GUARDIAN

Ethena

TimelockController

Security Assessment

May 26th, 2025

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Roberto Reigada, 0xCiphky,
Zdravko Hristov, Michael Lett

Client Firm Ethena

Final Report Date May 26, 2025

Audit Summary

Ethena engaged Guardian to review the security of their TimelockController with whitelist functionality. From the 15th of May to the 17th of May, a team of 5 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Confidence Ranking & Security Recommendation

Given the lack of critical issues detected and minimal code changes following the main review, Guardian assigns a Confidence Ranking of 5 to the protocol. Guardian advises the protocol to consider periodic review with future changes. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

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

📊 Code coverage & PoC test suite: https://github.com/GuardianOrg/timelock-contractethenatimelock-fuzz

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. **Recommendation:** Code is highly secure at time of audit. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	Code is clean, well-structured, and adheres to best practices. Only Low or Medium-severity issues were discovered. Design patterns are sound, and test coverage is reasonable. Small changes, such as modifying rounding logic, may introduce new vulnerabilities and should be carefully reviewed. **Recommendation:** Suitable for deployment after remediations; consider periodic review with changes.	0 High/Critical findings. Varied Low/Medium severity findings.
3: Moderate Confidence	Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas (e.g., weak modularity, risky patterns). No critical design flaws, though some patterns could lead to issues in edge cases. **Recommendation:** Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1 High finding and ≥ 3 Medium. Varied Low severity findings.
2: Low Confidence	Code shows frequent emergence of Critical/High vulnerabilities (~2/week). Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues. **Recommendation:** Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week.
1: Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (≥5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. **Recommendation:** Halt deployment and seek a comprehensive re-audit after substantial refactoring.	≥5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Overview

Project Summary

Project Name	Ethena
Language	Solidity
Codebase	https://github.com/ethena-labs/timelock-contract
Commit(s)	Initial commit: 7f02ab09de07836437d63f248ce28d1051ba6718 Final commit: 6441a5c107867a7b848c29375b8bc65a584e7852

Audit Summary

Delivery Date	May 26, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	0	0	0	0	0	0
● Medium	1	0	0	0	0	1
● Low	2	0	0	0	0	2
● Info	19	0	0	9	0	10

Audit Scope & Methodology

```
src/EthenaTimelockController.sol
```

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	● Critical	● High	● Medium
Likelihood: *Medium*	● High	● Medium	● Low
Likelihood: *Low*	● Medium	● Low	● Low

Impact

High	Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.
Medium	A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.
Low	Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High	The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.
Medium	An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.
Low	Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

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. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Invariants Assessed

During Guardian's review of Ethena, fuzz-testing was performed on the protocol's main functionalities. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 10,000,000+ runs with a prepared fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
INV-01	The setValue function in MockTarget correctly updates the value state variable to the provided input.	✅	✅	✅	10M+
INV-02	The toggleFlag function in MockTarget correctly toggles the flag state variable.	✅	✅	✅	10M+
INV-03	The complexOperation function in MockTarget correctly updates value to the sum of inputs and toggles the flag.	✅	✅	✅	10M+
INV-04	MockTarget functions (setValue, toggleFlag, complexOperation) do not cause unintended state changes.	✅	✅	✅	10M+
INV-05	Scheduled transactions in the timelock controller are marked as pending until executed.	✅	✅	✅	10M+
INV-06	Transactions cannot be executed before their scheduled delay period has elapsed.	✅	✅	✅	10M+
INV-07	Executed transactions in the timelock controller are marked as done and not pending.	✅	✅	✅	10M+

