HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Kresus Labs
Date: April 21, 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Kresus Labs
Approved By	Marcin Ugarenko | Lead Solidity SC Auditor at Hacken OU
Type	Storage Vault
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://www.kresus.com/
Changelog	09.03.2023 - Initial Review 03.04.2023 - Second Review 21.04.2023 - Third Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Kresus Labs (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project includes the following smart contracts from the provided repository:

Initial review scope

Repository	https://github.com/Kresus-Labs-Inc/vault-mpc/tree/feat/mpc
Commit	38db3f4
Whitepaper	-
Functional Requirements	Readme document
Technical Requirements	Readme document + NatSpec in the code
Contracts	File: ./contracts/Factory.sol SHA3: f40797d8faf22af7a041a9ca3dbb0c79661651fcb1ba3ffad8806064b6830046 File: ./contracts/modules/common/BaseModule.sol SHA3: ee124fec515881d2b24f9500ffd2082bcb09729e9f54329163fac5e594bfa88e File: ./contracts/modules/common/IModule.sol SHA3: 542848acc19cd245eecd6e573894df0a00f205f324eead65428396d80bb8627c File: ./contracts/modules/common/Utils.sol SHA3: 64bd5204cdc5b7329ea649c0cfb0e46ff9a7d04de362af7d86bb84c955541409 File: ./contracts/modules/KresusModule.sol SHA3: 3263598d35287698a79d836792af0e8a4ce83a06fce3f91ddff040f820b5f792 File: ./contracts/modules/KresusRelayer.sol SHA3: 82c405f031c474891996af3042a311182c91618b226d4e2c7745aadcc0712d53 File: ./contracts/modules/SecurityManager.sol SHA3: ed34b6c54169a3bed0be0c80346ffec0a6d0fa77e4dfd1d0676442a0fb2353bd File: ./contracts/modules/TransactionManager.sol SHA3: e12e61a9bfd02c2862a9bc1e17f827e894afd6580e18209879c74b324be239b0 File: ./contracts/storage/IStorage.sol SHA3: a3b674f90b6fd2a55ed4b2481d42942f7a56b231960ac45b4f58b39d147ec193 File: ./contracts/storage/Storage.sol SHA3: 5c98f0d0073cf1cb521f04605478e31e336ad97da51612922e4149347390824f File: ./contracts/vault/BaseVault.sol SHA3: d4922f1ab3e97c937761f13f99d2753e03a2262d981a05ade591d41cc0718772

	File: ./contracts/vault/IVault.sol SHA3: 7913db46db7ec36b15bcabb237666a4dbc3aa6cc0b1d54511ec7050fc87008b2 File: ./contracts/vault/Proxy.sol SHA3: c390b6dd7577c9f2cfab0d73a5ec89a975fecc3d07e36c8b9b937ecfc5741a31

Second review scope

Repository	https://github.com/Kresus-Labs-Inc/vault-mpc/tree/feat/mpc
Commit	e3b7611
Functional Requirements	Readme document
Technical Requirements	Readme document + NatSpec in the code
Contracts	File: ./contracts/infrastructure/Factory.sol SHA3: c51665cff75c792252c8b4b3fff931fd0c88c1642bd517d7f4d2cd619cd085b1 File: ./contracts/infrastructure/IModuleRegistry.sol SHA3: d7c023d66ad1fcbac4076b3c78261cee7829a3e1cc9615c5bc794576e10f7739 File: ./contracts/infrastructure/KresusUpgrader.sol SHA3: 01918cdaf48ad5894035b3b66f3f65c59f1861272f2502bd9c8287604edda2d5 File: ./contracts/infrastructure/ModuleRegistry.sol SHA3: 914744389e8d20de51cffc882c073dca03724a458054e7d3f934bde6494a0133 File: ./contracts/modules/common/BaseModule.sol SHA3: 0b16e91e581d9adfb21ee94498bae871a59ac6cec7a9512771cedb7cd8a56a49 File: ./contracts/modules/common/IModule.sol SHA3: afc9785569ce9cd91d36bc7e5e71dbbb3f9e9aada2b683df201fae64de3d55f7 File: ./contracts/modules/common/Utils.sol SHA3: 1e89156d919f172b3ba2129f0a01c81753ea2c0a0e3ed9339dc03b3d1f485f3c File: ./contracts/modules/KresusModule.sol SHA3: 2d7550cc995b13dbb6209ea10b72c8d52426a04be33f82c69d7c65fb1cc748a5 File: ./contracts/modules/KresusRelayer.sol SHA3: fe5642a8933dbddc1aa02fdce1858ede5e4eb798fb1fb07e4a60b4de1bfc275b File: ./contracts/modules/SecurityManager.sol SHA3: 8b3afac0d10ffc7b2b0f24323c09dc756f8dc2908b372f9a17551fe915ff2171 File: ./contracts/modules/TransactionManager.sol SHA3: 0fc037872aa12e2f31e8bafaba27d476c66be6392416417c0f7a184ae6535a5b File: ./contracts/storage/IStorage.sol SHA3: 9e107a4621e077b4dbe7ddb0f559dae582f6d76d1f05c353912655c48852c4c0 File: ./contracts/storage/Storage.sol SHA3: 09161f3b9fe537f229b11f1b8477fc7bb87bf9e9c1dc7c97546338160d793a57 File: ./contracts/vault/BaseVault.sol SHA3: a2cad7cceeee26ec158114c5856960b83f814633e39fff73c430298a00859251

	File: ./contracts/vault/IVault.sol SHA3: ba22d24efeff11bedd0a554969b466470aa95aaa3f21fe8ca768c9f2d5770395 File: ./contracts/vault/VaultProxy.sol SHA3: ad17b273bcadc6a6ba48dd0e83097117e2feb7f339cd4061772a8dccdb226165

Third review scope

Repository	https://github.com/Kresus-Labs-Inc/vault-mpc/tree/feat/mpc
Commit	1049921
Functional Requirements	Readme document
Technical Requirements	Readme document + NatSpec in the code
Contracts	File: ./contracts/infrastructure/Factory.sol SHA3: 92e99c2fcb9e755104dfe2e281d34401fe2577a0aca0585148f78227fd21eb3d File: ./contracts/infrastructure/IModuleRegistry.sol SHA3: a2f02b2cb861790d97d250661ce6219d1c5284d6d8f7bea59f3973d8a7952eb8 File: ./contracts/infrastructure/KresusUpgrader.sol SHA3: 6bb7e44a95c444534d3a3e2f9782b81d7ae5d53f836abaed3199ace75e26e5b7 File: ./contracts/infrastructure/ModuleRegistry.sol SHA3: d423b6a7e60cdbe2282147fc0f2dc9afab1e4291aed3a4a51c46a404b05effeb File: ./contracts/modules/KresusModule.sol SHA3: 9e2eceb885fcac01d1aad1a151dfc7baeb01396f2ebe0e362167dd21b38d94d5 File: ./contracts/modules/KresusRelayer.sol SHA3: fd2cb06576c082fce9e11f419eae0ad300f03df760642e49292b1f72670d0b97 File: ./contracts/modules/SecurityManager.sol SHA3: 73e50c53df1cc1c060ce632eb7b5d58a87ae6bd7fc9039d9d68078da9b6caf9e File: ./contracts/modules/TransactionManager.sol SHA3: 0fc037872aa12e2f31e8bafaba27d476c66be6392416417c0f7a184ae6535a5b File: ./contracts/modules/common/BaseModule.sol SHA3: bdc72d8abe7f01866dbeb361e29cb07c7e152bb48ca5695be8e3f81ad8b2b574 File: ./contracts/modules/common/IModule.sol SHA3: afc9785569ce9cd91d36bc7e5e71dbbb3f9e9aada2b683df201fae64de3d55f7 File: ./contracts/modules/common/Utils.sol SHA3: 1e89156d919f172b3ba2129f0a01c81753ea2c0a0e3ed9339dc03b3d1f485f3c File: ./contracts/storage/IStorage.sol SHA3: 9e107a4621e077b4dbe7ddb0f559dae582f6d76d1f05c353912655c48852c4c0 File: ./contracts/storage/Storage.sol SHA3: 09161f3b9fe537f229b11f1b8477fc7bb87bf9e9c1dc7c97546338160d793a57 File: ./contracts/vault/BaseVault.sol SHA3: 6812c8b9392b40c295cf223676856d67dc345fb84781e718d5be5e62f868ea39

	File: ./contracts/vault/IVault.sol SHA3: 68bacc3549b753b561fea9527e43f2e52fa1887e34d3bccc56db35e391611ce3 File: ./contracts/vault/VaultProxy.sol SHA3: 14c224523fa6db172a94376016377dfe4dfaa98b29b9444f8b8652e16a7c1d20

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor Gas optimization. These issues won't have a significant impact on code execution but affect code quality

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- No Whitepaper is provided.
- Functional and technical requirements are provided and well described the project.
- The NatSpec comments are extensive.

