#Hashlock.

Dione (Bridge)

SMART CONTRACT

Security Audit

Performed on Contracts:

Github Commit Hash:e97d45285416793212b778cd405828787586486b

Platform

BSC / Odyssey

hashlock.com.au

DECEMBER 2023

Table of Contents

Executive Summary	4
Project Context	4
Audit scope	7
Security Rating	8
Standardised Checks	9
Intended Smart Contract Functions	11
Code Quality	12
Audit Resources	12
Dependencies	12
Severity Definitions	13
Audit Findings	14
Centralisation	15
Conclusion	16
Our Methodology	17
Disclaimers	19
About Hashlock	20

CAUTION

THIS DOCUMENT IS A SECURITY AUDIT REPORT AND MAY CONTAIN CONFIDENTIAL INFORMATION. THIS INCLUDES IDENTIFIED VULNERABILITIES AND MALICIOUS CODE WHICH COULD BE USED TO COMPROMISE THE PROJECT. THIS DOCUMENT SHOULD ONLY BE FOR INTERNAL USE UNTIL ISSUES ARE RESOLVED. ONCE VULNERABILITIES ARE REMEDIATED, THIS REPORT CAN BE MADE PUBLIC. THE CONTENT OF THIS REPORT IS OWNED BY HASHLOCK PTY LTD FOR USE OF THE CLIENT.

Executive Summary

The Dione team partnered with Hashlock to conduct a security audit of their Bridge smart contracts and Golang Infrastructure. Hashlock manually and proactively reviewed the code in order to ensure the project's team and community that the deployed contracts are secure.

Project Context

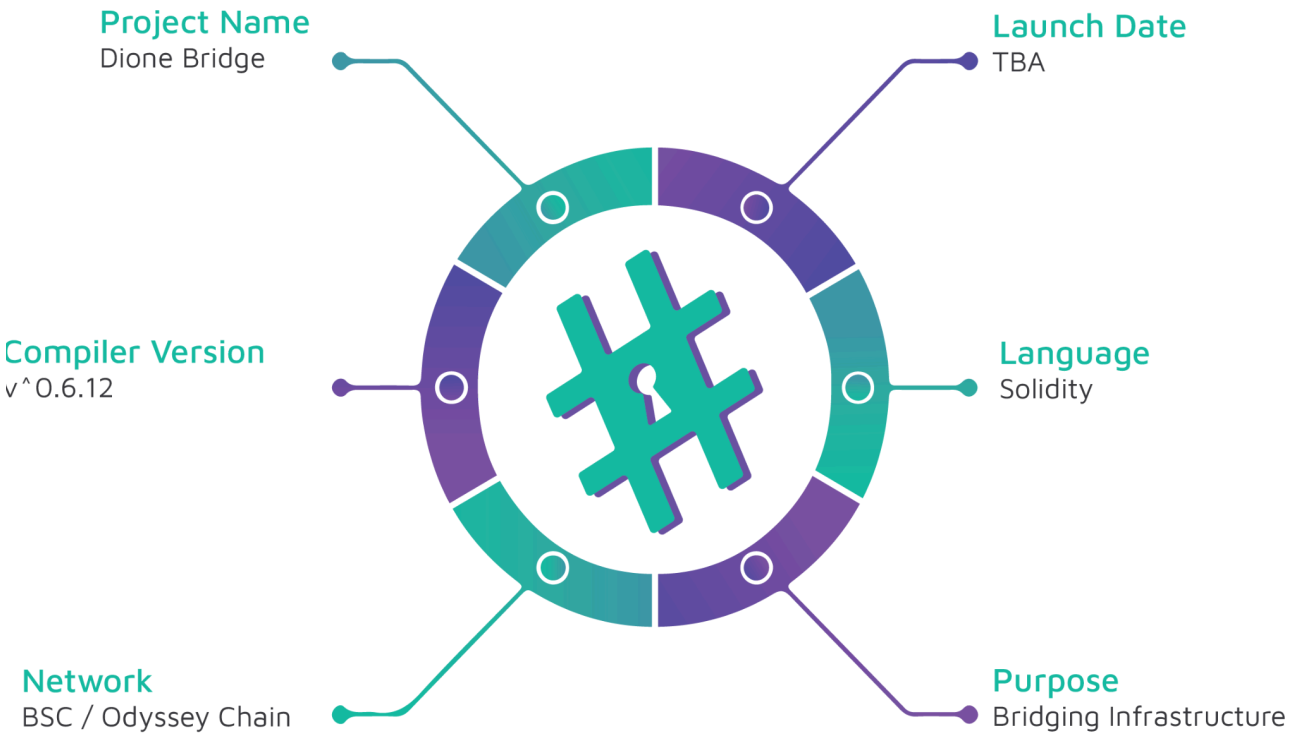
Dione is a revolutionary L1 blockchain initiative in development, enabling renewable energy trade, created by the best blockchain & development minds in crypto.

Project Name: Dione
Compiler Version: ^0.6.12
Website: https://www.dioneprotocol.com/

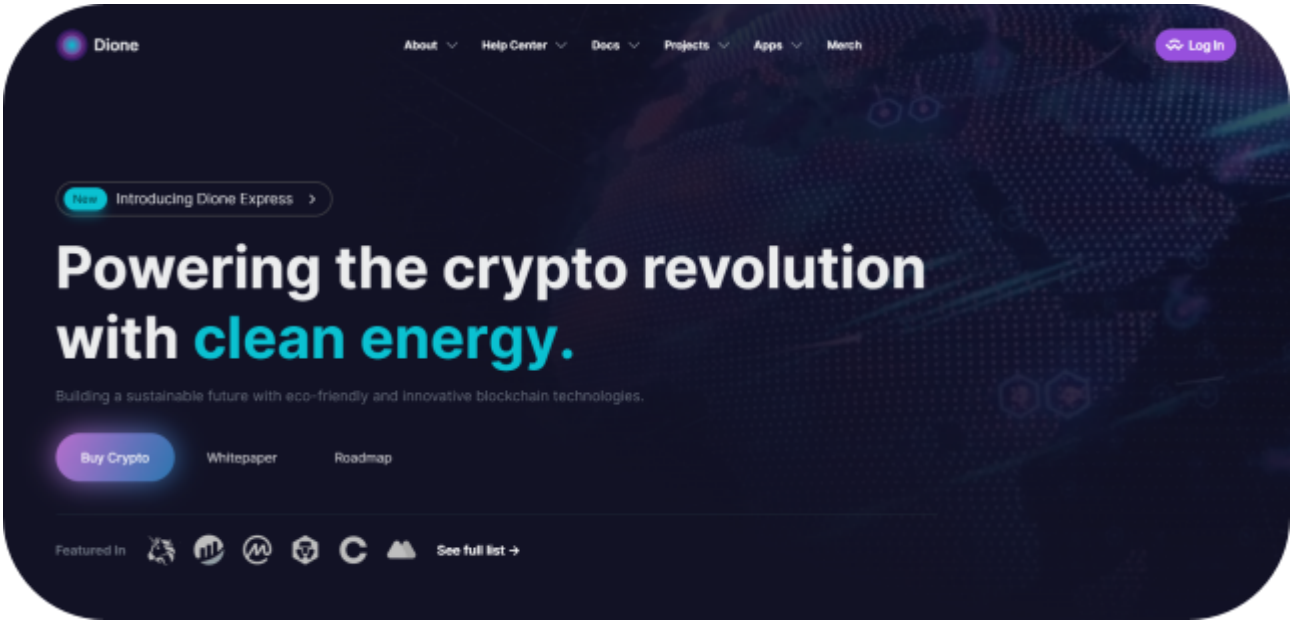
Logo:

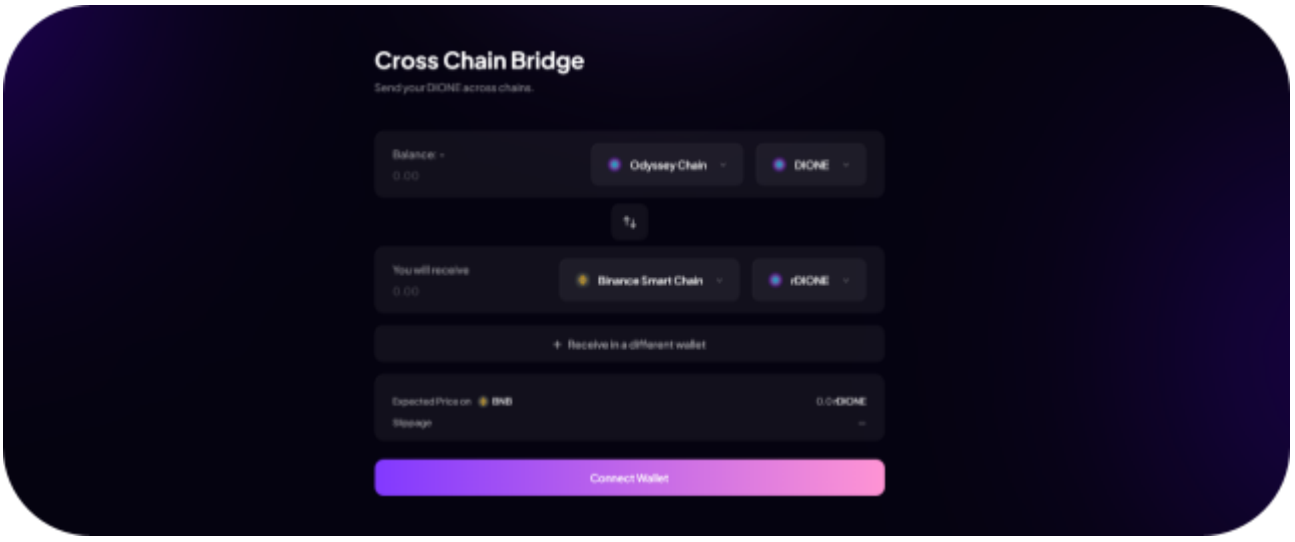
DIONE

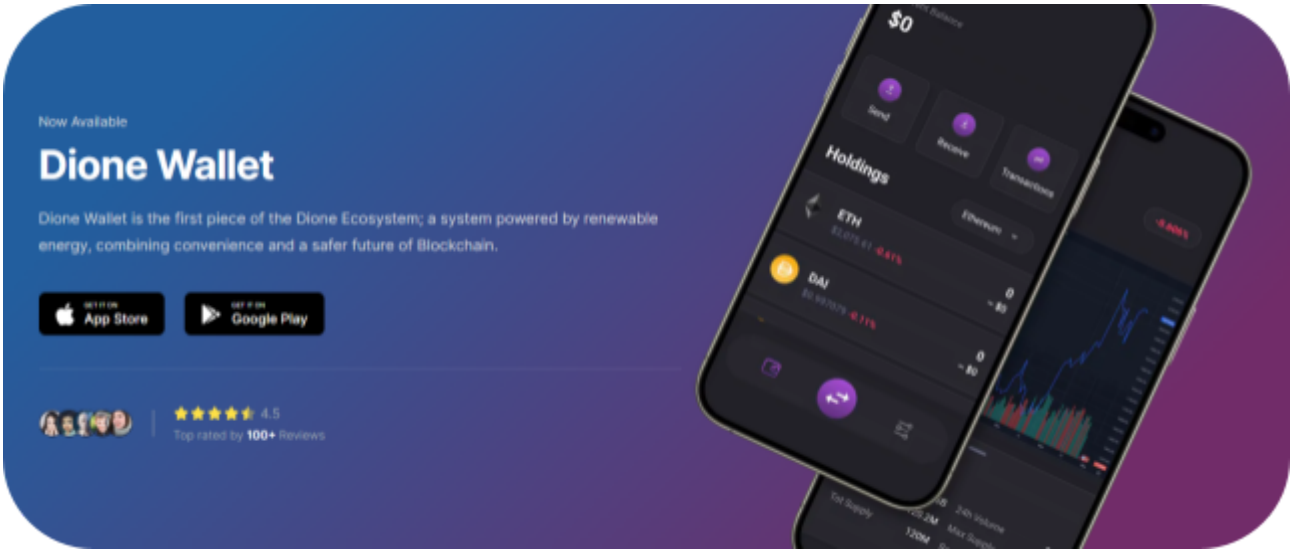
Visualised Context:



Project Visuals:







Audit scope

We at Hashlock audited the solidity & Golang code within the Diones Bridge, the scope of works included a comprehensive review of the code listed below. We tested the infrastructure to check for their security and efficiency. These tests were undertaken primarily through manual line by line analysis and were supported by software assisted testing.

Description	**Project Review and Security Analysis Report for Dione Protocol Smart Contracts and other factors.**
Platform	**Ethereum / Solidity / GoLang**
Audit Date	**Dec, 2023**
AmplificationUtils.sol	**e97d45285416793212b778cd4058287875864 86b**
LPToken.sol	**e97d45285416793212b778cd4058287875864 86b**
MathUtils.sol	**e97d45285416793212b778cd4058287875864 86b**
Swap.sol	**e97d45285416793212b778cd4058287875864 86b**
SwapDeployer.sol	**e97d45285416793212b778cd4058287875864 86b**
SwapEthWrapper.sol	**e97d45285416793212b778cd4058287875864 86b**
SwapFlashLoan.sol	**e97d45285416793212b778cd4058287875864 86b**
SwapUtils.sol	**e97d45285416793212b778cd4058287875864 86b**
BaseSwapDeposit.sol	**e97d45285416793212b778cd4058287875864 86b**
GenericERC20.sol	**e97d45285416793212b778cd4058287875864 86b**
Multicall2.sol	**e97d45285416793212b778cd4058287875864 86b**
BridgeConfig.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridge.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeERC20.sol	**e97d45285416793212b778cd4058287875864 86b**

DioneBridgeERC20DeterministicFactory.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeERC20Factory.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeERC677.sol	**e97d45285416793212b778cd4058287875864 86b**
ECDSAFactory.sol	**e97d45285416793212b778cd4058287875864 86b**
ECDSANodeManagement.sol	**e97d45285416793212b778cd4058287875864 86b**
ERC20Migrator.sol	**e97d45285416793212b778cd4058287875864 86b**
PoolConfig.sol	**e97d45285416793212b778cd4058287875864 86b**
BridgeStructs.sol	**e97d45285416793212b778cd4058287875864 86b**
SignedSafeMath.sol	**e97d45285416793212b778cd4058287875864 86b**
UniversalToken.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeAdapter.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeRouter.sol	**e97d45285416793212b778cd4058287875864 86b**
LocalBridgeConfig.sol	**e97d45285416793212b778cd4058287875864 86b**
SwapCalculator.sol	**e97d45285416793212b778cd4058287875864 86b**
SwapQuoter.sol	**e97d45285416793212b778cd4058287875864 86b**
AddressArrayUtils.sol	**e97d45285416793212b778cd4058287875864 86b**
BridgeConfigV3Lens.sol	**e97d45285416793212b778cd4058287875864 86b**
Disperse.sol	**e97d45285416793212b778cd4058287875864 86b**
EnumerableStringMap.sol	**e97d45285416793212b778cd4058287875864 86b**
MathUtils.sol	**e97d45285416793212b778cd4058287875864 86b**

TimelockController.sol	**e97d45285416793212b778cd4058287875864 86b**
L1BridgeZap.sol	**e97d45285416793212b778cd4058287875864 86b**
L2BridgeZap.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeCCTPEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeCCTPFeesEvents .sol	**e97d45285416793212b778cd4058287875864 86b**
MessageTransmitterEvents.s ol	**e97d45285416793212b778cd4058287875864 86b**
TokenMessengerEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeCCTPFees.sol	**e97d45285416793212b778cd4058287875864 86b**
Errors.sol	**e97d45285416793212b778cd4058287875864 86b**
MinimalForwarder.sol	**e97d45285416793212b778cd4058287875864 86b**
Request.sol	**e97d45285416793212b778cd4058287875864 86b**
RouterErrors.sol	**e97d45285416793212b778cd4058287875864 86b**
TypeCasts.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeCCTP.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeCCTPRouter.sol	**e97d45285416793212b778cd4058287875864 86b**
PrivateFactory.sol	**e97d45285416793212b778cd4058287875864 86b**
PrivatePool.sol	**e97d45285416793212b778cd4058287875864 86b**
AgentSecured.sol	**e97d45285416793212b778cd4058287875864 86b**
MessagingBase.sol	**e97d45285416793212b778cd4058287875864 86b**
MultiCallable.sol	**e97d45285416793212b778cd4058287875864 86b**

Version.sol	**e97d45285416793212b778cd4058287875864 86b**
BaseClient.sol	**e97d45285416793212b778cd4058287875864 86b**
MessageRecipient.sol	**e97d45285416793212b778cd4058287875864 86b**
PingPongClient.sol	**e97d45285416793212b778cd4058287875864 86b**
TestClient.sol	**e97d45285416793212b778cd4058287875864 86b**
AgentManagerEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
DestinationEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
ExecutionHubEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
GasOracleEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
InboxEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
OriginEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
SnapshotHubEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
StateHubEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
StatementInboxEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
SummitEvents.sol	**e97d45285416793212b778cd4058287875864 86b**
ExecutionHub.sol	**e97d45285416793212b778cd4058287875864 86b**
SnapshotHub.sol	**e97d45285416793212b778cd4058287875864 86b**
StateHub.sol	**e97d45285416793212b778cd4058287875864 86b**
Inbox.sol	**e97d45285416793212b778cd4058287875864 86b**
LightInbox.sol	**e97d45285416793212b778cd4058287875864 86b**
StatementInbox.sol	**e97d45285416793212b778cd4058287875864 86b**
Constants.sol	**e97d45285416793212b778cd4058287875864 86b**

Structures.sol	**e97d45285416793212b778cd4058287875864 86b**
Attestation.sol	**e97d45285416793212b778cd4058287875864 86b**
BaseMessage.sol	**e97d45285416793212b778cd4058287875864 86b**
ByteString.sol	**e97d45285416793212b778cd4058287875864 86b**
MemView.sol	**e97d45285416793212b778cd4058287875864 86b**
Message.sol	**e97d45285416793212b778cd4058287875864 86b**
Receipt.sol	**e97d45285416793212b778cd4058287875864 86b**
Snapshot.sol	**e97d45285416793212b778cd4058287875864 86b**
State.sol	**e97d45285416793212b778cd4058287875864 86b**
MerkleMath.sol	**e97d45285416793212b778cd4058287875864 86b**
MerkleTree.sol	**e97d45285416793212b778cd4058287875864 86b**
GasData.sol	**e97d45285416793212b778cd4058287875864 86b**
Header.sol	**e97d45285416793212b778cd4058287875864 86b**
Number.sol	**e97d45285416793212b778cd4058287875864 86b**
Request.sol	**e97d45285416793212b778cd4058287875864 86b**
Tips.sol	**e97d45285416793212b778cd4058287875864 86b**
AgentManager.sol	**e97d45285416793212b778cd4058287875864 86b**
BondingManager.sol	**e97d45285416793212b778cd4058287875864 86b**
LightManager.sol	**e97d45285416793212b778cd4058287875864 86b**
AuthVerifier.sol	**e97d45285416793212b778cd4058287875864 86b**
GasFeePricing.sol	**e97d45285416793212b778cd4058287875864 86b**
MessageBus.sol	**e97d45285416793212b778cd4058287875864 86b**

MessageBusReceiver.sol	**e97d45285416793212b778cd4058287875864 86b**
MessageBusSender.sol	**e97d45285416793212b778cd4058287875864 86b**
BatchMessageSender.sol	**e97d45285416793212b778cd4058287875864 86b**
PingPong.sol	**e97d45285416793212b778cd4058287875864 86b**
HeroCoreUpgradeable.sol	**e97d45285416793212b778cd4058287875864 86b**
StatScienceUpgradeable.sol	**e97d45285416793212b778cd4058287875864 86b**
AssistingAuctionUpgradeable.sol	**e97d45285416793212b778cd4058287875864 86b**
ERC721AuctionBaseUpgradeable.sol	**e97d45285416793212b778cd4058287875864 86b**
HeroAuctionUpgradeable.sol	**e97d45285416793212b778cd4058287875864 86b**
CrystalFeesUpgradeable.sol	**e97d45285416793212b778cd4058287875864 86b**
AuctionTypes.sol	**e97d45285416793212b778cd4058287875864 86b**
HeroBridgeUpgradeable.sol	**e97d45285416793212b778cd4058287875864 86b**
PetBridgeUpgradeable.sol	**e97d45285416793212b778cd4058287875864 86b**
TearBridge.sol	**e97d45285416793212b778cd4058287875864 86b**
GaiaTears.sol	**e97d45285416793212b778cd4058287875864 86b**
InventoryItem.sol	**e97d45285416793212b778cd4058287875864 86b**
LibGeneScience.sol	**e97d45285416793212b778cd4058287875864 86b**
RandomGenerator.sol	**e97d45285416793212b778cd4058287875864 86b**
CrystalTypes.sol	**e97d45285416793212b778cd4058287875864 86b**
HeroTypes.sol	**e97d45285416793212b778cd4058287875864 86b**

JobTiers.sol	**e97d45285416793212b778cd4058287875864 86b**
PetTypes.sol	**e97d45285416793212b778cd4058287875864 86b**
RandomTypes.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeMessagingReceiver.sol	**e97d45285416793212b778cd4058287875864 86b**
DioneBridgeMessagingReceiverUpgradeable.sol	**e97d45285416793212b778cd4058287875864 86b**
DefaultRouter.sol	**e97d45285416793212b778cd4058287875864 86b**
LinkedPool.sol	**e97d45285416793212b778cd4058287875864 86b**
DefaultAdapter.sol	**e97d45285416793212b778cd4058287875864 86b**
OnlyDelegateCall.sol	**e97d45285416793212b778cd4058287875864 86b**
CurveV1Module.sol	**e97d45285416793212b778cd4058287875864 86b**
DssPsmModule.sol	**e97d45285416793212b778cd4058287875864 86b**
GMXV1Module.sol	**e97d45285416793212b778cd4058287875864 86b**
GMXV1StableArbitrumModule.sol	**e97d45285416793212b778cd4058287875864 86b**
UniswapV3Module.sol	**e97d45285416793212b778cd4058287875864 86b**
VelodromeV2Module.sol	**e97d45285416793212b778cd4058287875864 86b**
DefaultPoolCalc.sol	**e97d45285416793212b778cd4058287875864 86b**
PoolQuoterV1.sol	**e97d45285416793212b778cd4058287875864 86b**
SwapQuoterV2.sol	**e97d45285416793212b778cd4058287875864 86b**
TokenTree.sol	**e97d45285416793212b778cd4058287875864 86b**
Factory.sol	**e97d45285416793212b778cd4058287875864 86b**

Migrations.sol	**e97d45285416793212b778cd4058287875864 86b**
MultiSigWallet.sol	**e97d45285416793212b778cd4058287875864 86b**
MultiSigWalletFactory.sol	**e97d45285416793212b778cd4058287875864 86b**
MultiSigWalletWithDailyLimit.sol	**e97d45285416793212b778cd4058287875864 86b**
MultiSigWalletWithDailyLimitFactory.sol	**e97d45285416793212b778cd4058287875864 86b**

Security Rating

After Hashlock's Audit, we found the smart contracts to be **"Secure"**. The contracts all follow simple logic, with correct and detailed ordering. They use a series of interfaces, and the protocol uses a list of Open Zeppelin contracts. We initially identified some significant vulnerabilities that have since been addressed.

The 'Hashlocked' rating is reserved for projects that ensure ongoing security via bug bounty programs or on chain monitoring technology.

All issues uncovered during automated and manual analysis were meticulously reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in the Function list section and all identified issues can be found in the Audit overview section.

All vulnerabilities initially identified have now been resolved and acknowledged.

Hashlock found:

7 High severity vulnerabilities

7 Medium severity vulnerabilities

Caution: *Hashlock's audits do not guarantee a project's success or ethics, and are not liable or responsible for security. Always conduct independent research about any project before interacting.*

Intended Smart Contract Purpose

Claimed Behaviour	Actual Behaviour
DioneBridge.sol Admin functions: Privileged roles in this contract are able to mint bridge tokens unconstrained. Other Specifications: This contract receives and mints tokens on behalf of the bridge.	**Contract achieves this functionality**
DioneBridgeERC20.sol Admin functions: This contract has a minter role which is given to an arbitrary number of addresses. This contract is upgradable by the owner and subject to change at any time. Other Specifications: This contract is upgradable ERC20 contract.	**Contract achieves this functionality**
DioneBridgeERC20DeterministicFactory.sol Admin functions: This is an ownable contract controlled by one address. Other Specifications: Can be used to deploy tokens to a predetermined address.	**Contract achieves this functionality**

DioneBridgeERC20Factory.sol Other Specifications: Factory contract that can deploy any number of DioneBridgeERC20 contracts.	**Contract achieves this functionality**
DioneBridgeERC677.sol Admin functions: This contract inherits from DioneBridgeERC20 and therefore has the minter role which is given to an arbitrary number of addresses. This contract is upgradable by the sole owner and subject to change at any time	**Contract achieves this functionality**

Code Quality

This audit scope involves the solidity smart contracts and Golang of the Dione bridge, as outlined in the Audit Scope section. All contracts, libraries and interfaces mostly follow standard best practices and to help avoid unnecessary complexity that increases the likelihood of exploitation, however some refactoring is required.

The code is very well commented and closely follows best practice nat-spec styling. All comments are correctly aligned with code functionality.

Audit Resources

We were given the Dione Protocol's bridge code in the form of Github access.

As mentioned above, code parts are well commented. The logic is straightforward, and therefore it is easy to quickly comprehend the programming flow as well as the complex code logic. The comments are helpful in understanding the overall architecture of the protocol.

Dependencies

As per our observation, the libraries used in this smart contracts infrastructure are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

Severity Definitions

Significance	Description
High	High severity vulnerabilities can result in loss of funds, asset loss, access denial, and other critical issues that will result in the direct loss of funds and control by the owners and community.
Medium	Medium level difficulties should be solved before deployment, but won't result in loss of funds.
Low	Low level vulnerabilities are areas that lack best practices that may cause small complications in the future.
Gas	Gas Optimisations, issues and inefficiencies

Audit Findings

High

[H-01] x * y = k / x * y * z = k invariant is violated leading liquidity providers to lose funds

Description

The AMM is used to allow users to swap tokens and provide liquidity to earn fees on the tokens they've provided. The invariant to these liquidity pools are x * y = k for two tokens and x * y * z = k for three tokens where x , y and z are denoted as the quantity of Token A and Token B (and Token C if three tokens are being used) and k is the constant value.

For example, Alice deposits 30 * 1e18 Eth and 66618 * 1e6 USDC into the pool to set the price of WETH to an accurate price of approximately $2,220.60 USDC per WETH token.

Bob comes along and has never used DEFI before but wants to make money on the 100 WETH tokens equal to approximately $222,060 USD. He chooses to deposit 100 * 1e18 Eth and 1 * 1e6 USDC into the liquidity pool. He wishes to withdraw immediately but he receives exactly 48.968198178208176788 WETH tokens in return equal to about $108,809.4 USD and 25093.941495 USDC equal to about $25,093 USD. As a result he managed to claim back ~$133,902.4 back in value which is an immediate loss of approximately ~$88,157.6 in value without facing any realistic risk.

Impact

Users will immediately lose funds should they choose to withdraw from the protocol as an LP Provider.

Recommendations

It's recommended that the x * y = k / x * y * z = k is enforced when users provide and remove liquidity. The contract should revert if the liquidity provided is disproportional and the swapPool should return a proportionate amount back to the user.

Status

Resolved - The AMM has been removed from the project.

[H-02] Incorrect price calculation may cause users to receive less funds than expected

Description

The AMM is designed to facilitate swaps between the liquidity pool and the user initiating a swap. There exists an issue where a user is sometimes underpaid when initiating a swap. For example, bob swaps 1 Eth and receives 6 USDC (If _a == 5**10, 1042 USDC if _a == 1 otherwise) in return where the initial liquidity provider gave 30 * 1e18 in Ether and 66618 * 1e6 in USDC which should mean ETH costs approximately 2,228 USDC according to the initial pool balances.

Impact

Users will lose funds when attempting to make a swap, liquidity users may be able to retrieve more funds than expected when burning liquidity tokens.

Recommendations

With token decimal point precision, I recommend refactoring the swapUtils to perform fixed-point arithmetic in order to get an accurate amount of tokens out. This can be done by implementing the following:

```
Unset
// Get decimal difference
uint8 decimalsFrom = fromToken.decimals();
uint8 decimalsTo = toToken.decimals();
int8 decimalDifference = int8(decimalsFrom) - int8(decimalsTo);

// Calculate the amount to transfer with decimal adjustment
uint256 adjustedAmount;
if (decimalDifference > 0) {
    adjustedAmount = amount / (10 ** uint256(decimalDifference));
} else {
    adjustedAmount = amount * (10 ** uint256(-decimalDifference));
}
```

In addition to this, to get an accurate price of tokens, it's recommended that once [H-01] is addressed, that the constant product k is taken into account and enforced when performing swap operations.

Status

Resolved - The AMM has been removed from the project.

[H-05] Price manipulation in out of date pool UniswapV3Module.sol

Description

The poolSwap(...) function in the UniswapV3Module.sol takes a pool address that is set in config of the caller contract. To swap tokens Uniswap V3 uses liquidity pools organised into fee tiers, each with their own liquidity. Initially there were 3 fee tiers (0.05%, 0.3%, 1%), after a governance vote they introduced the option of a 0.01% fee tier. The likelihood of adding a new fee tier via governance vote is non zero and can result in liquidity migrating from existing pools. Where poolSwap is used, if there exists no path through alternative pools this would present a possible attack vector where an attacker could sell tokens at a highly unfavourable price through a pool with low liquidity.

Impact

This would result in loss of user funds.

Recommendation

To ensure the swap routing is working as intended, the manually configured pools should be replaced with dynamically searching for the fee tier with the highest liquidity available.

Status

Resolved - The UniswapV3Module has been removed from the project.

[H-06] Amount out set to 0 UniswapV3Module.sol

Description

When the swap is executed, the amountOutMinimum is set to zero. This means that regardless of how much slippage the swap incurs, the execution will continue. This poses a security risk, as attackers can perform a sandwich attack on this swap on uniswapV3 and steal funds.

Impact

Setting the min amount out to zero effectively disables security protections from frontrunning which could mean the amount out could be less than expected.

Recommendation

To set the amountOutMinimum to a reasonable value with slippage accounted for using an oracle such as chainlink.

Status

Resolved - The UniswapV3Module has been removed from the project.

[H-07] Amount out set to 0 VelodromeV2Module.sol

Description

When the swap is executed, the amountOutMin is set to zero. This means that regardless of how much slippage the swap incurs, the execution will continue. This poses a security risk, as attackers can perform a sandwich attack on this swap on VelodromeV2 and steal funds.

Impact

Setting the min amount out to zero effectively disables security protections from frontrunning which could mean the amount out could be less than expected.

Recommendation

To set the amountOutMinimum to a reasonable value with slippage accounted for using an oracle such as chainlink.

Status

Resolved - The VelodromeV2Module has been removed from the project.

[H-08] Inconsistent Logic in SwapQuoterV2.sol

Description

_isConnected(...) does not correctly implement the case for bridgePool.poolType == PoolType.Default. This should be handled the same as in _getAmountOut by following this sequence of checks to see if a pool is connected:

1. check if it is linked to defaultPool 2. check if it is linked to linkedPool
2. check if it possible to swap on a bridgePool

Impact

_isConnected does not return an accurate result if the tokens are connected leading to token trade routes not being accurately quoted.

Recommendation

For the case when bridgePool.poolType == PoolType.Default in _isConnected(...).

1. checking if tokenIn is connected to tokenOut via a default pool.
2. checking if tokenIn is connected to tokenOut via a linked pool.
3. using _isConnectedViaDefaultPool(...) to check if it is connected via any whitelisted pool

Status

Resolved - The SwapQuoterV2 has been removed from the project.

Medium

[M-01] unbounded permissioned bridging functions in DioneBridge.sol

Description

withdraw(...) mint(...) mintAndSwap(...) and withdrawAndRemove(...) are all permissioned using the role NODEGROUP_ROLE . This EOA has full control over minting and burning rights for tokens used by this bridge and proper validation and bounds should be imposed on the parameters passed into these functions.

Impact

Access control violation will allow a malicious owner to have full control over sensitive functions.

Recommendation

- The fee amount should be calculated on-chain which will reduce the impact of bugs in any off-chain code.
- The mint and burn amounts should be bounded by the total supply of the tokens.

Status

Resolved.

[M-02] Locked ETH in contract DioneBridgeERC20.sol

Description

DioneBridgeERC20.sol is able to receive native ETH via the receive() function. If this contract receives ETH there is no way to recover it.

Impact

ETH tokens may be locked inside the DioneBridgeERC20 contract.

Recommendation

Add a function to recover stuck ETH in this contract.

Status

Resolved - Added `rescueTokens` function to secure ETH tokens stuck in the contract.

[M-03] Unvalidated msg.value TimelockController.sol

Description

execute and executeBatch are both payable with no checks to ensure that the value and value[] match the msg.value amount meaning ETH can be accumulated in the contract with no reliable way to redeem without queuing another timelock action. This contract also has a receive function with no way to withdraw stuck ETH except queuing up another timelock action.

Impact

This may result in ETH being temporarily stuck in the contract.

Recommendation

Logic should be added to execute and executeBatch to ensure that the msg.value is the amount required for execution. A recoverEth() function should be added to this contract to recover any ETH that is sent to the contract.

Status

Resolved.

[M-04] Locked ETH in contract DefaultAdapter.sol

Description

DefaultAdapter.sol is able to receive native ETH via the receive() function. If this contract receives ETH there is no way to recover it.

Impact

If users transfer ETH to the contract, it will remain stuck in the contract.

Recommendation

Add a function to recover stuck ETH in this contract.

Status

Resolved.

[M-05] DioneBridge.sol role assigned with no timelock.

Description

The DioneBridge.sol contract inherits from AccessControl to assign node roles to addresses that are responsible for relaying messages across chain. Assigning roles can be done by the GOVERNANCE_ROLE . If this is assigned to an EOA or multisig controlled by a small group they would be capable of assigning mint and burn permissions for tokens used by the bridge unchecked.

Impact

In the event of a compromised EOA with the GOVERNANCE_ROLE all bridge assets would be at risk.

Recommendation

A function should be added to only allow roles to be assigned to users after a period of more than 1 day. This gives users and the protocol time to recover from a compromised EOA.

Status

Resolved - Implemented custom role assign function with timelock.

[M-06] Timeout for http.Serve in backend

Description

http.Serve uses net.Conn which represents the underlying network connection, without setting a timeout for this connection the application will keep this connection open which can lead to unbounded resource allocation.

Impact

By not setting a timeout for http.Serve(...) usage throughout the project, you are opening up the possibility of a DDOS. Not having a timeout here can lead to unbounded resource allocation.

Recommendation

Recommendation is to add a timeout for all usage of http.Serve(...)

Status

Resolved - timeout was added to http.Serve(...)

[M-07] Usage of ssh.InsecureIgnoreHostKey()

Description

Using ssh.InsecureIgnoreHostKey() in the debug proxy will allow for all hosts to connect to debug without verification of the host key.

Impact

Usage of ssh.InsecureIgnoreHostKey() is not recommended and can allow for a man in the middle attack via the debug proxy.

Recommendation

It is recommended to verify the host key before the connection is established to ensure the identity of the host.

Status

Resolved - the ssh key is validated when connecting to debug.

Centralisation

The project values security and utility over decentralisation.

The owner executable functions within the protocol increase security and functionality but depend highly on internal team responsibility.

Conclusion

After Hashlocks analysis, the Dione project seems to have a sound and well tested code base, however our findings need to be resolved in order to achieve security. Overall, most of the code is correctly ordered and follows industry best practices. The code is well commented as well. To the best of our ability, Hashlock is not able to identify any other vulnerabilities than the ones found within this report.

Our Methodology

Hashlock strives to maintain a transparent working process and to make our audits a collaborative effort. The objective of our security audits are to improve the quality of systems and upcoming projects we review and to aim for sufficient remediation to help protect users and project leaders. Below is the methodology we use in our security audit process.

Manual Code Review:

In manually analysing all of the code, we seek to find any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behaviour when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our methodologies include manual code analysis, user interface interaction, and whitebox penetration testing. We consider the project's website, specifications, and whitepaper (if available) to attain a high level understanding of what functionality the smart contract under review contains. We then communicate with the developers and founders to gain insight into their vision for the project. We install and deploy the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We undergo a robust, transparent process for analysing potential security vulnerabilities and seeing them through to successful remediation. When a potential issue is discovered, we immediately create an issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is vast because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyse the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinised by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the contracts details are made public.

Disclaimers

Hashlock's Disclaimer

Hashlock's team has analysed these smart contracts in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in the smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Hashlock is not responsible for the safety of any funds, and is not in any way liable for the security of the project.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to attacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

About Hashlock

Hashlock is an Australian based company aiming to help facilitate the successful widespread adoption of distributed ledger technology. Our key services all have a focus on security, as well as projects that focus on streamlined adoption in the business sector.

Hashlock is excited to continue to grow its partnerships with developers and other web3 oriented companies to collaborate on secure innovation, helping businesses and decentralised entities alike.

Website: hashlock.com.au
Contact: info@hashlock.com.au


#Hashlock.