



SOLIDProof

Bring trust into your projects

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

MADE IN GERMANY

XDuplify

AUDIT

SECURITY ASSESSMENT

28. 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	6
Audit Information	7
Vulnerability & Risk Level	7
Auditing Strategy and Techniques Applied	8
Methodology	8
Overall Security	9
Upgradeability	9
Ownership	10
Ownership Privileges	11
Minting tokens	11
Burning tokens	12
Blacklist addresses	13
Fees and Tax	14
Lock User Funds	15
Components	16
Exposed Functions	16
Capabilities	17
Inheritance Graph	18
Centralization Privileges	19
Audit Results	21



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	XDuplify
Website	https://www.xduplify.com/
About the project	XDuplify brings a redefined way to invest through profit sharing and dual investment from XDuplify. The double investment system designed by XDuplify is a first in the crypto industry by going against high-risk rewards and introducing a low-risk – higher rewards system instead. In short, greater profits with less capital at risk. We achieved this by introducing a profit share dual investment system in crypto.
Chain	Ethereum
Language	Solidity
Codebase Link	Provided as Files
Unit Tests	Not Provided

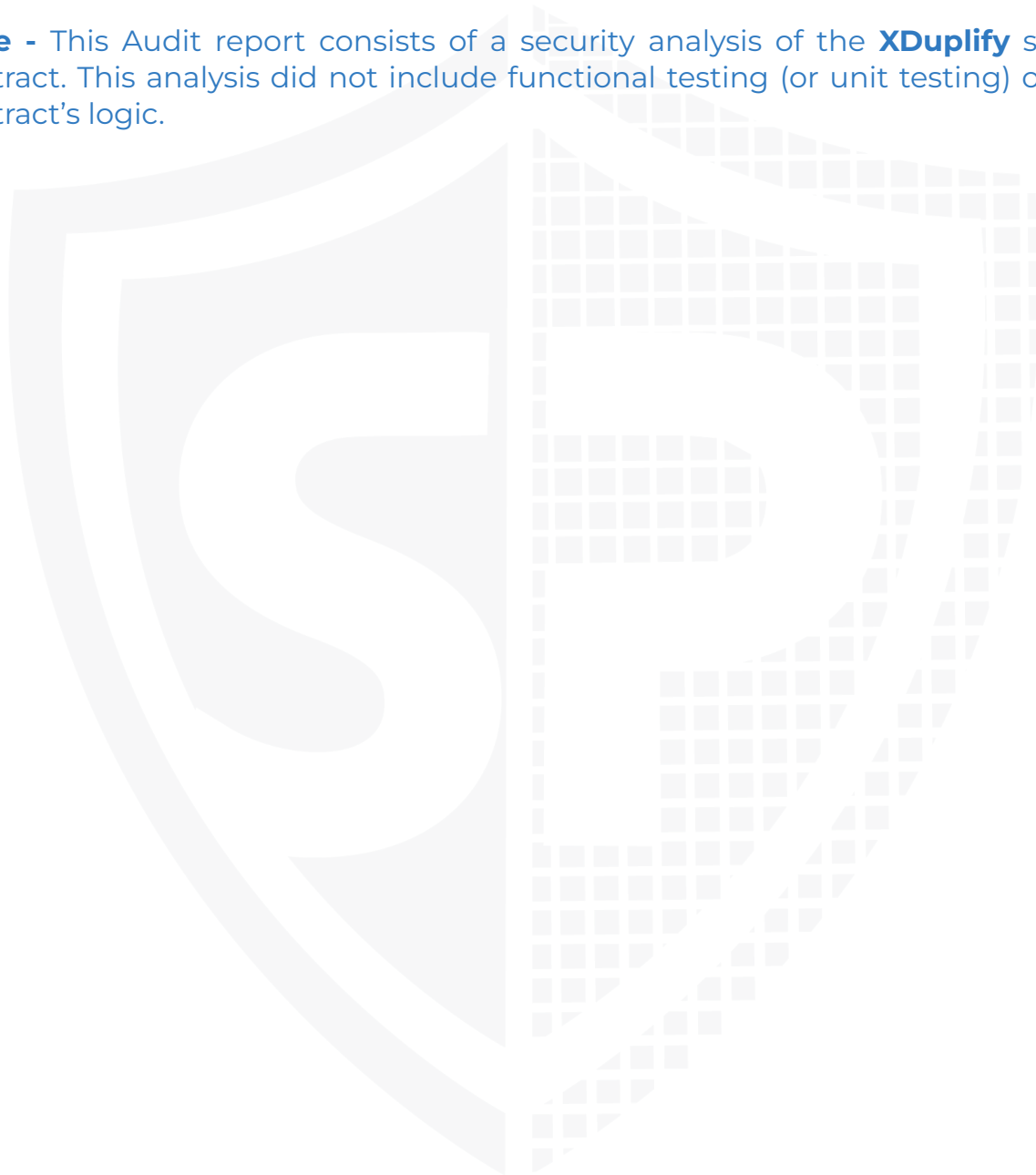
Social Medias

Telegram	https://t.me/Xduplify
Twitter	https://twitter.com/xduplify
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	05. July 2023	<ul style="list-style-type: none"> • Layout Project • Automated- /Manual-Security Testing • Summary
v1.1	24. July 2023	<ul style="list-style-type: none"> • Reaudit

Note - This Audit report consists of a security analysis of the **XDuplify** smart contract. This analysis did not include functional testing (or unit testing) of the contract's logic.





File Overview

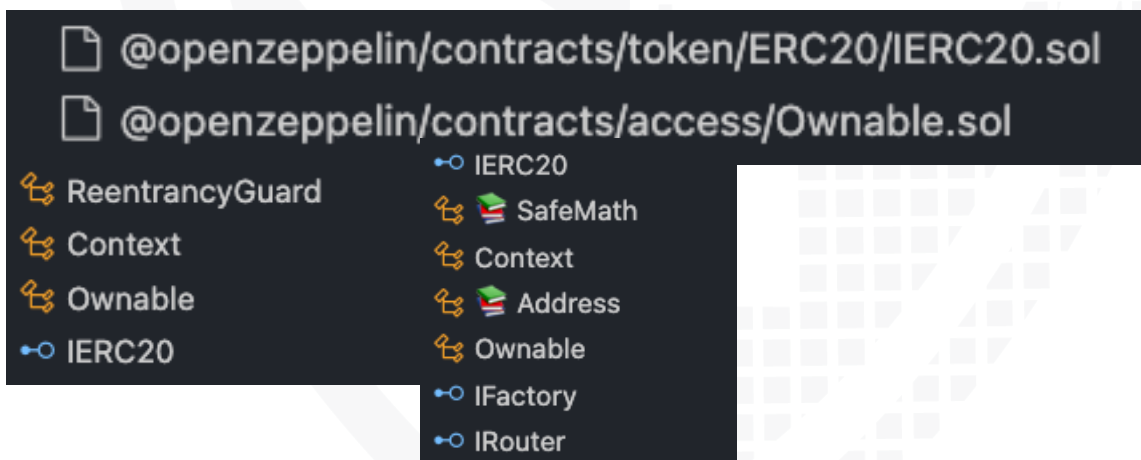
The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/Xduplify Lottery.sol	cc7b2968842c7e4dd68b76e176812a0eb729c166
contracts/UpdatedLoan.sol	c187cbd37a9c31bf4cfe12815ae431b5bc351b4e
