



# SMART CONTRACT SECURITY AUDIT OF



# Summary

**Audit Firm:** Guardian Audits

**Client Firm:** Bridges

**Final Report Date - May 21, 2022**

## Audit Summary


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

- Bridges' smart contracts have a **MEDIUM RISK SEVERITY**
- Bridges' smart contracts have an **ACTIVE OWNERSHIP**
- Bridges' smart contract owner has multiple "write" privileges. Centralization risk correlated to the active ownership is **MEDIUM**

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: **BSC**

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

 Comprehensive penetration + fuzzing test suite:  
<https://github.com/GuardianAudits/BridgesTestSuite>

# Table of Contents

## Project Information

Project Overview ..... 4

Audit Scope & Methodology ..... 5

## Smart Contract Risk Assessment

Inheritance Graph ..... 7

Findings & Resolutions ..... 8

## Report Summary

Auditor’s Verdict ..... 34

## Addendum

Disclaimer ..... 35

About Guardian Audits ..... 36

# Project Overview

## Project Summary

Project Name	Bridges
Language	Solidity
Codebases	<a href="https://github.com/bridges-team/bridges-exchange-farm">https://github.com/bridges-team/bridges-exchange-farm</a> <a href="https://github.com/bridges-team/TokenVesting">https://github.com/bridges-team/TokenVesting</a> <a href="https://github.com/bridges-team/bridges-exchange-periphery">https://github.com/bridges-team/bridges-exchange-periphery</a> <a href="https://github.com/bridges-team/bridges-exchange-swap-core">https://github.com/bridges-team/bridges-exchange-swap-core</a>
Commits	a7ee7dbb8a95d27fcb99f04729e816e3d9d590eb 699436ff2cb35716c887ae653b55a68659d2e4d1 2696b09d53316682341c5d7187553a20428293b8 37c349a1b455a647fe3244ea7072d730d0280de7

## Audit Summary

Delivery Date	May 21, 2022
Audit Methodology	Static Analysis, Manual Review, Fuzzing

## Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	2	0	0	0	0	2
● Medium	9	0	0	1	1	7
● Low	12	0	0	0	0	12

# Audit Scope & Methodology

## Scope

ID	File	SHA-1 Checksum
GG	GoldenGate.sol	6DB849074A31AFCA02CFE7DDB36062A21B3F9BA4
TV	TokenVault.sol	5CB5EF9D4ABCAB491CDDA8737020231C0029F667
BRT	BridgesRouter.sol	D2E911EDD0EBFCCFC4A060B39DEF675FCE595BF8
BRF	BridgesRef.sol	326435A36C632A2DCE26B2DCCF0C9F42DEB7676A
FACT	BridgesFactory.sol	7F784FE5EB937411DEB10C773F1A6D7481E508E9
PAIR	BridgesPair.sol	378376FA2640038FA241383D9F509737D01D8399

# Audit Scope & Methodology

## Methodology

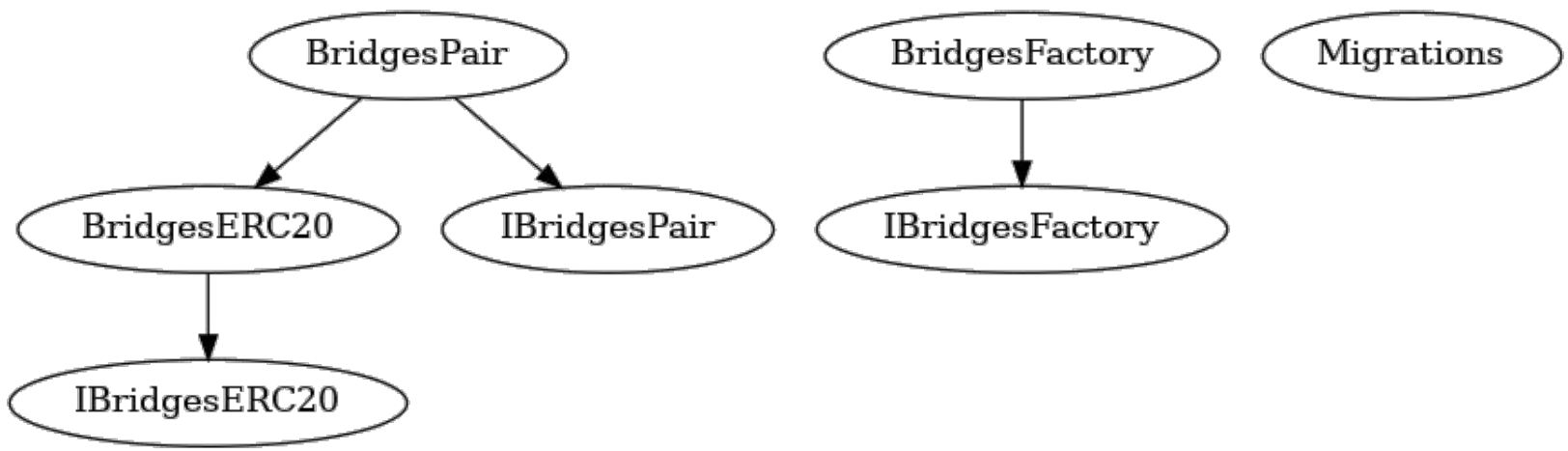
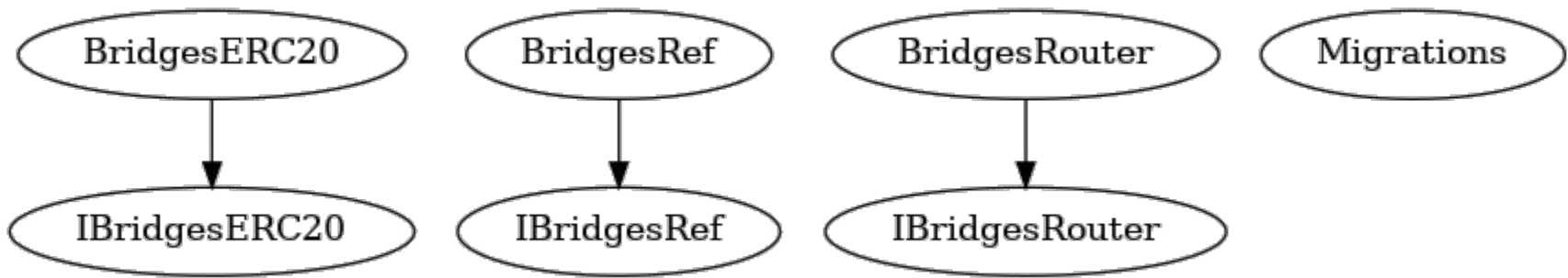
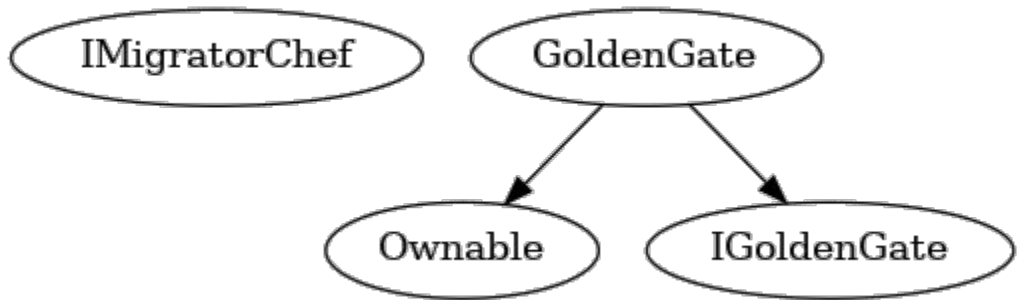
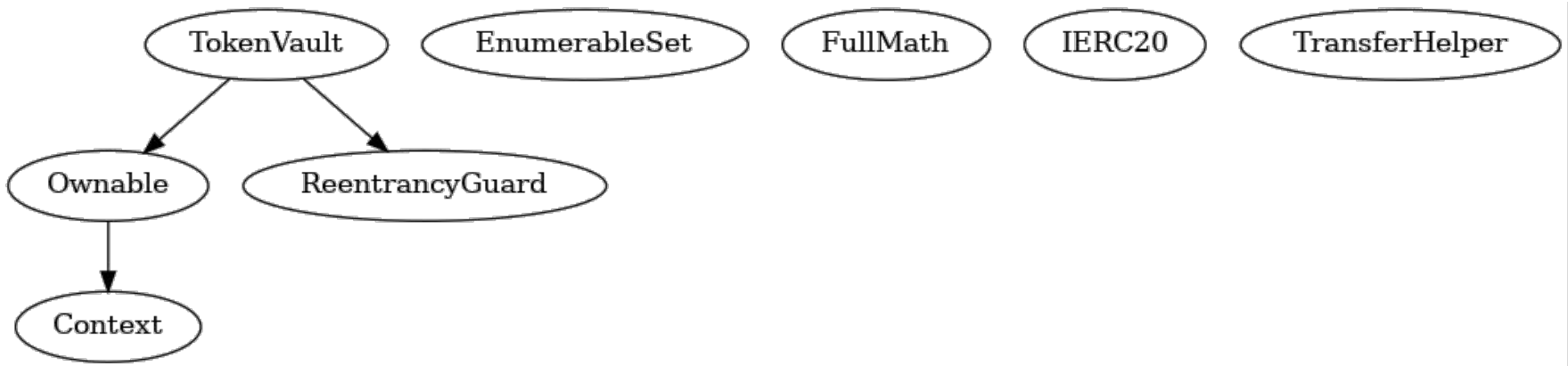
The auditing process pays special attention to the following considerations:

- 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
<u>GLOBAL-1</u>	Centralization Risk	Centralization / Privilege	 Medium	Partially Resolved
<u>GG-1</u>	DoS Dividends	Denial-of-Service	 Medium	Resolved
<u>GG-2</u>	Steal All BNB	Reentrancy	 High	Resolved
<u>GG-3</u>	Shorten Lock	Logical Error	 Medium	Resolved
<u>GG-4</u>	DoS Deposit and Withdraw	Denial-of-Service	 Medium	Resolved
<u>GG-5</u>	Dividend Sniping	Frontrunning	 Medium	Acknowledged
<u>GG-6</u>	Redundant Boolean Check	Optimization	 Low	Resolved
<u>GG-7</u>	Typo	Typo	 Low	Resolved
<u>GG-8</u>	Duplicate Code Lines	Optimization	 Low	Resolved
<u>GG-9</u>	Superfluous Code	Optimization	 Low	Resolved
<u>GG-10</u>	Cannot Withdraw Max Amount	Logical Error	 Low	Resolved
<u>GG-11</u>	Superfluous Code	Optimization	 Low	Resolved
<u>GG-12</u>	Lack of camelCase	Code Cleanliness	 Low	Resolved



# Findings & Resolutions

ID	Title	Category	Severity	Status
<u>TV-1</u>	Setting Default Values	Optimization	<div><div></div></div> Low	Resolved
<u>BRT-1</u>	Unable to Swap ETH	Logical Error	<div><div></div></div> Medium	Resolved
<u>BRF-1</u>	Accidental Magnification	Logical Error	<div><div></div></div> Medium	Resolved
<u>BRF-2</u>	Unbounded disRate	Privilege / Logical Error	<div><div></div></div> Medium	Resolved
<u>BRF-3</u>	Superfluous Code	Optimization	<div><div></div></div> Low	Resolved
<u>BRF-4</u>	Typo	Typo	<div><div></div></div> Low	Resolved
<u>BRF-5</u>	Cannot Withdraw Max Amount	Logical Error	<div><div></div></div> Low	Resolved
<u>FACT-1</u>	Unimplemented Interface Methods	Logical Error	<div><div></div></div> Low	Resolved
<u>PAIR-1</u>	Duplicate Dividends	Logical Error	<div><div></div></div> High	Resolved
<u>PAIR-2</u>	DoS Dividends	Denial-of-Service	<div><div></div></div> Medium	Resolved

# GLOBAL-1 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Medium	Global	Partially Resolved

## Description

Privileged addresses have authority over many functions that may be used to negatively disrupt the project. Some important privileges include:

### GoldenGate

- owner can withdraw all funds.
- owner can set admins which are able to dilute allocation of other pools.
- owner can set the migrator contract which can lead to loss of LP if malicious.

### TokenVesting

- owner can arbitrarily set the fee and fee address which can lead to loss of user funds.

### BridgesRef

- feeToSetter can arbitrarily set the distribution rate.
- feeToSetter can withdraw any ERC-20 token in the contract.

### BridgesRouter

- feeSetter can set a arbitrary referral and dividend tracker contract.

### BridgesFactory

- feeToSetter can set an arbitrary start time for when a pair can be tradeable.

## Recommendation

Ensure that the privileged addresses are multi-sig and/or introduce timelock for improved community oversight. Optionally introduce require statements to limit the scope of the exploits that can be carried out by the privileged addresses.

# GLOBAL-1 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Medium	Global	Partially Resolved

## Resolution

Bridges Team:

- All centralized BNB and token withdrawal functions have been removed from the GoldenGate contract,
- The possibility to change fees on the TokenVault has been removed as well.
- The GoldenGate migrator has to be there for an eventual V2 in the future.
- For the same reason all update functions on the tracker are necessary.
- Every privileged address will be a multi-sig with trusted members in production.

# GG-1 | DoS Dividends

Category	Severity	Location	Status
Denial-of-Service	● Medium	GoldenGate.sol: 181	Resolved

## Description

Due to the unbounded `for` loop in `distributeDividends`, there is a risk of a DoS attack. Anytime a new address deposits to pool 0, they are added to the `usersBridges` list. A malicious party can keep generating new addresses and deposit miniscule amounts of LP to make `distributeDividends` exceed the block gas limit, stopping all dividends.

## Recommendation

Process the users in smaller batches, set a cap on number of users who can receive dividends, or modify the dividend allocation logic entirely such that a `for` loop is not needed.

For an alternative approach, see this “pointsPerShare” implementation:  
<https://github.com/indexed-finance/dividends/tree/master/contracts>

## Resolution

Bridges Team:

- The dividend distribution mechanism was updated to a dividendsPerShare model.

# GG-2 | Reenter Dividends

Category	Severity	Location	Status
Reentrancy	● High	GoldenGate.sol: 228, 258	Resolved

## Description

Because the dividends paid to a user is only updated after the external call sending them funds, it is possible for a malicious contract to re-enter and keep draining `div` amount of BNB on each call.

## Recommendation

Add a `nonReentrant` modifier from OpenZeppelin's ReentrancyGuard or utilize the check-effects-interactions pattern.

## Resolution

Bridges Team:

- Added the `lock` modifier to `deposit`, `depositLocked`, `relock`, and `withdraw`.

# GG-3 | Shorten Lock

Category	Severity	Location	Status
Logical Error	● Medium	GoldenGate.sol: 310	Resolved

## Description

In the transferLock function `_to.stakeUntil` is set to `_from.stakeUntil`. Therefore, it is possible to shorten the lock period by transferring from an address with a shorter lock to an address with a longer lock.

## Recommendation

When transferring a lock, adopt a push then pull pattern where the receiver needs to accept an incoming lock, and then adopt the longer lock period when combining locks. Alternatively, make each lock its own unique NFT token.

## Resolution

Bridges Team:

- Now requires that the `_to.amount = 0`.

# GG-4 | DoS Deposit and Withdraw

Category	Severity	Location	Status
Denial-of-Service	● Medium	GoldenGate.sol: 228, 258	Resolved

## Description

Because depositing and withdrawing from pool 0 relies on a successful BNB transfer for the dividends payment, it is possible to prevent deposits and withdrawals. If a user were to drain the BNB from the contract using the re-entrancy described earlier or the owner drained the BNB using the BNB function, then the call would fail and the transaction would revert.

## Recommendation

Refactor the dividend payments so they are separate from withdrawals and deposits.

## Resolution

Bridges Team:

- Dividends have been refactored and the emergency BNB function has been removed.

# GG-5 | Dividend Sniping

Category	Severity	Location	Status
Frontrunning	● Medium	GoldenGate.sol	Acknowledged

## Description

Because it is possible to publicly see transactions that are sending value to `distributeDividends`, bots can frontrun the distribution. This way addresses may sandwich a deposit and withdrawal around a distribution in order to unfairly accumulate dividends while never effectively holding the token.

## Recommendation

Introduce a warmup period, or require locking for dividends.

## Resolution

Bridges Team:

- We think this is unlikely as it would require swapping for tokens and providing/removing liquidity to achieve a return, it would likely be gas/slippage cost prohibitive.



# GG-6 | Redundant Boolean Check

Category	Severity	Location	Status
Optimization	● Low	GoldenGate.sol: 208	Resolved

## Description

The check `user.alreadyHere == false` can be simplified to `!user.alreadyHere`.

## Recommendation

Replace `user.alreadyHere == false` with `!user.alreadyHere`.

## Resolution

Bridges Team:

- The simplification was made.

# GG-7 | Typo

Category	Severity	Location	Status
Typo	● Low	GoldenGate.sol: 265	Resolved

## Description

Withdraw is spelled **witdhraw** in the error message on line 265.

## Recommendation

Correct it to **withdraw**.

## Resolution

Bridges Team:

- Typo has been fixed.

# GG-8 | Duplicate Code Lines

Category	Severity	Location	Status
Optimization	● Low	GoldenGate.sol: 273	Resolved

## Description

The statement `user.rewardDebt = user.amount.mul(pool.accBRGPerShare).div(1e12)` is repeated on line 275.

## Recommendation

Remove the first occurrence on line 273.

## Resolution

Bridges Team:

- The first duplicate was removed.

# GG-9 | Superfluous Code

Category	Severity	Location	Status
Optimization	● Low	GoldenGate.sol: 288	Resolved

## Description

In the emergencyWithdraw function, the statement `user.userLockedAmount = user.userLockedAmount.sub(user.userLockedAmount)` is equivalent to `user.userLockedAmount = 0`.

## Recommendation

Replace the inefficient statement with `user.userLockedAmount = 0`.

## Resolution

Bridges Team:

- Updated to `= 0`.

# GG-10 | Cannot Withdraw Max Amount

Category	Severity	Location	Status
Logical Error	● Low	GoldenGate.sol: 330, 336	Resolved

## Description

In the Bep20 and BNB functions the require statements specify a value < the contract balance, meanwhile it may be intended to remove a value equal to the contract balance.

## Recommendation

Confirm whether or not the exact contract balance should be able to be withdrawn and optionally update accordingly.

## Resolution

Bridges Team:

- Updated to <=

# GG-11 | Superfluous Code

Category	Severity	Location	Status
Optimization	● Low	GoldenGate.sol: 330, 336	Resolved

## Description

In the Bep20 function the payable cast on line 331 is unnecessary as the safeTransfer function simply accepts an address.

## Recommendation

Remove the payable cast.

## Resolution

Bridges Team:

- Removed the payable cast.

# GG-12 | Lack of camelCase

Category	Severity	Location	Status
Code Cleanliness	● Low	GoldenGate.sol: 347	Resolved

## Description

The function pendingbridges does not abide by camelCase naming conventions.

## Recommendation

Rename it to pendingBridges.

## Resolution

Bridges Team:

- The function has been renamed.

# TV-1 | Setting Default Values

Category	Severity	Location	Status
Optimization	● Low	TokenVault.sol: 82	Resolved

## Description

The `shares` variable is initialized to 0. This is unnecessary because the default value for the `uint256` type is 0.

## Recommendation

Remove the assignment.

## Resolution

Bridges Team:

- Removed the assignment.



# BRT-1 | Unable to Swap ETH

Category	Severity	Location	Status
Logical Error	● Medium	BridgesRouter.sol: 168	Resolved

## Description

Because the trading fee `tradingFee` is taken before `amounts` is calculated, there may not be enough BNB to deposit into the WBNB contract. Thus, the transaction will revert and the swap will fail.

## Recommendation

Take the trading fee after the swap has occurred, or account for the trading fee in `getAmountsIn`.

## Resolution

Bridges Team:

- Added amounts adjustment for the fee.

# BRF-1 | Accidental Magnification

Category	Severity	Location	Status
Logical Error	● Medium	BridgesRef.sol: 131	Resolved

## Description

In the `distribute` function each `user.earned` amount is multiplied by the `disRate` but in the `withdraw` function, the `user.earned` amount is not divided by some divisor that corresponds to the `disRate`.

## Recommendation

Add a divisor for the `disRate` and use it to adjust `user.earned` values in either the `distribute` or `withdraw` functions as needed.

## Resolution

Bridges Team:

- This is intended behavior, we treat the `disRate` as a multiplier.

# BRF-2 | Unbounded disRate

Category	Severity	Location	Status
Privilege / Logical Error	● Medium	BridgesRef.sol: 127	Resolved

## Description

There is no bound to how high the `disRate` variable can be set in the `setDisRate` function. Therefore, the distributed amount in the `distribute` function may exceed the provided `amount`.

## Recommendation

Implement a maximum cap for `disRate`. A cap of ~ 57% (1/1.75) would allow for a maximum distribution of `amount`.

## Resolution

Bridges Team:

- Set a `disRate` cap of 1000.

# BRF-3 | Superfluous Code

Category	Severity	Location	Status
Optimization	● Low	BridgesRef.sol: 26	Resolved

## Description

The Distributed variable is not referenced at all.

## Recommendation

Remove the Distributed variable.

## Resolution

Bridges Team:

- The variable has been removed.

# BRF-4 | Typo

Category	Severity	Location	Status
Typo	<div><div></div>Low</div>	BridgesRef.sol: 63, 67, 72	Resolved

## Description

The `withelistTokens`, `withelistUsers`, and `withelistUser` functions all have a typo.

## Recommendation

Correct them to `whitelistTokens`, `whitelistUsers`, and `whitelistUser`.

## Resolution

Bridges Team:

- Typos have been corrected.

# BRF-5 | Cannot Withdraw Max Amount

Category	Severity	Location	Status
Logical Error	● Low	BridgesRef.sol: 156	Resolved

## Description

In the `withdraw` function the `require` statement specifies a value `<` the contract balance, meanwhile it may be intended to remove a value equal to the contract balance.

## Recommendation

Confirm whether or not the exact contract balance should be able to be withdrawn and optionally update accordingly.

## Resolution

Bridges Team:

- Updated to `<=` to be able to withdraw the max amount.

# FACT-1 | Unimplemented Interface Methods

Category	Severity	Location	Status
Logical Error	● Low	BridgesFactory.sol	Resolved

## Description

The BridgesFactory contract fails to provide implementations for the feeTo, tradingStart, and setFeeTo functions defined in the IBridgesFactory interface.

## Recommendation

Either add implementations for these functions in the BridgesFactory contract or remove them from the IBridgesFactory interface.

## Resolution

Bridges Team:

- Unimplemented functions have been removed from the interface.

# PAIR-1 | Duplicate Dividends

Category	Severity	Location	Status
Logical Error	● High	BridgesPair.sol: 124	Resolved

## Description

In the `mint` function, the only precondition for adding an address to the `users` list is if the balance of the address is 0. Additionally, an address is not removed from the `users` list if it transfers it's balance of the BridgesPair token.

This way an address can continually `mint` and transfer/burn it's tokens to enter the `users` list multiple times. Multiple entries in the `users` list will result in multiple dividends being paid out in the `distributeDividends` function.

## Recommendation

Add a check for addresses that are already in the `users` list.

## Resolution

Bridges Team:

- Dividends have been refactored to a `dividendsPerShare` implementation.



# PAIR-2 | DoS Dividends

Category	Severity	Location	Status
Denial-of-Service	● Medium	BridgesPair.sol: 198	Resolved

## Description

Due to the unbounded `for` loop in `distributeDividends`, there is a risk of a DoS attack. A malicious party can keep generating new addresses and minting minimal amounts of the BridgesPair token to make `distributeDividends` exceed the block gas limit, stopping all dividends.

## Recommendation

Process the users in smaller batches, set a cap on number of users who can receive dividends, or modify the dividend allocation logic entirely such that a `for` loop is not needed.

For an alternative approach, see this “pointsPerShare” implementation:  
<https://github.com/indexed-finance/dividends/tree/master/contracts>

## Resolution

Bridges Team:

- Dividends have been refactored to a `dividendsPerShare` implementation.

# Auditor's Verdict

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

- Bridges' smart contracts have a **MEDIUM RISK SEVERITY**
- Bridges' smart contracts have an **ACTIVE OWNERSHIP**
- Bridges' smart contract owner has multiple "write" privileges. Centralization risk correlated to the active ownership is **MEDIUM**

# 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>