Code quality

The total Code Quality score is **10** out of **10**.

- The code is well structured with separation of concerns.
- The development environment is configured, though it requires manually setting environment variables from the developer.

Test coverage

Code coverage of the project is **98.94%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Only a few negative cases are covered.
- Interactions by several users are not tested thoroughly.

Security score

As a result of the audit, the code contains **2** low severity issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.9**.

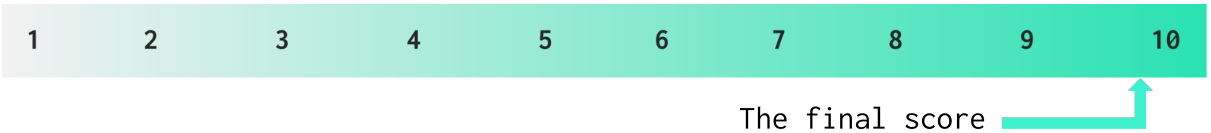

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
09 March 2023	31	6	7	0
3 April 2023	10	1	1	0

21 April 2023	2	0	0	0

Risks

- Front-end may receive zero address as a valid return value instead of an error.
- The systems rely heavily on low-level calls using function selectors, which may lead to vulnerabilities and unexpected behavior.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

System Overview

Kresus Vault is a blockchain-based secure storage facility for digital assets like cryptocurrency, NFTs, and other tokens. The system consists from the following contracts:

- *BaseModule* – an abstract contract for a module, containing mainly internal view functions.
- *Utils* - a library which contains helper methods dealing with signatures and function selectors.
- *KresusModule* - a contract representing a *module* i.e. a vault of valuables.
- *KresusRelayer* - an abstract contract with methods for executing transactions signed by accounts outside of Ethereum.
- *SecurityManager* - an abstract contract dealing with security features of the *KresusModule* i.e. guardians, lock, recovery etc.
- *TransactionManager* - an abstract contract for executing transactions in sequence or calling third-party contracts.
- *Storage* - a contract containing methods on voting, setting locks and assigning guardians.
- *Proxy* - a contract which delegated all calls to another fixed contract.
- *Factory* - a contract which deploys new vaults against a set deployer, which can be changed.

Privileged roles

- The *Module* in *Storage* contract can set locks, toggle votings, add/revoke guardians, add/revoke heirs.
- The *Deployer* of the *Factory* contract can set a new deployer address and deploy a new vault.
- The *KresusGuardian* in the *KresusModule* contract can revoke guardian from a specific vault, lock or unlock vault. When voting is enabled - KresusGuardian signature is also required to revoke guardian.
- The *Guardian* in the *KresusModule* contract can revoke guardian. During voting guardian signature is also required to enable ERC1155 token receiver, add new module, add new guardian, transfer vault ownership, revoke guardian, lock vault, perform multicall, toggle voting and set time delay.
- The *Owner*'s signature in the *KresusModule* can be necessary for certain types of transactions.
- The *Module* in the *BaseVault* can authorize modules, enable static calls, set an owner, or invoke a generic transaction.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Passed
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Passed
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Passed
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP Standards Violation	EIP	EIP standards should not be violated.	Passed
Assets Integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant
Token Supply Manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the Customer.	Not Relevant
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style Guide Violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

H01. Requirements Violation

Storage contract allows authorized modules to add and revoke heir, but this functionality is not implemented in the current system, which makes *vaultStorage[_vault].heir* redundant or is a sign of non-finalized code.

Path:
./contracts/storage/Storage.sol : addHeir(), revokeHeir()

Recommendation: Implement add/remove *heir* logic in the *BaseModule* contract or remove this functionality.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

H02. Undocumented Behavior; Requirements Violation

The correctness of the *getRequiredSignatures* function could not be verified because of missing documentation.