contracts/Xduplify Token.sol	a81a443d02eaf7e238195de9730ec5b438c4955f

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).



Note for Investors: We only Audited a loan, lottery, and a Token token contract for **XDuplify**. However, If the project has other contracts (for example, a Presale or staking 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

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

We assume that you have funds in the contract and it has been audited by any security audit firm. Now the audit has passed. After that, the deployer can upgrade the contract to allow him to transfer the funds you purchased without any approval from you. This has the consequence that your funds can be taken by the creator.

Comment

N/A

Note - If the contract is not deployed then we would consider the ownership to be not renounced. Moreover, if there are no ownership functionalities then the ownership is automatically considered renounced.




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.

Contract owner cannot mint new tokens

 **The owner cannot mint new tokens**


Description	The owner is not able to mint new tokens once the contract is deployed.
-------------	-------------------------------------------------------------------------

Comment	N/A
---------	-----




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 cannot burn tokens		 The owner cannot burn tokens
Description	The owner is not able burn tokens without any allowances.	
Comment	N/A	

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	The owner is able to manually blacklist addresses from the loan contract, and those addresses won't be able to use the Loan contract's functionality anymore
Comment	N/A



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	N/A



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 withdraw tokens anymore.
Example	An example of locking is by blacklisting any addresses. That causes that the blacklisted address is not able to transfer (deposit/withdraw) anymore.
Comment	N/A

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
3	2	6	5


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
105	8




External	Internal	Private	Pure	View
60	120	22	27	40

StateVariables

Total	 Public
57	34



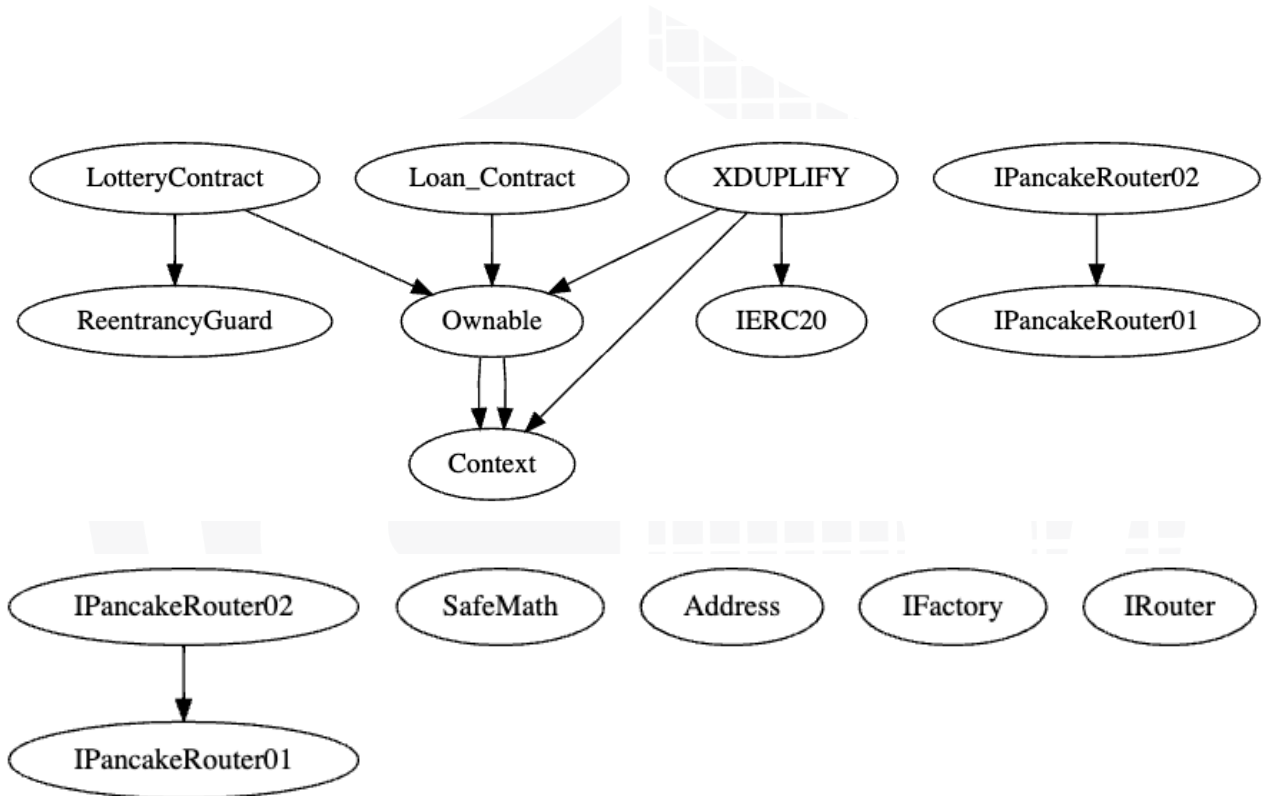
Capabilities

Solidity Versions observed	 Transfers ETH	 Can Receive Funds	 Delegate Call	 Has Destroyable Contracts
<code>^0.8.0</code> <code>^0.8.17</code> <code>^0.8.11</code>	YES	YES	YES	-----



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. XDuplify Token.sol	<ul style="list-style-type: none"> ❖ onlyOwner <ul style="list-style-type: none"> - Include/Exclude wallets from reward and fee - Set max buy and sell amount to a minimum of 10 - Set/Update liquidity threshold to any arbitrary value - Enable/Disable the swap - Set Router Address - Withdraw BNB from the contract's balance
2. UpdatedLoan.sol	<ul style="list-style-type: none"> ❖ onlyOwner <ul style="list-style-type: none"> - Add/Remove wallets from the blacklist - Add/Remove supported tokens - Withdraw Comission and Mortgage fund from the contracts balance - Add mortgage fund manually - Change Fund Manager address - The fund manager address can manually end deposit of any user if the loss is above 30%
1. XDuplify Lottery.sol	<ul style="list-style-type: none"> ❖ onlyOwner <ul style="list-style-type: none"> - Start and End Lottery at any time - Draw the lottery even before the end time - Withdraw complete BUSD balance from the contract - Update ticket limit to any arbitrary value including zero. Which will stop users from buying any tickets. - Set the price of a ticket to any arbitrary value except 0. This means that the owner is able to set the price very low and buy a lots of tickets for personal gain.

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 can no longer modify any state variables of the contract. Make sure to set up everything before renouncing.

Audit Results

#1 | Logical Flaw

File	Severity	Location	Status
UpdatedLoan	High	L382	Fixed

Description - The “_percentageGain” variable must be at least 10_000 otherwise the withdrawn amount will be equal to the invested amount which will result in no gain at the time of withdrawal.

Remediation - We recommend putting a check to verify the percentage gain before a withdrawal is available

#2 | Owner can lock funds

File	Severity	Location	Status
UpdatedLoan	Medium	L695	Open

Description - The owner can lock user funds by blacklisting the accounts, and once blacklisted after depositing funds in the contract, the users won't be able to withdraw their funds.

Remediation - We recommend allowing withdrawal for all addresses.

#3 | Trading needs to be enabled manually

File	Severity	Location	Status
Token	Medium	L672	Fixed

Description - The trading needs to be enabled by the owner in order for regular users to transfer tokens. On the contrary, the owner can exclude addresses from the fees manually and those addresses will be able to trade tokens. This functionality can be exploited in the following way, For example, there is a presale and the wallets used for the presale can be excluded from fees by the owner. All the tokens obtained can be consolidated into a final wallet address and facilitate trading and selling of the acquired tokens, the last wallet address can be excluded from fees.

#4 | Owner can Lock Funds

File	Severity	Location	Status
Token	Medium	L805	Fixed

Description - The owner can lock user funds by setting the max buy/sell amount to zero.

Remediation - We recommend putting a hard cap on the minimum amount.

Fix - The max buy and sell values can still be set to a bare minimum of 10.

#5 | Owner can drain tokens

File	Severity	Location	Status
Xduplify Lottery	Medium	L436	Open

Description - The owner of the contract is able to drain the complete balance of the contract by calling this function. Moreover, the owner can get the BUSD for their own account and if done so then the lottery contract will not be able to distribute any rewards.

Remediation - Make sure that it is not possible to take out the BUSD that is used for the lottery rewards. Moreover, the owner should only be able to get out any extra BUSD which will not affect the overall lottery prize distribution functionality

#6 | Owner can Lock Contract

File	Severity	Location	Status
Xduplify Lottery	Medium	L417	Open

Description - The owner can lock the lottery functionality setting the ticket limit amount to zero. Once done, then no user will be able to participate in the lottery.

Remediation - We recommend putting a hard cap on the minimum amount.

#7 | Missing Timelock

File	Severity	Location	Status
Xduplify Lottery	Medium	L325	Open

Description - The owner can lock draw the lottery at any time regardless of the end time set at the time of starting the lottery. Moreover, as mentioned in Issue #5 the owner can withdraw all the BUSD after selling the tickets and when the lottery is drawn, the winner will get nothing.

Remediation - We recommend putting in a check to verify that the lottery can only be drawn once the end time has passed.

#8 | Missing Zero Value Check

File	Severity	Location	Status
Xduplify Lottery	Low	L417	Open

Description - Make sure to validate that the value is not zero, otherwise no user will be able to buy any tickets.

#9 | Missing Events

File	Severity	Location	Status
UpdateLoan	Low	L695—759	Open

Description - Make sure to emit events for all the critical parameter changes in the contract to ensure the transparency and trackability of all the state variable changes in the contract.

#10 | Weak Randomization

File	Severity	Location	Status
Xduplify Lottery	Low	L336	Open

Description - The contract uses a weak on-chain randomization which is not recommended as it can be easily predictable as it is happening on the chain which is visible to the public.

#11 | Missng “isContract” check

File	Severity	Location	Status
UpdateLoan	Low	L687	Open

Description

- The contract doesn’t have any checks to verify whether the endDeposit function is being called by an EOA or a contract.

Remediation - We recommend putting a check to verify that the caller of the function must be an EOA to avoid the risk of bots.

#12 | Floating Pragma

File	Severity	Location	Status
All	Informational	L2	Open

Description - The current pragma Solidity directives are “^0.8.0 and ^0.8.11”. Contracts should be deployed with the same compiler version and flag that they have been tested thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using other versions

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

File	Severity	Location	Status
Xduplify Token	Informational	N/A	Open

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

Recommendation/Warning for the Project Team - “The LOAN contract uses the “getAmountsOut” function which is vulnerable to the known Flash Loan and Oracle Manipulation attacks as an attacker can manipulate the price of an asset by taking a flash loan. We would advise you to proceed with caution with this function as there is no fix available for it now. The fix can be done by changing the business logic and not using vulnerable functions.”



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