Findings & Resolutions

ID	Title	Category	Severity	Status
M-01	Native Asset Transfers Always Revert	Logical Error	● Medium	Resolved
L-01	Ether Refunds Break Balance Invariant	Trapped Funds	● Low	Resolved
L-02	Stuck Actions Due To Missing Whitelist Check	Validation	● Low	Resolved
I-01	Ether Sent Is Trapped	Trapped Funds	● Info	Resolved
I-02	Misleading Event Emission	Events	● Info	Resolved
I-03	Lacking minDelay Validation	Validation	● Info	Acknowledged
I-04	Upgradeable Target Whitelisting Risk	Warning	● Info	Acknowledged
I-05	Missing Documentation	Documentation	● Info	Acknowledged
I-06	Missing Event Data	Events	● Info	Resolved
I-07	Potential Censoring With Open Execution	Warning	● Info	Acknowledged
I-08	Lacking Admin Rules	Best Practices	● Info	Acknowledged
I-09	renounceRole Called Without Timelock	Warning	● Info	Resolved
I-10	Missing Zero Address Checks	Validation	● Info	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
I-11	Deployment Script Missing Validations	Validation	● Info	Resolved
I-12	Magic Selector Length Number	Best Practices	● Info	Resolved
I-13	Timelock Cannot Interact With Precompiles	Warning	● Info	Acknowledged
I-14	Unnecessary Unchecked Blocks	Gas Optimization	● Info	Resolved
I-15	Predecessor Is Not Used For Whitelisted Actions	Warning	● Info	Resolved
I-16	Executor Role Can Execute Whitelisted Functions	Documentation	● Info	Acknowledged
I-17	Malicious Canceller Can DOS The Timelock	DoS	● Info	Acknowledged
I-18	Timelock Is Not Compatible With Some Chains	Warning	● Info	Resolved
I-19	Timelock Unable To Blacklist	Logical Error	● Info	Acknowledged

M-01 | Native Asset Transfers Always Revert

Category	Severity	Location	Status
Logical Error	● Medium	EthenaTimelockController.sol: 192	Resolved

Description

When execute or executeWhitelistedBatch attempts to parse the function selector via bytes4(data[:4]), any call where data.length < 4 (e.g. a pure Ether transfer that passes an empty data payload) will revert because slicing data[:4] triggers an out-of-bounds error.

This deviates from the standard TimelockController behavior, which allows zero-length data for native asset transfers.

Notice the following check in the _execute function:

```
function _execute(address target, uint256 value, bytes calldata data) internal virtual override {
      if (data.length > 0 target.code.length = 0) revert InvalidTarget(target);
      super._execute(target, value, data);
}
```

The current test testExecuteWithValue passes because:
1. target is a smart contract.
2. abi.encodeWithSignature("") actually returns non-empty calldata: 0xc5d24601

However, if the target was an EOA it would revert with InvalidTarget(EOA_ADDR) error, because data.length check would pass in the _execute function but target.code.length = 0 would not.

And, if we tried to, set data.length to 0, it would revert in the EthenaTimelockController.execute function, when trimming the call data as the call data would be empty as we initially stated.

Recommendation

Consider checking that data.length is > than 4 before setting: bytes4 selector = bytes4(data[:4]);.

Resolution

Ethena Team: The issue was resolved in PR#4. We have added _extractSelector to mitigate this and followed up with the test testCanWhitelistAndExecuteEmptyBytes4DataToSmartContract to verify the functionality.

L-01 | Ether Refunds Break Balance Invariant

Category	Severity	Location	Status
Trapped Funds	● Low	EthenaTimelockController.sol	Resolved

Description

In the execute, executeBatch, and executeWhitelistedBatch functions the msg.value is validated to be exactly equal to the total of the value parameters provided. This validation is performed in an effort to ensure that no ETH is retained in the contract as mentioned in the README.

However, in the event that Ether is sent to a function and any portion is refunded by the arbitrary function call then this amount of ETH will be left in the EthenaTimelockController function and will not be rescuable due to the msg.value validations.

Refunds from arbitrary function calls are made possible by the receive function in the OpenZeppelin base TimelockController contract. Furthermore, if any calls with nonzero value are made with the EthenaTimelockController contract as the target, this Ether will be left within the contract.

Recommendation

Consider performing a balance check at the end of the execute, executeBatch, and executeWhitelistedBatch functions and refunding any ether in the contract to the caller, under the assumption that the caller can accept ether, or a holding address.

Resolution

Ethena Team: The issue was resolved in PR#4. We have removed this check to allow funds to end up in the TimelockController contract and followed up with test testEthCanBeRescuedFromTimelock and testErc20TokensCanBeRescuedFromTimelock to validate that funds can be recovered if they do end up in the controller. Under normal circumstances we do not expect funds to end up in the controller unless there is a difference between msg.value and the value parameters or a refunds from an arbitrary function call is made or someone sends Ether directly into the TimelockController contract.

L-02 | Stuck Actions Due To Missing Whitelist Check

Category	Severity	Location	Status
Validation	● Low	EthenaTimelockController.sol	Resolved

Description

In the schedule function there is no validation to ensure that a whitelisted function is not scheduled through the normal timelock flow. As a result, when the scheduled whitelisted function becomes executable, it cannot be executed because the overridden execute function does not invoke super.execute and clear the action from the timelock.

If any actions were based on the whitelisted function call that was scheduled as a predecessor, these subsequent actions are stuck. Even in the case where the scheduled whitelisted action is cancelled, as it never reaches a OperationState.Done state.

Consider the following series of actions as an example:

• Action A is created with a whitelisted function selector in it's payload

• Action B is created for a non-whitelisted function with Action A as its predecessor

• Action A cannot be completed from the scheduled queue since the execute function does not allow the base TimelockController logic to be executed.

• Action B cannot be executed since Action A is listed as its predecessor, even if Action A is cancelled.

• All actions will have to be cancelled and re-queued, forcing an additional wait time of the minDelay.

Furthermore, this scenario can also arise if the function selector of a pending action is whitelisted after that action was already scheduled.

Recommendation

Consider overriding the schedule function to prevent whitelisted actions from being queued. Be aware that if this solution is adopted, whitelisting a function in the queue will also yield this behavior.

Alternatively, consider separating the whitelisted execution logic and non-whitelisted execution logic into separate functions. Notice that this is similar to the approach for batch executions, which do not exhibit this deficiency.

Resolution

Ethena Team: The issue was resolved in PR#4. We have opted to separate the whitelisted execution logic and non-whitelisted execution logic into execute and executedWhitelisted.

I-01 | Ether Sent Is Trapped

Category	Severity	Location	Status
Trapped Funds	● Info	EthenaTimelockController.sol	Resolved

Description

The base TimelockController contract from OpenZeppelin implements a receive function and therefore ether can be sent directly to the EthenaTimelockController contract.

Any ether sent to the EthenaTimelockController contract this way will be unrescuable as a result of the msg.value checks in the execute, executeBatch, and executeWhitelistedBatch functions.

Recommendation

Consider either adding a rescue function, implementing refunds in the execution functions as mentioned in L-01, or implementing a receive function that reverts to disable direct Ether transfers.

Resolution

Ethena Team: The issue was resolved in PR#4.

I-02 | Misleading Event Emission

Category	Severity	Location	Status
Events	● Info	EthenaTimelockController.sol: 91	Resolved

Description

In the removeFromWhitelist function the FunctionRemovedFromWhitelist event is emitted regardless of whether the specific target address and function selector were in the whitelist to begin with.

This could hypothetically be misleading to consumers of the FunctionRemovedFromWhitelist in the event that the target and selector combination value was not previously set as true in the _functionWhitelist mapping.

Similarly the addToWhitelist function performs no validation that the function was not already added to the whitelist.

Recommendation

Consider validating that the _functionWhitelist entry for the specified target and selector is true in the removeFromWhitelist function and false in the addToWhitelist function.

Resolution

Ethena Team: The issue was resolved in PR#4.

I-03 | Lacking minDelay Validation

Category	Severity	Location	Status
Validation	● Info	EthenaTimelockController.sol: 46	Acknowledged

Description

There is no validation on the `minDelay` value in the constructor of the `EthenaTimelockController` contract. As a result the `minDelay` may technically be assigned to an insufficient delay, even though the deployment script currently sets the minDelay as 1 day.

Recommendation

Consider implementing validation for the `minDelay` value, otherwise be aware of this lack of validation during deployment.

Resolution

Ethena Team: Acknowledged.

I-04 | Upgradeable Target Whitelisting Risk

Category	Severity	Location	Status
Warning	● Info	EthenaTimelockController.sol	Acknowledged

Description

Arbitrary target contracts may be whitelisted with the `addToWhitelist` function. The target address in question may be a proxy contract which can have the implementation of it's whitelisted selector upgraded.

For this reason, care should be taken to examine the centralized risk of whitelisted target contracts.

Recommendation

Be aware of this risk and consider the centralized upgrade risks for any upgradeable contracts which are whitelisted.

Resolution

Ethena Team: Acknowledged. We do not intend to use `addToWhitelist` on functions outside of contracts that are currently controlled by the Ethena `multisig`. Assuming that is true and that the upgrade function on the target contract is not whitelisted, any upgrades happening to target whitelist contracts would have to go through the `minDelay`.

I-05 | Missing Documentation

Category	Severity	Location	Status
Documentation	● Info	README.md	Acknowledged

Description

In the README it is mentioned that there are strict `msg.value` checks for the `execute` and `executeWhitelistedBatch` functions, however there is also the same strict `msg.value` check in the `executeBatch` function which is not mentioned.

Recommendation

Amend the README file to include that the `executeBatch` function also implements the `msg.value` invariant check.

Resolution

Ethena Team: Acknowledged.

I-06 | Missing Event Data

Category	Severity	Location	Status
Events	● Info	EthenaTimelockController.sol	Resolved

Description

The `WhitelistedFunctionExecuted` event does not include relevant information such as the value and payload of the execution. These data fields are in the base `TimelockController` `CallExecuted` event and if anything would provide parity with this event.

Recommendation

Consider including value and payload fields in the `WhitelistedFunctionExecuted` event.

Resolution

Ethena Team: Resolved.

I-07 | Potential Censoring With Open Execution

Category	Severity	Location	Status
Warning	● Info	EthenaTimelockController.sol	Acknowledged

Description

The `EXECUTOR_ROLE` is planned to be left as an open role, however this introduces a potential censoring vector whereby a malicious actor may be able to control the execution flow of certain external calls.

For a particular external call where the execution of the target function uses a try/catch or allows the failure of a nested external call, a malicious actor may be able to perform the execution while providing insufficient gas to the `execute` or `executeBatch` functions such that the nested external call runs out of gas and fails but allows the top level transaction to succeed with the remaining 1/64 gas.

Recommendation

None of the existing permissioned functions which may be used by the `EthenaTimelockController` were observed to use a try/catch or allow the failure of an external call, therefore this is not an immediate concern. However be aware of this risk when adapting future use-cases for the timelock contract.

Resolution

Ethena Team: Acknowledged.

I-08 | Lacking Admin Rules

Category	Severity	Location	Status
Best Practices	● Info	EthenaTimelockController.sol	Acknowledged

Description

The base TimelockController contract does not implement the DefaultAdminRules contract nor any base default admin role protections.

Similarly, other Ethena related contracts use a SingleAdminAccessControl contract which provides a simpler implementation of default admin role protections.

Recommendation

Consider implementing the protections from the SingleAdminAccessControl contract for the default admin role as seen in other Ethena contracts.

Resolution

Ethena Team: Acknowledged. Worst case here is that someone could propose the Admin to renounce or revoke their role which would mean that other roles cannot be managed. In this scenario it is still possible for the timelock contract the propose the underlying contract to change its admin.

I-09 | renounceRole Called Without Timelock

Category	Severity	Location	Status
Warning	● Info	EthenaTimelockController.sol	Resolved

Description

In the AccessControl contract the renounceRole function is exposed to be called by any address without going through the timelock period. This does not pose any notable risk and this finding serves only to document that this action is not kept behind a timelock.

Recommendation

Be aware of this behavior, and if desired consider overriding the renounceRole function and requiring that it goes through the queued delay with the onlyTimelock modifier.

Resolution

Ethena Team: The issue was resolved in commit 573bbdb. We've removed the ability for addresses to renounce roles directly. They must be revoked through the timelock https://github.com/ethena-labs/timelock-contract/pull/4/commits/573bbdb8aa5e213df79f2196939fecb3b3c82cf7.

I-10 | Missing Zero Address Checks

Category	Severity	Location	Status
Validation	● Info	EthenaTimelockController.sol	Resolved

Description

In the constructor of the EthenaTimelockController contract the proposers list is validated to have all nonzero entries, however no such validation is performed for the whitelistedExecutors list which should also not contain the zero address.

Recommendation

Consider implementing a zero address validation for the whitelistedExecutors list.

Resolution

Ethena Team: The issue was resolved in PR#4.

I-11 | Deployment Script Missing Validations

Category	Severity	Location	Status
Validation	● Info	DeployEthenaTimelockController.sol	Resolved

Description

In the deployment script for the EthenaTimelockController contract there is validation performed on the first entry of the proposers and executors lists.

Firstly, the entire proposers and executors lists could be validated to ensure that all proposers and executors received the expected roles, in the event that these lists were extended before deployment.

Secondly, the whitelistedExecutors list is not validated to ensure that the intended whitelisted executors received the expected WHITELISTED_EXECUTOR_ROLE role.

Finally, the proposer addresses will each also receive the CANCELLER_ROLE which can also be validated for.

Recommendation

Be aware of these potential deployment script improvements and consider implementing them if you wish.

Resolution

Ethena Team: Resolved.

I-12 | Magic Selector Length Number

Category	Severity	Location	Status
Best Practices	● Info	EthenaTimelockController.sol	Resolved

Description

In the `EthenaTimelockController` contract the value 4 is repeatedly used to denote the length of a function selector and create a slice of the selector. As a best practice magic numbers can be referenced as named constants to improve code readability.

Recommendation

Consider creating a constant `SELECTOR_LENGTH` value to use when slicing selectors.

Resolution

Ethena Team: Resolved.

I-13 | Timelock Cannot Interact With Precompiles

Category	Severity	Location	Status
Warning	● Info	EthenaTimelockController.sol: 192	Acknowledged

Description

Within the `_execute` override in `EthenaTimelockController`, there is a conditional that reverts if `data.length > 0` and `target.code.length = 0`:

```
if (data.length > 0 && target.code.length = 0) {revert InvalidTarget(target);}
```

This prevents the contract from calling addresses where `code.length` is zero. While this is intended to block whitelisting for addresses with empty code, it also blocks calls to Ethereum precompiles (e.g., 0x1 through 0x9 on mainnet), which often have no bytecode in the traditional sense.

Consequently, any attempt to schedule or execute interactions with precompiles will revert.
Keep in mind that since the Pectra update EOAs can also have code in their accounts since they via delegatation. This will bypass the check in `_execute` and allow execution of an EOA's code.

Recommendation

• If interacting with precompiles is a requirement, remove or relax the `target.code.length = 0` check.
• Alternatively, explicitly allow known precompile addresses in the code or via a separate allow-list so that legitimate precompiles can be called without inadvertently allowing empty addresses.

Resolution

Ethena Team: The issue was resolved in PR#4. Precompiled are not intended to be called directly by the `TimelockController` contract. Fixed through the splitting of execute and `executeWhitelisted` which allows us to remove this override.

I-14 | Unnecessary Unchecked Blocks

Category	Severity	Location	Status
Gas Optimization	● Info	EthenaTimelockController.sol	Resolved

Description

Since Solidity 0.8.22, the compiler performs advanced range analysis to determine when overflow checks can safely be omitted.

In a typical loop increment scenario such as for `(uint256 i = 0; i < length; i++)`, the compiler can prove that i will not overflow a `uint256` and therefore it automatically removes the checks.

Manually placing increments in an unchecked block is now redundant, making the code less clear without providing a meaningful gas optimization.

Recommendation

Replace `unchecked { ++i; }` with a simple `++i`, allowing the compiler's range analysis to handle overflow optimizations automatically. This cleaner style increases code clarity while maintaining code efficiency in Solidity 0.8.22 and above.

Resolution

Ethena Team: The issue was resolved in PR#4.

I-15 | Predecessor Is Not Used For Whitelisted Actions

Category	Severity	Location	Status
Warning	● Info	EthenaTimelockController.sol	Resolved

Description

When a whitelisted operation is being executed by calling `execute()`, the `predecessor` parameter is ignored, even though the `NatSpec` says `* @param predecessor The operation that must be executed before this one.`

Because of this, whitelisted executors may expect that their operations will be executed only if the predecessor is already done, but in reality their operations will be executed regardless of the predecessor.

Recommendation

Either clarify that the predecessor is not used for whitelisted operations in the `NatSpec` or modify the code to actually use it.

Resolution

Ethena Team: The issue was resolved in PR#4. Fixed through splitting out of `executeWhitelisted` function.

I-16 | Executor Role Can Execute Whitelisted Functions

Category	Severity	Location	Status
Documentation	● Info	EthenaTimelockController.sol: 141	Acknowledged

Description

The protocol documentation states that “only whitelisted executors can execute whitelisted functions” as a invariant. It also claims that batches cannot be scheduled if they contain whitelisted functions.

However, the current implementation allows scheduling such batches. This breaks the documented feature and effectively bypasses the invariant, as any address with `EXECUTOR_ROLE` can then execute whitelisted functions via the executeBatch function.

Recommendation

If this behaviour is intended, clarify it in the documentation. If not, add a check to prevent scheduling batches that include whitelisted functions.

Resolution

Ethena Team: Acknowledged. One caveat here is that a non-whitelisted function can be scheduled and then subsequently added to the whitelist before the scheduled function is executed.

I-17 | Malicious Canceller Can DOS The Timelock

Category	Severity	Location	Status
DoS	● Info	TimelockController.sol	Acknowledged

Description

When TimelockController is created, all of the proposers receive the `PROPOSER_ROLE` and `CANCELLER_ROLE`, which allow them to both propose and cancel transactions.

This gives them a lot of power, since they can cancel any transaction they want to before the timelock delay has passed.

Normally, if any of the proposers becomes malicious, their role should be revoked by calling AccessControl.revokeRole() which can be only executed by the admin of the given role.

In the case of the timelock, the admin is the address with the `DEFAULT_ADMIN_ROLE` - this address is the timelock itself.

Because of this, the revoke transaction itself is a subject to the timelock delay, which makes it possible for the malicious proposer to just cancel it, keeping their `CANCELLER_ROLE`.

Once a proposer is compromised, the result is DOS of the timelock and permanent freezing of the funds it holds (since any transaction can be cancelled).

Recommendation

Be careful with who you give the `CANCELLER_ROLE`, ideally only to one account.

Resolution

Ethena Team: Acknowledged. Proposer and canceller roles are intended to be assigned only to the multisig that is the current controller of the ethena protocol contracts.

I-18 | Timelock Is Not Compatible With Some Chains

Category	Severity	Location	Status
Warning	● Info	EthenaTimelockController.sol	Resolved

Description

The timelock inherits from `ReentrancyGuardTransient` which makes use of the `tload` and `tstore` opcodes introduced in `EIP-1153`. If the contract were to be deployed to a chain with no support for these opcodes, the functionality would be broken.

Recommendation

Keep in mind you need to change the reentrancy guard if you are going to deploy to such chains.

Resolution

Ethena Team: Resolved. We've opted to replace with `ReentrancyGuard` instead. The gas saving is not a concern.

I-19 | Timelock Unable To Blacklist

Category	Severity	Location	Status
Logical Error	● Info	StakedUsde.sol: 100	Acknowledged

Description

The NatSpec for the addToBlacklist function in the StakedUSDe contract states that it Allows the owner (DEFAULT_ADMIN_ROLE) and blacklist managers to blacklist addresses.

However, the addToBlacklist and removeFromBlacklist functions use the onlyRole(BLACKLIST_MANAGER_ROLE) modifier which only allows the blacklist manager role to execute these functions.

As a result if the EthenaTimelockController is only granted the DEFAULT_ADMIN_ROLE over the contracts it will be unable to invoke the addToBlacklist or removeFromBlacklist functions.

Recommendation

Be aware of this discrepancy and be sure to assign the BLACKLIST_MANAGER_ROLE to the EthenaTimelockController in addition to the DEFAULT_ADMIN_ROLE for the StakedUSDe contract.

Resolution

Ethena Team: Acknowledged. It should be noted that the DEFAULT_ADMIN_ROLE can manage addresses with the BLACKLIST_MANAGER_ROLE. The blacklisting role in StakedUSDe is intended to be given to addresses other than the admin.

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