During the second review, we found mismatch which should be fixed: *removeGuardian* functionality should be available for owner only when voting is not enabled, but existing implementation allows Owner, or Guardian, or KWG to execute it.

addModule functionality is not documented and cannot be verified.

executeBequeathal functionality, according to the docs, should not be added to the queue.

According to the documentation, owner should be able to cancel queued request, but *getCancelRequiredSignatures* function does not contain *SecurityManager.executeBequeathal.selector* check, so this functionality is not allowed to the owner, which violates the requirements.

Path:
./contracts/modules/KresusModule.sol: getRequiredSignatures(), getCancelRequiredSignatures();

Recommendation: Add documentation for *addModule*. Fix code or update documentation for *removeGuardian* and *executeBequeathal*. Add *executeBequeathal.selector* to the *getCancelRequiredSignatures*.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: 1049921)

H03. Requirements Violation

According to NatSpec for the *addGuardian* function - “The first guardian is added immediately. All following additions must be confirmed by calling the confirmGuardianAddition() method.”

The existing implementation allows the addition of only a single guardian and *confirmGuardianAddition* function does not exist.

Path:
./contracts/vault/BaseVault.sol

Recommendation: Update NatSpec or implement *confirmGuardianAddition* function.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: e3b7611)

H04. Requirements Violation

The existing implementation does not allow revoking an existing guardian - the *confirmGuardianRevokation* function does not exist.

Path:
./contracts/vault/BaseVault.sol

Recommendation: Update NatSpec or implement *confirmGuardianRevokation* function.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: e3b7611)

H05. Requirements Violation

In the *cancel()* function, it is mentioned that it is used to "cancels a transaction which was queued", but not all transactions in queue can be canceled.

The *getCancelRequiredSignatures()* function does not return needed Signature enum for all the possible queueable transactions.

In result *addModule()*, *unlock()*, *setTimeDelay()* cannot be canceled.

Path:
./contracts/modules/KresusRelayer.sol : cancel()

Recommendation: Implement canceling functionality for all types of transactions.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Mitigated (Certain operations cannot be canceled as part of business requirement.)

H06. EIP Standard Violation

The *isValidSignature* is not implemented in the *TransactionManager*, despite the contract responding to the selector in *supportsStaticCall*(). This can be a result of *ERC1271_IS_VALID_SIGNATURE* not being implemented properly.

Path:
./contracts/modules/TransactionManager.sol

Recommendation: Implement the functionality related to *ERC1271_IS_VALID_SIGNATURE*.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: e3b7611)

H07. Undocumented Behavior

Inside the *setTimeDelay()* function, in case of *VotingEnabled == true*, the function returns *false* for queue transactions, where it returns *true* in case of *VotingEnabled == false*. This is probably an unintended behavior.

Path:
./contracts/modules/KresusModule.sol : getRequiredSignatures()

Recommendation: Return false for queue translations in case of *VotingEnabled == false* or mention the original behavior in the documentation.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: e3b7611)

Medium

M01. Undocumented Behavior

There is no documentation regarding when and where the Received event should be emitted. For example, in the current form of how the Factory contract deploys the Proxy contract, emitting Received in the init() function appears to be redundant.

Directly emitting the Received event in the receive() function in the Proxy contract is a bad practice, as the call to the receive() function should be forwarded to the BaseVault implementation and the event should be emitted inside the receive() function there.

Path:
./contracts/vault/BaseVault.sol

Recommendation: Rethink where the Received event should be emitted and change the code accordingly.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

M02. Requirements Violation

According to the NatSpec documentation, the function *enabled()* *"Returns the module responsible for a static call redirection."*. However, the code returns the state variable staticCallExecutor in case of happy-path or a zero-address otherwise.

Path:
./contracts/vault/BaseVault.sol : enabled()

Recommendation: Change the documentation or the logic of the function.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

M03. Sign of Non-Finalized Code

BaseVault provides modules an API to add or remove modules, but only add functionality is implemented (in KresusModule) and there is no possibility to remove the previously added module.

Path:
./contracts/vault/BaseVault.sol : authoriseModule()

Recommendation: Add the possibility to remove modules.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: e3b7611)

M04. Unclear Logic

TransactionManager uses *enableDefaultStaticCalls* to enable static calls (for *ERC1271_IS_VALID_SIGNATURE* and *ERC721_RECEIVED*).

BaseVault *enableStaticCall* ignores method signature and enables static calls for the whole contract. So there is no need to call it multiple times.

It also violates the NatSpec from the parent interface IVoult, which states: "Enables a static method by specifying the target module to which the call must be delegated."

The logic might be incomplete.

Path:
./contracts/modules/TransactionManager.sol : enableDefaultStaticCalls()

Recommendation: Check if implemented logic is correct. Clarify requirements or remove duplicated calls.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: e3b7611)

M05. Unclear Behavior

uint256 nonceBlock = (_nonce & 0xffffffffffffffffffffffffffffffff00000000000000000000000000000000) >> 128; will return value bigger than 0 only when *_nonce* is bigger than *1 * 10^40*. Such a big nonce looks unclear and redundant.

Path: ./contracts/modules/KresusRelayer.sol : checkAndUpdateUniqueness()

Recommendation: Clarify requirements, remove this validation, or construct nonce from block.number and block.timestamp.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Mitigated (This is a part of business logic.)

M06. Possible False-Negative Validation

The *validateSignatures* function has multiple places, where two signatures are required to perform an action, but this check requires a stick order of signing: owner is first, guardian or Kresus guardian is the second.

In case the guardian signs it first - this check would fail, despite the fact that the requirement is correct - both addresses signed the message.

Path:
./contracts/modules/KresusModule.sol : validateSignatures()

Recommendation: Add reverse order of signatures check as well.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Mitigated (Transaction must be initiated by the owner.)

M07. Interface Mismatch

Interface *IModuleRegistry* does not match *ModuleRegistry* contract. Implementation does not contain *registerUpgrader*, *deregisterUpgrader*, *recoverToken*, *upgraderInfo*, *isRegisteredUpgrader* functions. Interface declares *bytes32* parameters, when contract implements them as *string*.

Path:
./contracts/infrastructure/IModuleRegistry.sol

Recommendation: Match interface with contract.

Found in: e3b7611b27715983868526ae6dc858ba3a7f5933

Status: Fixed (Revised commit: 1049921)

Low

L01. Missing Zero Address Validation

Address parameters are being used without checking against the possibility of 0x0. This can lead to unwanted external calls to 0x0.

During the second review, we identified the same issue in the *init()* function of *BaseVault*.

Paths:
./contracts/modules/KresusModule.sol : constructor()
./contracts/modules/KresusRelayer.sol : constructor(), execute()
./contracts/vault/BaseVault.sol : init()
./contracts/Factory.sol : constructor(), changeDeployer()

Recommendation: Implement zero address checks.

Found in: 38db3f4

Status: Fixed (Revised commit: 1049921)

L02. Unindexed Events

Having indexed parameters in the events makes it easier to search for these events using indexed parameters as filters.

Paths:
./contracts/vault/BaseVault.sol : OwnerChanged
./contracts/modules/common/BaseModule.sol : ModuleCreated
./contracts/Factory.sol : NewVaultDeployed

Recommendation: Use the “indexed” keyword to the event parameters.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L03. Missing Events

Events for critical state changes should be emitted for tracking things off-chain.

Paths:
./contracts/vault/BaseVault.sol : receive(), authoriseModule(), enableStaticCall(), init()
./contracts/Factory.sol : constructor(), changeDeployer()

Recommendation: Create and emit related events.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L04. Unused Constants

Unused constants and variables should be removed from the contracts. Unused constants are allowed in Solidity and do not pose a direct security issue.

It is best practice to avoid them as they can cause an increase in computations (and unnecessary Gas consumption) and decrease readability.

Paths:
./contracts/modules/common/BaseModule.sol : ETH_TOKEN
./contracts/modules/common/Utils.sol : ERC20_TRANSFER, ERC20_APPROVE, ERC721_SET_APPROVAL_FOR_ALL, ERC721_TRANSFER_FROM, ERC721_SAFE_TRANSFER_FROM, ERC721_SAFE_TRANSFER_FROM_BYTES, ERC1155_SAFE_TRANSFER_FROM, OWNER_SIG

Recommendation: Remove unused constants.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L05. State Variables Can Be Declared Immutable

Compared to regular state variables, the Gas costs of constant and immutable variables are much lower. Immutable variables are evaluated once at construction time, and their value is copied to all the places in the code where they are accessed.

Path:
./contracts/modules/KresusRelayer.sol : refundAddress

Recommendation: Declare mentioned variables as immutable.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L06. Style Guide Violation: Variable Naming

Unlike types, variables must begin with a lowercase letter. Naming a variable with an uppercase letter may confuse developers and auditors, as it is a convention reserved for types.

All variables must be written in camel case.

During the second review, we identified “Storage” naming in BaseModule constructor.

Paths:
./contracts/modules/common/BaseModule.sol : Storage
./contracts/modules/KresusModule.sol : line 81
./contracts/modules/KresusRelayer.sol : line 23, line 33

Recommendation: Rename state variable *Storage* to *storage*, *arrayindex* to *arrayIndex*, *VotingEnabled* to *votingEnabled*, *SignatureRequirement* to *signatureRequirement.*

Found in: 38db3f4

Status: Fixed (Revised commit: 1049921)

L07. Style Guide Violation: Incorrect Ordering

Enum and struct definitions must be placed before state variables.

External functions cannot go after external view functions.

Paths:
./contracts/modules/common/BaseModule.sol : Signature
./contracts/modules/KresusRelayer.sol : RelayerConfig
./contracts/modules/TransactionManager.sol : Call
./contracts/storage/Storage.sol : StorageConfig
./contracts/vault/IVault.sol : setOwner()

Recommendation: Follow the official Solidity guidelines: https://docs.soliditylang.org/en/v0.8.13/style-guide.html

Use *solhint* static analyzer with ordering *rule* enabled in order to get hints for layout order.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L08. State Variables Default Visibility

The explicit visibility makes it easier to catch incorrect assumptions about who can access the variable.

Paths:
./contracts/storage/Storage.sol : vaultStorage
./contracts/Factory.sol : deployer, baseVaultImpl

Recommendation: Specify variables as public, internal, or private. Explicitly define visibility for all state variables.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L09. Typos

Any typos encountered in the code or documentation should be addressed.
During the second review, there is still typo in “initilization”

Paths:
./contracts/modules/SecurityManager.sol : OwnershipTransfered, line :106, line :142;
./contracts/vault/BaseVault.sol : line :40;

Recommendation: Rename:

OwnershipTransfered to *OwnershipTransferred*,
votig to *voting*,
Revokation to *Revocation,*
adress to *address,*
queuedActionExectionTime to *queuedActionExecutionTime,*
initilization to *initialization*

Found in: 38db3f4

Status: Fixed (Revised commit: 1049921)

L10. Wrong Data Type in Event

The event *ModuleCreated* uses bytes32 for the field *name*. While technically bytes and strings are internally the same data, however, since events can be used only on the client code, it makes more sense to store this data as a string.

Path:
./contracts/modules/common/BaseModule.sol : ModuleCreated

Recommendation: Change data type of *name* to *string*.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L11. Misleading Event

Events should provide useful information for the users or client apps. The event *ModuleCreated* has the only field *name* which can contain arbitrary data unrelated to the emitter contract.

Path:
./contracts/modules/common/BaseModule.sol : ModuleCreated

Recommendation: Consider adding useful information to the event e.g. the created contract’s address.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L12. Conversion From Bytes32 to Uint256

The function expects a parameter *_timeDelay* in the form of bytes32. Later the parameter is converted to uint256. This can potentially lead to wrong conversion, since bytes in Solidity are big-endian, while integers are little-endian.

Path:
./contracts/modules/KresusModule.sol : init()

Recommendation: Change the parameter *_timeDelay* to uint256. If bytes32 is used by design, it should be mentioned in the documentation.

Found in: 38db3f4

Status: Mitigated (As Only Kresus can deploy the contract and pass this init value, the possibility of an attacker taking advantage of this is low.)

L13. Missing Balance Validation

The function *invoke()* sends the value defined by the function parameter. If the contract does not have enough balance, this may lead to an unclear transaction error.

Path:
./contracts/vault/BaseVault.sol : invoke()

Recommendation: Add balance validation check before sending native tokens.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L15. Unused Code

The contract *SecurityManager* contains a modifier *onlyGuardianOrSelf* which is never used throughout the project.

Path:
./contracts/modules/SecurityManager.sol : onlyGuardianOrSelf()

Recommendation: Remove the modifier *onlyGuardianOrSelf*.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L16. Boolean Equality

Boolean constants can be used directly and do not need to be compared to true or false.

Paths:
./contracts/vault/BaseVault.sol : authoriseModule()

./contracts/modules/KresusRelayer.sol : execute()
./contracts/modules/KresusModule.sol : getRequiredSignatures()

Recommendation: Remove boolean equality.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: e3b7611)

L17. Function Name - Functionality Mismatch

addGuardian function makes it clear from its name that it is possible to add multiple guardians, but this function actually sets a single guardian.

The same is applicable to the *addHeir* function.

Path:
./contracts/storage/Storage.sol : addGuardian(), addHeir()

Recommendation: Rename this function to *setGuardian*.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: e3b7611)

L18. Code Duplication

SecurityManager.lock.selector check is implemented twice in the *getRequiredSignatures* function, which is redundant and could be removed.

Path:
./contracts/modules/KresusModule.sol: getRequiredSignatures()

Recommendation: Remove code duplication.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: e3b7611)

L19. Best Practices Violation

Proxy implementation looks like a custom non-standard modification of EIP-897 standard, as *receive()* function does not delegate a call to *_implementation*, but just emits an event.

Path:
./contracts/vault/Proxy.sol

Recommendation: Use Proxy contract from OpenZeppelin.

Found in: 38db3f4d089054820f6734ca20647748eaa475ac

Status: Fixed (Revised commit: e3b7611)

L20. CEI Pattern Violation

The Checks-Effects-Interactions pattern is violated. During the function *invoke()*, an event is emitted after the external call. This is partly remediated by the fact that only a module can call this function.

Path:
./contracts/vault/BaseVault.sol : invoke()

Recommendation: Emit the event before the external call.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L21. Redundant Imports

Unused imports should be removed from the contracts. Unused imports are allowed in Solidity and do not pose a direct security issue. It is best practice to avoid them as they can decrease readability.

During the second review, we identified that KresusRelayer related imports were not deleted.

Paths:
./contracts/modules/KresusRelayer.sol : Utils, IStorage (not fixed)
./contracts/modules/SecurityManager.sol : SafeCast, Utils, KresusRelayer

Recommendation: Remove the redundant imports.

Found in: 38db3f4

Status: Fixed (Revised commit: 1049921)

L22. Gas Optimisation

`Storage.getTimeDelay(_vault)` is read twice in the KresusRelayer in case of `if(queue == true)`. This increases Gas cost of the function execution.

Path:
./contracts/modules/KresusRelayer.sol : execute()

Recommendation: Store the value in a local variable.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L23. Gas Optimisation

The check `require(isActionQueued(_vault, _actionHash), "KR: Invalid hash")` is redundant in the `cancelAll()` function, as the execution logic is going in loop and takes `_actionHash` from the queue.

Path:
./contracts/modules/KresusRelayer.sol : cancelAll()

Recommendation: Remove the redundant check.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611

L24. Commented Code

The projects contain commented-out blocks of code:

```
// KresusRelayer(_vault).cancelAll(_vault);
// import "@openzeppelin/contracts/utils/math/SafeCast.sol";
```

Commented code is a sign of unfinished code and should be removed from the finalized code.

Paths:
./contracts/modules/SecurityManager.sol : lock()
./contracts/modules/TransactionManager.sol

Recommendation: Remove the commented code blocks.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L25. Redundant Code

The `enableERC1155TokenReceiver()` function does not add any business value and has no use. This can probably be an unfinalized code.

Path:
./contracts/modules/TransactionManager.sol :
enableERC1155TokenReceiver()

Recommendation: Remove the mentioned code block.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L26. Style Guide Violation - Variable Naming

The enum elements `OwnerandGuardian` and `OwnerandGuardianOrOwnerandKWG` break the camel-case naming convention by not having an uppercase *And*. This makes reading and understanding code more difficult.

Path:
./contracts/modules/common/BaseModule.sol : Signature

Recommendation: Rename the enum elements to follow the camel-case convention.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L27. Redundant Override

The contract *BaseVault* contains redundant overrides: owner, authorised, modules, authoriseModule(), enabled(), enableStaticCall(), setOwner().

Path:
./contracts/vault/BaseVault.sol

Recommendation: Remove the redundant *override* modifiers.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L28. Usage of Uint instead of Uint256

Although *uint* is no more than an alias of *uint256* in Solidity, it is generally recommended to type this data type as *uint256* to avoid confusion and make intention clear.

Paths:
./contracts/modules/KresusRelayer.sol: refund()
./contracts/modules/TransactionManager.sol : multiCallWithApproval()
./contracts/modules/common/Utils.sol : recoverSigner()
./contracts/vault/BaseVault.sol : Invoked, Received, invoke()

Recommendation: Replace *uint* declarations with *uint256*.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L29. Recommendation

Adding a *constructor()* to the *BaseVault* with *owner = msg.sender* will prevent implementation contract initialization.

Path:
./contracts/vault/BaseVault.sol

Recommendation: Add a constructor to the *BaseVault* contract.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L30. Gas Optimisation

The check *require(authorised[_modules[i]] == false, "BW: module is already added")* is redundant as there already was a check *modules == 0*.

The redundant check increases Gas spending.

Path: ./contracts/vault/BaseVault.sol : init()

Recommendation: Remove the redundant check.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L31. Wrong Strict Check

The strict check `_newTimeDelay > MIN_TIME_DELAY` is likely to be non-strict, i.e. `newTimeDelay >= MIN_TIME_DELAY`.

Path:
./contracts/storage/Storage.sol : setTimeDelay()

Recommendation: Replace strict-check with non-strict.

Found in: 38db3f4

Status: Fixed (Revised commit: e3b7611)

L32. Unused Constants

Unused constants and variables should be removed from the contracts. Unused constants are allowed in Solidity and do not pose a direct security issue.

It is best practice to avoid them as they can cause an increase in computations (and unnecessary Gas consumption) and decrease readability.

Path:
./contracts/modules/KresusModule.sol : NAME

Recommendation: Remove unused constants.

Found in: e3b7611

Status: Fixed (Revised commit: 1049921)

L33. Missing Zero Address Validation

Address parameters are being used without checking against the possibility of 0x0. This can lead to unwanted external calls to 0x0.

Paths:
./contracts/modules/common/SecurityManager.sol : validateNewOwner()
./contracts/infrastructure/KresusUpgrader.sol : constructor()

Recommendation: Implement zero address checks.

Found in: e3b7611

Status: Reported

L34. Missing Events

Events for critical state changes should be emitted for tracking things off-chain.

Path:
./contracts/vault/BaseVault.sol : setOwner()

Recommendation: Create and emit related events.

Found in: e3b7611

Status: Reported

L35. State Variables Can Be Declared Immutable

Compared to regular state variables, the Gas costs of constant and immutable variables are much lower. Immutable variables are evaluated once at construction time, and their value is copied to all the places in the code where they are accessed.

Path:
./contracts/infrastructure/KresusUpgrader.sol : registry

Recommendation: Declare mentioned variables as immutable.

Found in: e3b7611

Status: Fixed (Revised commit: 1049921)

L36. Usage of Uint instead of Uint256

Although *uint* is no more than an alias of *uint256* in Solidity, it is generally recommended to type this data type as *uint256* to avoid confusion and make intention clear.

Path:
./contracts/infrastructure/ModuleRegistry.sol : isRegisteredModule

Recommendation: Replace *uint* declarations with *uint256*.

Found in: e3b7611

Status: Fixed (Revised commit: 1049921)

L37. State Variables Default Visibility

The explicit visibility makes it easier to catch incorrect assumptions about who can access the variable.

Paths:
./contracts/modules/KresusRelayer.sol : BLOCKBOUND
./contracts/vault/VaultProxy.sol : implementation

Recommendation: Specify variables as public, internal, or private. Explicitly define visibility for all state variables.

Found in: e3b7611

Status: Fixed (Revised commit: 1049921)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in 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).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

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 recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

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 vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.