DeHacker

Code Security Assessment

KILT Protocol

May 9th, 2025

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best prac tices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	KILT Protocol
Platform	Base Blockchain | Web Application
Website	https://www.kilt.io
Type	DeFi
Language	Solidity, TypeScript
Codebase	3ad8ac1e62665ba0c6bd3c6fd4b175b06d26c23273 42ad1d1886e8b0d35a433c

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	2	0	0	0	2	0
Medium	2	0	0	1	0	1
Minor	1	0	0	0	0	1
Informational	4	0	0	1	0	3
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
KIB	KILT_migration.sol	a009899e4a793c3743f2d7b70f4b253e6a59d1d0497cc80b4d09f2080a1de273
KIL	pages/index.js	8aea20296c96d561fa2369307894b94e994980761958f7ce2a454582bf783cbd
KIT	pages/dashboard.js	b7889db2b73e2ec3fd76c7eea5a4bd26229bcdf2d094c48b8d57d0dba9083a46
KIA	KILT	3ad8ac1e62665ba0c6bd3c6fd4b175b06d26c2327342ad1d1886e8b0d35a433c

Findings

ID	Title	Severity	Status
KIP-01	Centralization Risks In KILT_migration.Sol	Major	Partially Resolved
KIP-02	Initial Token Distribution	Major	Partially Resolved
KIP-03	Fixed 1.75:1 Token Migration Ratio May Cause Value Discrepancies And Unfair Distribution	Medium	Acknowledged
KIP-06	Silent And Forced Wallet Network Switch	Medium	Resolved
KIP-07	Missing Security Headers	Minor	Resolved
KIP-04	Inconsistent ETH Handling: Contract Reverts On Direct ETH Transfers But Includes Recover ETH For ETH Recovery	Informational	Resolved
KIP-05	Zero Migration Amount Incorrectly Triggers Approval Status	Informational	Resolved
KIP-08	Using Components With Known Vulnerabilities	Informational	Resolved
KIP-09	GetMigrationStatusAlways ReturnsFalse Due To MisuseOf Paused Variable	Informational	Acknowledged

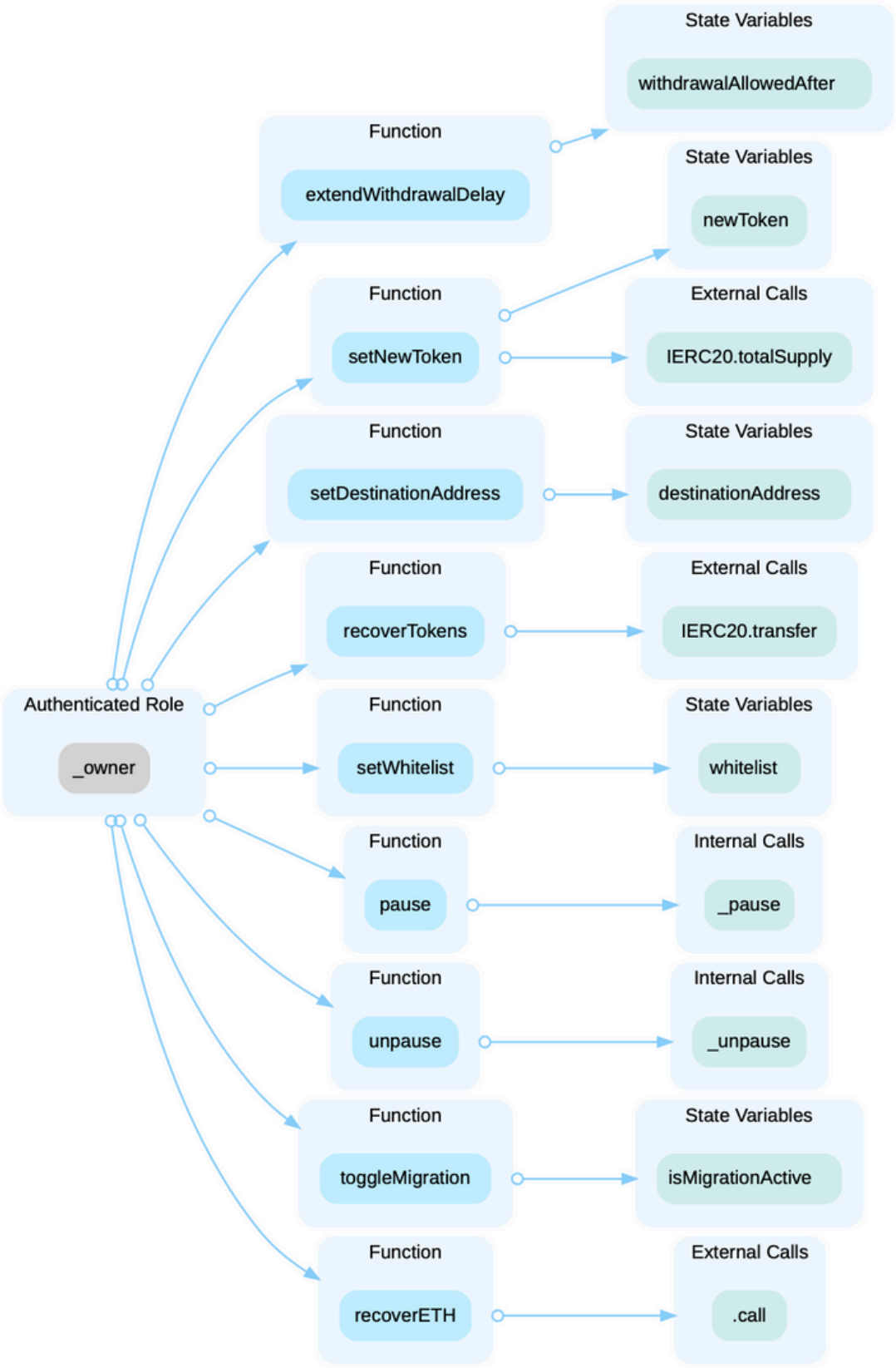
MAJOR

KIP-01 | Centralization Risks In KILT_migration.Sol

Issue	Severity	Location	Status
Centralization	Major	KILT_migration.sol (v1): 103,114, 123, 132, 139, 146, 180, 191, 200	Partially Resolved

Description

In the contract KILT Migration , the role _owner has authority over the functions shown in the diagram below. Any compromise to the _owner account may allow the hacker to take advantage of this authority and extend the withdrawal delay, set a new token address, set the destination address, recover specified tokens after a certain time, set white list status for an account, pause the contract, unpause the contract, toggle the migration status, and recover ETH after the allowed withdrawal time.


State Variables
withdrawalAllowedAfter
Function
extendWithdrawalDelay
State Variables
newToken
Function
setNewToken
External Calls
IERC20.totalSupply
Function
setDestinationAddress
State Variables
destinationAddress
Function
recoverTokens
External Calls
IERC20.transfer
Authenticated Role
_owner
Function
setWhitelist
State Variables
whitelist
Function
pause
Internal Calls
_pause
Function
unpause
Internal Calls
_unpause
Function
toggleMigration
State Variables
isMigrationActive
Function
recoverETH
External Calls
.call

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:
Timelock and Multi sign (⅔, ⅗) combination mitigate by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:
Timelock and DAO, the combination, mitigate by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:
Renouncing the ownership or removing the function can be considered fully resolved.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

MAJOR

KIP-02 | Initial Token Distribution

Issue	Severity	Location	Status
Centralization	Major	KILT (v1): 25~26	Partially Resolved

Description

13,587,418 KILT tokens are sent to the treasury address, and 276_972_582 KILT tokens are sent to the migrationcontract address. This is a centralization risk because the deployer or the owner(s) of the EOAs can distribute tokens withoutobtaining the consensus of the community. Any compromise to these addresses may allow a hacker to steal and sell tokenson the market, resulting in severe damage to the project.

Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution planshould be published in a public location that the community can access. The team should make efforts to restrict access tothe private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point offailure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vestingschedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greateraccountability.

MEDIUM

KIP-03 | Fixed 1.75:1 Token Migration Ratio May Cause Value Discrepancies And Unfair Distribution

Issue	Severity	Location	Status
Logical Issue	Medium	KILT_migration.sol (v1): 157	Acknowledged

Description

The contract handles migration from old to new tokens using a hardcoded exchange ratio of 1.75:1. This assumes that 1 oldtoken is always worth 1.75 new tokens in terms of value. However, if the market value of the two tokens diverges from thisratio — due to trading, speculation, or price manipulation — then the migration may result in either overcompensation orundercompensation. Malicious users may also exploit temporary price differences to gain profit, while honest users mayreceive less value than intended.Additionally, an incorrect ratio may also result in an insufficient number of new tokens for migration.

Recommendation

Avoid using fixed migration ratios unless the tokens are tightly pegged or have guaranteed price equivalence. Consider integrating a trusted price oracle (e.g., Chainlink) to fetch real-time price data and adjust the conversion rate accordingly. If a dynamic rate is not feasible, clearly communicate the risks and restrict the migration period to reduce exposure to marketvolatility.

MEDIUM

KIP-06 | Silent And Forced Wallet Network Switch

Issue	Severity	Location	Status
Logical Issue	Medium		Resolved

Description

The front-end code automatically checks whether the user's wallet is connected to the intended network. If not, it silently andforcibly switches the wallet's network without user interaction:

```
useEffect(() => {
    if (network?.chain?.id !== 84532 && switchNetwork) {
      switchNetwork(84532);
    }
}, [network, switchNetwork]);
```

This implementation introduces the following security and usability concerns:

1. **Interference with Multi-DApp Usage:** If the user is interacting with multiple Dapps in the same browser, opening the migration page may forcibly switch the wallet network, disrupting other active sessions.

2. **Phishing Pattern Mimicry:** Silent network switching is a behavior commonly associated with phishing or malicious Dapps. Legitimate applications should request user confirmation before attempting to switch networks to build trustand maintain transparency.

Recommendation

To securely handle the network, it's recommended that the website perform the following when handling wallet's network:

1. Check if the user's wallet is on the correct network.
2. If not on the correct network, display a prompt to warn the user and ask for the user's permission to change the network.
3. If the user does not switch to the correct network, gray out the Approve/Migrate button and prompt the user to switch to the correct network.

This approach minimizes the risk of accidental or suspicious behavior and aligns with best practices for user consent and security.

MINOR

KIP-07 | Missing Security Headers

Issue	Severity	Location	Status
Security Misconfiguration	Minor		Resolved

Description

HTTP headers are well known and their implementation can make your application more versatile and secure. Modern browsers support many HTTP headers that can improve web application security to protect against click jacking, cross-site scripting, and other common attacks. The important ones are missing and should be added to the response headers. They are listed below.

- Content-Security-Policy - Content Security Policy is an effective measure to protect your site from XSS attacks.By white listing sources of approved content, you can prevent the browser from loading malicious assets.
- X-Frame-Options - X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like click jacking. Recommended value"X-Frame-Options: SAMEORIGIN".
- X-Content-Type-Options - X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declaration.

Recommendation

It is recommended to set the missing Security HTTP response headers. Following are the examples to securely implement them. (Note: Currently the X-XSS-Protection header is deprecated)

- X-Content-Type-Options: nosniff
- X-Frame-Options: SAMEORIGIN
- Content-Security-Policy: default-src 'self' (Note: Content Security Policy has a significant impact on how the browser renders pages, so careful tuning is required.)

INFORMATIONAL

KIP-04 | Inconsistent ETH Handling: Contract Reverts On Direct ETH Transfers But Includes Recover ETH For ETH Recovery

Issue	Severity	Location	Status
Inconsistency	Informational	KILT_migration.sol (v1): 200, 239	Resolved

Description

The KILTMigration contract is designed to reject ETH transfers by reverting any transaction that sends ETH to it directly. Despite this, it exposes a recoverETH function, which enables the contract owner to retrieve any ETH held by the contract.This introduces a logical inconsistency: if ETH is not allowed to be received, there should be no need for recovery. This could mislead users or auditors into assuming ETH might be accepted under certain edge conditions. If ETH is only expected to arrive via mechanisms like self destruct, this should be documented and tested, as these scenarios can have security implications.

Recommendation

Clarify the expected sources of ETH in the contract's documentation. If ETH is not expected under any conditions, consider removing the recoverETH function entirely to simplify the contract and its trust assumptions.

INFORMATIONAL

KIP-05 | Zero Migration Amount Incorrectly Triggers Approval Status

Issue	Severity	Location	Status
Volatile Code	Informational	pages/index.js (v1): 133	Resolved

Description

In the migration UI, entering a migration amount of zero triggers the checkAllowance function, which sets the isApproved flag to true . This causes the action button to display "Migrate," misleading users into thinking the operation is valid. Upon clicking the button, an alert appears with the message "Amount must be positive and less than or equal to your balance." The UX inconsistency stems from allowing isApproved to be true for a non-positive amount.

Recommendation

Add a check to ensure the migration amount is greater than zero before marking isApproved as true in checkAllowance . This will prevent confusion and improve the user experience.

INFORMATIONAL

KIP-08 | Using Components With Known Vulnerabilities

Issue	Severity	Location	Status
Security Misconfiguration	Informational		Resolved

Description

Applications typically rely on several open-source components. Developers can develop rich and complex applications with little cost and effort by integrating open-source development tools, frameworks, and languages. Security researchers and testers typically use automated tools to uncover compromised or vulnerable components, then publish their findings on issue trackers, security advisories, or the National Vulnerability Database (NVD). Any competent attacker who finds this information can use it to exploit particular application surfaces.

The result of the npm audit command run against the repository shows that the application is using outdated libraries with known vulnerabilities.

```
26 vulnerabilities (6 low, 19 high, 1 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force
Peiyus-MacBook-Pro:migration_portal-796d0c1806db55480c635c9a4704055a96aec9ed fly$ npm audit
```

Recommendation

There should be a patch management process in place to:

- Remove unused dependencies, unnecessary features, components, files, and documentation.

- Continuously inventory the versions of both client-side and server-side components (e.g. frameworks, libraries) and their dependencies using tools like versions, DependencyCheck, retire.js, etc.
- Continuously monitor sources like CVE and NVD for vulnerabilities in the components. Use software compositionanalysis tools to automate the process. Subscribe to email alerts for security vulnerabilities related to components you use.
- Only obtain components from official sources over secure links. Prefer signed packages to reduce the chance of including a modified, malicious component.
- Monitor for libraries and components that are unmaintained or do not create security patches for older versions. If patching is not possible, consider deploying a virtual patch to monitor, detect, or protect against the discovered issue.

Organizations should ensure that there is an ongoing plan for monitoring, triaging, and applying updates or configuration changes for the lifetime of the application or portfolio.

INFORMATIONAL

KIP-09 | GetMigrationStatus Always Returns False Due To Misuse Of Paused Variable

Issue	Severity	Location	Status
Inconsistency	Informational		Acknowledged

Description

The getMigrationStatus function uses the paused variable to determine whether migration is active. However, due to how the paused variable is used, the function always returns false, regardless of the actual migration state.

```
function getMigrationStatus() external view returns (
        bool active,
        bool paused,
        uint256 withdrawalDelay,
        address treasury,
        uint256 newTokenBalance
) {
    return (
        isMigrationActive,
        paused,
        withdrawalAllowedAfter,
        destinationAddress,
        newToken.balanceOf(address(this))
    );
}
```

Recommendation

Update the 'getMigrationStatus' implementation to reflect the correct migration status logic and ensure it returns the intended state.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could prese nt security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced b y or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will n ot be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style
Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code
Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue
Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

- Ethereum
- Cosmos
- Eos
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- c++

CONTACTS

- https://dehacker.io
- https://twitter.com/dehackerio
- https://github.com/dehacker/audits_public
- https://t.me/dehackerio
- https://blog.dehacker.io/

May 9th 2025