the blacklist
3. BlacklistRole.sol	<ul style="list-style-type: none"> ❖ onlyOwner <ul style="list-style-type: none"> - This address can add/remove wallets from the blacklist
4. ManagerRole.sol	<ul style="list-style-type: none"> ❖ onlyOwner <ul style="list-style-type: none"> - The owner can add/remove wallets from the manager role, and the wallets with this role can pause the transfer function of the ERC20 contract

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g., Gnosis Safe
- Use of a timelock at least with a latency of e.g., 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement



- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.



Audit Results

#1 | Owner can drain tokens

File	Severity	Location	Status
ERC20	Low	L294	Acknowledged

Description - The contract owner can drain tokens from the user accounts by granting the minter and blacklist roles to an address (also from the zero/dead address) they control. Then that wallet will have the authority to blacklist addresses manually and then take out the funds from those wallets.

Alleviation - The Owner role is assigned to the DAO in the USDFI environment, ensuring that no single individual can access the role and make unilateral decisions. Every action must be approved in advance by a majority decision of the DAO. This ensures that no centralization of the protocol can occur.

Draining tokens from the wallet is essential for the protocol to ensure burning mechanisms. Similar functions have become standard for bridges.

USDFI-DAO-ADMIN

<https://app.safe.global/home?safe=bnb:0x06050eda6368855B29f7B00f41Bc5B234842f651>

#2 | Owner can mint tokens

File	Severity	Location	Status
ERC20	Low	L338	Acknowledged

Description - The contract owner can assign the minter role and then mint unlimited tokens without any limitations. We strongly recommend against this practice as this may be abused to manipulate the price of the token

Alleviation - The minting function is required to create new tokens to incentivize platform users. However, this function is restricted by the blacklist. The "minter" role is specifically designated to control the minting. The DAO can define a new identity as a minter, so this right must be transferrable. The Owner role does the assignment of this role.

#3 | Owner can lock funds

File	Severity	Location	Status
ERC20	Low	L338	Acknowledged

Description - The owner of the contract can assign the manager role to any arbitrary address, and that address can pause the transfer function in the contract, which will result in the loss of user funds

Alleviation - The blacklist is a central element within the protocol. It is the basis for many security mechanisms, such as bridges to other chains, swapping, burning, and minting. It only acts as a temporary lock for accessing tokens rather than a permanent lock. After 24 hours, users can remove themselves from the list.

This allows us to intervene in malicious transactions through a DAO decision. For example, the blacklist controls the stop function of the USDFI minter. Similar to the minting function, control over the blacklist is defined by a separate role that must be able to be passed on. Hence, the Owner role must have this capability.



#4 | Old Compiler version

File	Severity	Location	Status
All	Low	—	Acknowledged

Description

- The contracts use outdated compiler versions, which are not recommended for deployment as they may be susceptible to known vulnerabilities.



#5 | Missing zero/dead address check

File	Severity	Location	Status
Blacklist	Low	L30, L40	Acknowledged
ERC20	Low	L252-L253, L276, L295-L296,	Acknowledged
Manager	Low	L37, L57	Acknowledged
Manager Role	Low	L44, L55, L66, L77, L89	Acknowledged
MinterRole	Low	L44, L55, L66, L77, L89	Acknowledged

Description

- It is recommended to check the zero/dead address before blacklisting the address

#6 | Local Variable Shadowing

File	Severity	Location	Status
ERC20	Informational	L135, 381, 384, 385	Acknowledged

Description

- Rename the variable that shadows other components. In this case, the “owner” variable shadows the variable in the “Ownable” contract.

#7 | Contract doesn't import npm packages from source (like OpenZeppelin etc.)

File	Severity	Location	Status
All	Informational	N/A	Acknowledged

Description

- We recommend importing all packages from npm directly without flattening the contract. Functions could be modified or can be susceptible to vulnerabilities.

#8 | Check for minter before adding free mints to an address

File	Severity	Location	Status
Manager	Informational	L37	Acknowledged

Description

- Checking if the address is a minter before adding free mints to the address is recommended. Otherwise, the free mints would be useless for the address because only minters can mint new tokens.

#9 | Rename “sender” to “from”

File	Severity	Location	Status
ERC20	Informational	L295	Acknowledged

Description

- Since the caller is not the sender in the “redeemBlackFunds” the “sender” variable can be confusing. Renaming the “sender” to “from” is recommended.

#10 | Add blacklisted modifier

File	Severity	Location	Status
ERC20	Informational	L251, L277, ...	Acknowledged

Description

- Since the “isBlacklisted” function is called, often refactor the code so that the blacklist check is in a modifier. Add this modifier to all functions where you are checking the blacklist status of an address.



#11 | Missing state variable visibility

File	Severity	Location	Status
Blacklist	Informational	L18	Acknowledged

Description

- The state variable visibility is missing. It is recommended to set the visibility to any state variable.





Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.





**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY