GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

UltiBets

Betting on Blockchain

Summary

Audit Firm: Guardian Audits

Client Firm: UltiBets

Final Report Date

Audit Summary

After a line by line manual analysis and automated review, Guardian Audits has concluded that:

- UltiBets's smart contracts have an HIGH RISK SEVERITY
- UltiBet's smart contracts have an **ACTIVE OWNERSHIP**
- UltiBets's smart contract owner has multiple "write" privileges. Centralization risk correlated to the active ownership is **VERY HIGH**

Notice that the examined smart contracts are not resistant to internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Fantom Opera**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

Table of Contents

Project Overview

Project Summary

Project Name	UltiBets
Language	Solidity
Codebase	https://github.com/UltiBets/Audits
Commit	3b44d739f6b725199871cdd3048312f57bd4f7ba

Audit Summary

Delivery Date	Preliminary Report
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	3	3	0	0	0	0
High	11	11	0	0	0	0
Medium	3	3	0	0	0	0
Low	21	21	0	0	0	0

Audit Scope & Methodology

Scope

ID	File	SHA-1 Checksum
UB	UltiBets.sol	72075D675DC6F45EDA17CAD8C0FE2CF8828D9B42
UBF	UltiBetsFactory.sol	41D01179EB175CDC00E8C4F0D7AA052863BAA34E
UBT	UltiBetsTreasury.sol	E5ED8B42BB423E026EF67262D503F6ABCBE9DBD4
CA	CustomAdmin.sol	8BE5A5A0387D95B325CD4BFAE925363F0B45799B
20A	ERC20Airdrop.sol	B5061E18E3250A6BB4A0EE6A8279F991E0534591
721A	ERC721Airdrop.sol	B544C544C6BE2E0CB2A46B8680222B31E953B3B8
MS	Multisig.sol	6C505A9EE0D08D95302B6A839E0D07C9BB48DA97
SQDR	SquidBetPlayersRegistration.sol	80E0BE67CAE797DFAE168808EB09DF64D280EC7F
SQD1	SquidBetStartRound.sol	2462CBDC917ED51698B853E2BA71712BBA5C477A
SQD2	SquidBetSecondRound.sol	0935D1CB4E4D13738626D2620156B74E1B12A772
SQD3	SquidBetThirdRound.sol	075EA8E69C10446A7AC7AC7C839B1C2DD05254A9
SQD4	SquidBetForthRound.sol	A1A8C6298B7CD4A3E9497FA211F15495278046EC
SQDF	SquidBetFinalRound.sol	8E836837749EB320FDFFE4AC79AF23C4C61BCABF
SQDP	SquidBetPrizePool.sol	56F74199AAC8DD56F4AEFD334B341421D135D070

Audit Scope & Methodology

Methodology

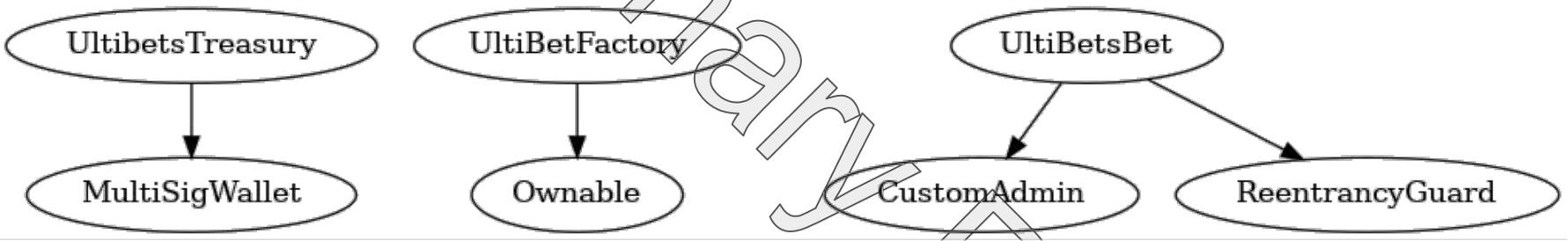
The auditing process pays special attention to the following considerations:

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

Vulnerability Classifications

Vulnerability Level	Classification
🔴 Critical	Easily exploitable by anyone, causing loss/manipulation of assets or data.
🟠 High	Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data.
🟡 Medium	Inherent risk of future exploits that may or may not impact the smart contract execution.
🟢 Low	Minor deviation from best practices.

Inheritance Graph



Findings & Resolutions

ID	Title	Category	Severity	Status
GLOBAL-1	Poor Practices	Best Practices	Medium	Unresolved
GLOBAL-2	Centralization Risk	Centralization / Privilege	High	Unresolved
UB-1	DoS Attack	Denial-of-Service	High	Unresolved
UB-2	DoS Attack	Denial-of-Service	High	Unresolved
UB-3	Lost Funds On Cancel	Logical Error	High	Unresolved
UB-4	Requires in For Loop	Best Practices	Low	Unresolved
UB-5	Inaccurate Enum Comment	Inaccurate Comments	Low	Unresolved
UB-6	Declare Variables Immutable	Best Practices	Low	Unresolved
UBT-1	Stuck ETH Funds	Logical Error	Critical	Unresolved
UBT-2	Lack of Access Control	Logical Error	Critical	Unresolved
UBT-3	Payment Pushed Back	Centralization / Privilege	Medium	Unresolved
UBT-4	Arbitrary Salary	Logical Error	Critical	Unresolved
UBT-5	Set Withdrawal Frequency	Best Practices	Low	Unresolved

Findings & Resolutions

ID	Title	Category	Severity	Status
UBT-6	Declare Variable Immutable	Best Practices	Low	Unresolved
MS-1	Principal Admin Abuse	Centralization / Privilege	High	Unresolved
MS-2	Lack of 0 Check	Best Practices	Low	Unresolved
MS-3	Break For Loop	Optimization	Low	Unresolved
20A-1	Unnecessary transferFrom	Best Practices	Low	Unresolved
721A-1	Wrong Token Sent	Logical Error	High	Unresolved
721A-2	Inaccurate Comment	Inaccurate Comments	Low	Unresolved
CA-1	Inaccurate Comment	Inaccurate Comments	Low	Unresolved
SQD[1-F]-1	Cannot Stop Betting	Logical Error	High	Unresolved
SQD[1-F]-2	Invalid Bet Side Management	Logical Error	Low	Unresolved
SQD[1-F]-3	Winner Can Be Changed	Logical Error	High	Unresolved
SQD[1-F]-4	Unnecessary For Loop	Optimization	Medium	Unresolved
SQDF-1	Winner Can Be Changed	Logical Error	High	Unresolved

Findings & Resolutions

ID	Title	Category	Severity	Status
SQDF-2	Weak Source of Randomness	Randomness	High	Unresolved
SQDF-3	Unnecessary Casting	Best Practices	Low	Unresolved
SQDF-4	Unnecessary For Loop	Optimization	Low	Unresolved
SQDF-5	Unnecessary Variable	Optimization	Low	Unresolved
SQDR-1	Unnecessary Increment	Optimization	Low	Unresolved
SQDR-2	Inaccurate Error Message	Inaccurate Message	Low	Unresolved
SQDR-3	Invalid Player Number Management	Logical Error	Low	Unresolved
SQDR-4	Function Typo	Typo	Low	Unresolved
SQDR-5	Redundant Variables	Optimization	Low	Unresolved
SQPR-1	Lack of 0 Checks	Best Practices	Low	Unresolved
SQPR-2	Winners Can't Get Prize	Denial-of-Service	High	Unresolved
SQPR-3	Unnecessary Function	Optimization	Low	Unresolved

GLOBAL-1 | Poor Practices

Category	Severity	Location	Status
Best Practices	Medium	Global	Unresolved

Description

Throughout the contracts there are myriad instances of:

- Lack of camelcase
- Unnecessary local variables which waste gas
- Redundant boolean checks e.g. `== true`
- Typos in the comments
- Unnecessary for-loops which waste gas and enable DoS
- Inefficient computations e.g. `feeBalance -= feeBalance`
- Many functions can be declared `external`

Recommendation

Use camelcase throughout the contracts, remove redundant and unnecessary local variables, do not perform redundant boolean checks, revise comments, refactor operations to avoid unnecessary for-loops, avoid inefficient computations e.g. use `feeBalance = 0`, declare all functions that are not called within the contracts `external` rather than `public`

Resolution

GLOBAL-2 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	High	Global	Unresolved

Description

Throughout the contracts there is a risk of admins and oracles using their privilege to benefit themselves or act malicious toward user's holdings. Contracts utilizing the CustomAdmin access control model face more risk as the number of admins or oracles increase because it only takes one to act mischievous.

Additionally, the treasury which is under the in-house MultiSig.sol contract can be can be manipulated by the principal admin who can unethically force a quorum.

Recommendation

Ensure that privileged addresses such as admins are all a multi-sig and/or introduce a timelock for improved community oversight. Secure a KYC for increased community trust.

Resolution

UB-1 | DoS Attack

Category	Severity	Location	Status
Denial-of-Service	● High	UltiBets.sol:174	Unresolved

Description

A malicious actor can call placeBet with different addresses, sending a tiny amount of ETH per call.

As a result, the YesBettors array and NoBettors array will expand to the point where it either exceeds the block gas limit, or costs too much to reportResult. This will render the function inoperable.

Recommendation

Use a pull-over-push withdrawal pattern such that the “for” loop can be avoided.

Resolution

UB-2 | DoS Attack

Category	Severity	Location	Status
Denial-of-Service	● High	UltiBets.sol:194	Unresolved

Description

`feeBettorBet` is calculated based on total amount bet for a user i.e. winning side + losing side.

If the user bet more on the losing side, it is possible for the `feeBettorBet` to exceed the amount bet on the winning side, causing a subtraction underflow which will revert. Therefore, reportResult will consistently fail and users will not get their winnings.

Recommendation

Calculate the fee based on the user's bet for the winning side. Or, place a cap on the fee.

Resolution

UB-3 | Lost Funds On Cancel

Category	Severity	Location	Status
Logical Error	● High	UltiBets.sol:160	Unresolved

Description

The contract allows placing bets on both sides which makes sense if the user would like to perform arbitrage. If the event is canceled upon an emergency, a user can only withdraw funds from one side. If they were betting on both sides for arbitrage, they lose the amount they bet on the other side.

Recommendation

Don’t set both sides of a user’s bet to 0.

Resolution

UB-4 | Requires in For Loop

Category	Severity	Location	Status
Best Practices	● Low	UltiBets.sol:195-198, 215-218,	Unresolved

Description

Place the `require` statements before the if statement in `reportResult`. It is not necessary to check those conditions upon each loop and also redundant to have them in loops for both sides while being extremely gas expensive.

Recommendation

Move the `require` statements to the top of the function.

Resolution

UB-5 | Inaccurate Enum Comment

Category	Severity	Location	Status
Inaccurate Comments	Low	UltiBets.sol:105	Unresolved

Description

The comment states that 1 represent `YES` and 0 represents `NO`. Because `YES` is the the first and default value of the enum, `YES` is 0 and 1 is `NO`.

Recommendation

Update the comment to accurately reflect the enum.

Resolution

UB-6 | Declare Variables Immutable

Category	Severity	Location	Status
Best Practices	● Low	UltiBets.sol:41	Unresolved

Description

`feePercentage` and `UltibetsTreasury` are never mutated once set.

Recommendation

Add the “immutable” keyword to `UltiBetsTreasury` and add the “constant” keyword to `feePercentage` since it is not being set in the constructor..

Resolution

UBT-1 | Stuck ETH Funds

Category	Severity	Location	Status
Logical Error	● Critical	UltiBetsTreasury.sol	Unresolved

Description

There is no way to withdraw the Ether sent to the treasury. Contracts like `SquidBetPrizePool` send ether to the treasury when `EmergencySafeWithdraw` is called.

Recommendation

Add a function to convert the Ether to the `fundingToken`, or implement allocations to be able to use the Ether.

Resolution

UBT-2 | Lack of Access Control

Category	Severity	Location	Status
Logical Error	● Critical	UltiBetsTreasury.sol	Unresolved

Description

Functions `deleteAllocation` and `changeSalary` have no Admin requires so anyone can delete an allocation and change a team member's salary.

Recommendation

Add a check that the `msg.sender` is an admin.

Resolution

UBT-3 | Payment Pushed Back

Category	Severity	Location	Status
Centralization / Privilege	Medium	UltiBetsTreasury.sol	Unresolved

Description

Admin can just keep calling createAllocation so the allocation for a particular address cannot be withdrawn as payoutday keeps getting pushed back.

Recommendation

Adopt a solution that doesn't allow such manipulation, or ensure trust via a multi-sig for every privileged address.

Resolution

UBT-4 | Arbitrary Salary

Category	Severity	Location	Status
Logical Error	● Critical	UltiBetsTreasury.sol:184	Unresolved

Description

The `salary` could be set arbitrarily high before someone calls `withdraw` to drain the `fundingToken` balance. The `salary` could also be set arbitrarily high to prevent withdrawal during `totalpayout` calculation by causing an overflow.

Recommendation

Add a cap to the salary and restrict access to `changeSalary`.

Resolution

UBT-5 | Set Withdrawal Frequency

Category	Severity	Location	Status
Best Practices	● Low	UltiBetsTreasury.sol:	Unresolved

Description

withdrawalFrequency can be set in the constructor. It doesn't need to be updated each time an allocation is created. Especially since withdrawalFrequency is used in other functions like withdraw

Recommendation

Set the withdrawal frequency in the constructor.

Resolution

UBT-6 | Declare Variable Immutable

Category	Severity	Location	Status
Best Practices	Low	UltiBetsTreasury.sol:13	Unresolved

Description

`Admin` is not mutated outside of the constructor so it can be declared `immutable`.

Recommendation

Declare Admin with the `immutable` keyword.

Resolution

MS-1 | Principal Admin Abuse

Category	Severity	Location	Status
Centralization / Privilege	High	MultiSig.sol:51	Unresolved

Description

The principalAdmin can keep calling setApproverAddr to add as many addresses as needed to reach quorum and approve any arbitrary transaction. This is not a true multi-sig if a principal admin can obtain all the power and decision making.

Recommendation

Make it so a majority of admins must agree to add or remove another admin or partake in other important decisions with the treasury.

Resolution

MS-2 | Lack of 0 Check

Category	Severity	Location	Status
Best Practices	Low	MultiSig.sol:19	Unresolved

Description

There is no check to make sure that `quorum` is not set to 0.

Recommendation

Add a require that `quorum` is greater than 0.

Resolution

MS-3 | Break For Loop

Category	Severity	Location	Status
Optimization	Low	MultiSig.sol:32	Unresolved

Description

If you found that the msg.sender is an admin you can break out of the for loop to save gas.

Recommendation

Break out of the for loop once the msg.sender is verified to be an admin

Resolution

20A-1 | Unnecessary transferFrom

Category	Severity	Location	Status
Best Practices	Low	ERC20Airdrop.sol:65	Unresolved

Description

Because the transfer is from the current address to another address, the `ERC20` function `transfer` could be used.

Recommendation

Replace the use of `transferFrom` with `transfer`. Be sure to check the return value or opt for a `safeTransfer` alternative.

Resolution

721A-1 | Wrong Token Sent

Category	Severity	Location	Status
Logical Error	High	ERC721Airdrop.sol:56	Unresolved

Description

The `tokenId+1` is sent to the caller although it was not used in the check for a valid leaf. Instead, `tokenId` was verified to correspond with the `msg.sender`.

Recommendation

Send the current `tokenId` to `msg.sender`.

Resolution

721A-2 | Inaccurate Comment

Category	Severity	Location	Status
Inaccurate Comments	● Low	ERC721Airdrop.sol:42	Unresolved

Description

The comment states that the proof is to check that the address and the `amount` are in tree. However, for the ERC721 airdrop, you are checking if the address and `tokenId` are in the tree.

Recommendation

Update the comment to reflect the `ERC721` NFT airdrop.

Resolution

CA-1 | Inaccurate Comment

Category	Severity	Location	Status
Inaccurate Comments	Low	CustomAdmin.sol:46	Unresolved

Description

The comment states that the function adds the specified address to the list of `administrators` but it adds the address to the mapping of `Oracles`.

Recommendation

Update the comment to reflect what the function does.

Resolution

SQD[1-F]-1 | Cannot Stop Betting

Category	Severity	Location	Status
Logical Error	High	SquidBet*Round.sol	Unresolved

Description

In the placeBet function there is no check to ensure that isEventCancelled is false.

Recommendation

Add a require statement to all of these placeBet functions such that you cannot place a bet if the event is cancelled.

Resolution

SQD[1-F]-2 | Invalid Bet Side Management

Category	Severity	Location	Status
Logical Error	Low	SquidBet*Round.sol	Unresolved

Description

In the placeBet function the following assignment playerSide[msg.sender] += choice should simply read playerSide[msg.sender] = choice

Recommendation

Refactor this line to be playerSide[msg.sender] = choice.

Resolution

SQD[1-F]-3 | Winner Can Be Changed

Category	Severity	Location	Status
Logical Error	High	SquidBet*Round.sol	Unresolved

Description

Once the event is finished, it is possible to for a malicious `oracle` to call `reportResult` several times to add both sides as the winner.

Recommendation

Prevent the `result` from being changed by using a variable to check if the `result` was already reported.

Resolution

SQD[1-4]-4 | Unnecessary For Loop

Category	Severity	Location	Status
Optimization	Medium	SquidBet*Round.sol	Unresolved

Description

In `reportResult` the `for` loop that loops through each bettor in the `playersSide` mapping is gas expensive and unnecessary.

Recommendation

Infer whether or not players are winners based on the `result` contract variable, don't maintain the `iswinner` mapping.

Resolution

SQDF-1 | Winner Can Be Changed

Category	Severity	Location	Status
Logical Error	High	SquidBetFinalRound.sol: 111	Unresolved

Description

Once the event is finished, it is possible to for a malicious `oracle` to repeatedly call `reportResult` to modify which side is the winner.

Recommendation

Prevent the `result` from being changed by using a variable to check if the `result` was already reported.

Resolution

SQDF-2 | Weak Source of Randomness

Category	Severity	Location	Status
Randomness	High	SquidBetFinalRound.sol: 193	Unresolved

Description

`pickWinner` uses weak sources of on-chain randomness. A validator can exploit this in order to obtain a winner that is beneficial to themselves.

In addition, an Admin can keep calling `pickWinner`, then `reportResult` to clear `isCompetitionEnded`, then call `pickWinner` again and so on until the winner is favorable to them.

Recommendation

Utilize a strong source of randomness whether it be the on-chain randomness pattern or an oracle. In addition, prevent repeated calls to `pickWinner` by tracking whether a winner was already chosen.

Resolution

SQDF-3 | Unnecessary Casting

Category	Severity	Location	Status
Best Practices	Low	SquidBetFinalRound.sol:162-164	Unresolved

Description

There is no need to cast a positive integer to a `uint`.

Recommendation

Remove the unnecessary `uint` surrounding 1 and 2.

Resolution

SQDF-4 | Unnecessary For Loop

Category	Severity	Location	Status
Optimization	Low	SquidBetFinalRound.sol:162-166	Unresolved

Description

The `for` loop over an arbitrary number of votes can be avoided by tallying the votes in the `Vote` function.

Recommendation

Get rid of the `for` loop and move on-demand tallying logic to `Vote`.

Resolution

SQDF-5 | Unnecessary Variable

Category	Severity	Location	Status
Optimization	Low	SquidBetFinalRound.sol	Unresolved

Description

The `playerVote` state variable is never meaningfully used.

Recommendation

Remove the `playerVote` variable.

Resolution

SQDR-1 | Unnecessary Increment

Category	Severity	Location	Status
Optimization	Low	SquidBetPlayersRegistration.sol: 30	Unresolved

Description

It is a waste of gas to initialize the `nextPlayerNumber` to 0 and then immediately increment it in the constructor.

Recommendation

Simply initialize `nextPlayerNumber` to 1 rather than initializing it to 0 and spending gas to increment it in the constructor.

Resolution

SQDR-2 | Inaccurate Error Message

Category	Severity	Location	Status
Inaccurate Message	Low	SquidBetPlayersRegistration.sol: 45	Unresolved

Description

The error message indicates that the cost is 0.01 Ether while it is in fact 1 Ether.

Recommendation

Update the message to reflect the real cost.

Resolution

SQDR-3 | Invalid Player Number Management

Category	Severity	Location	Status
Logical Error	Low	SquidBetPlayersRegistration.sol: 55	Unresolved

Description

The `playersNumbers` mapping is updated as follows: `playersNumbers[msg.sender] += nextPlayerNumber`. It should instead be assigned like so `playersNumbers[msg.sender] = nextPlayerNumber`

Recommendation

Update the assignment to use `=` rather than `+=`.

Resolution

SQDR-4 | Function Typo

Category	Severity	Location	Status
Typo	Low	SquidBetPlayersRegistration.sol: 63	Unresolved

Description

The function `getIsRegisterdPlayer` contains a typo

Recommendation

Update the name to be `getIsRegisteredPlayer`.

Resolution

SQDR-5 | Redundant Variables

Category	Severity	Location	Status
Optimization	Low	SquidBetPlayersRegistration.sol	Unresolved

Description

The `numberOfPlayersRegistered` variable can simply be derived as one less than the `nextplayerNumber` and is therefore unnecessary.

Recommendation

Remove the `numberOfPlayersRegistered` variable and rely on the `nextplayerNumber - 1`.

Resolution

SQPR-1 | Lack of 0 Checks

Category	Severity	Location	Status
Best Practices	Low	SquidBetPrizePool.sol	Unresolved

Description

Nothing prevents the UltiBetsTreasury from being set to the zero address in the constructor.

In addWinnerAddress, the winner address can be set to 0.

Recommendation

Add zero address checks using require statements.

Resolution

SQPR-2 | Winners Can’t Get Prize

Category	Severity	Location	Status
Denial-of-Service	● High	SquidBetPrizePool.sol:61-68	Unresolved

Description

If there are a large amount of winners, the amount of gas can exceed the block limit and winners will not be able to get the prize money,

Recommendation

Utilize a pull-over-push withdrawal patten.

Resolution

SQPR-3 | Unnecessary Function

Category	Severity	Location	Status
Optimization	● Low	SquidBetPrizePool.sol	Unresolved

Description

There is not a need to have both `winnerClaimPrizePool` and `winnersClaimEqualPrizePool`. A single winner address could be stored as an “equal” winner and the prize money would just be sent to that single address.

Recommendation

Remove the need to store the `winnerAddress` and solely store winners in `equalWinners`.

Resolution

Auditor’s Verdict

After a line by line manual analysis and automated review, Guardian Audits has concluded that:

- UltiBets’s smart contracts have an EXTREMELY HIGH RISK SEVERITY
- UltiBet’s smart contracts have an **ACTIVE OWNERSHIP**
- UltiBets’s smart contract owner has multiple “write” privileges. Centralization risk correlated to the active ownership is **VERY HIGH**

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits