ABDK CONSULTING

SMART CONTRACT AUDIT

Proxima Capital

Vault

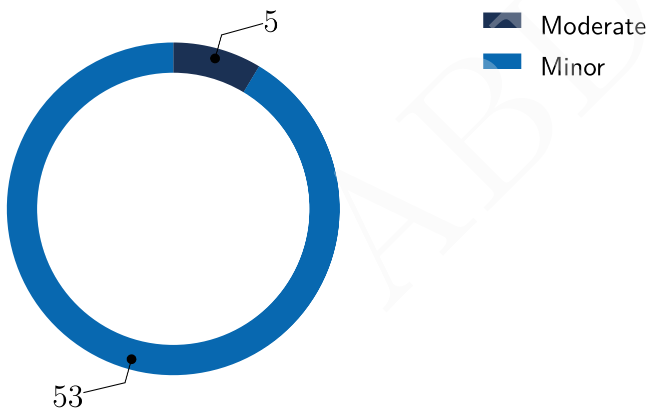
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
24th August 2022

We've been asked to review 9 files in a Github repository. We found 5 moderate, and a few less important issues.



Findings

ID	Severity	Category	Status
CVF-1	Minor	Procedural	Fixed
CVF-2	Minor	Procedural	Fixed
CVF-3	Minor	Bad naming	Fixed
CVF-4	Minor	Suboptimal	Info
CVF-5	Minor	Suboptimal	Info
CVF-6	Minor	Suboptimal	Info
CVF-7	Minor	Bad naming	Info
CVF-8	Minor	Suboptimal	Fixed
CVF-9	Moderate	Unclear behavior	Info
CVF-10	Minor	Procedural	Fixed
CVF-11	Minor	Procedural	Info
CVF-12	Minor	Documentation	Fixed
CVF-13	Minor	Procedural	Fixed
CVF-14	Minor	Suboptimal	Fixed
CVF-15	Minor	Suboptimal	Info
CVF-16	Minor	Suboptimal	Info
CVF-17	Minor	Suboptimal	Fixed
CVF-18	Minor	Suboptimal	Fixed
CVF-19	Minor	Suboptimal	Fixed
CVF-20	Minor	Suboptimal	Fixed
CVF-21	Minor	Suboptimal	Fixed
CVF-22	Minor	Bad datatype	Fixed
CVF-23	Minor	Suboptimal	Fixed
CVF-24	Minor	Flaw	Fixed
CVF-25	Minor	Readability	Fixed
CVF-26	Minor	Suboptimal	Info
CVF-27	Minor	Suboptimal	Fixed

ID	Severity	Category	Status
CVF-28	Moderate	Unclear behavior	Info
CVF-29	Minor	Suboptimal	Info
CVF-30	Minor	Procedural	Info
CVF-31	Minor	Readability	Info
CVF-32	Minor	Procedural	Info
CVF-33	Minor	Suboptimal	Info
CVF-34	Moderate	Unclear behavior	Info
CVF-35	Minor	Suboptimal	Info
CVF-36	Minor	Procedural	Fixed
CVF-37	Minor	Procedural	Fixed
CVF-38	Minor	Suboptimal	Info
CVF-39	Minor	Suboptimal	Info
CVF-40	Minor	Procedural	Fixed
CVF-41	Minor	Unclear behavior	Info
CVF-42	Minor	Suboptimal	Info
CVF-43	Moderate	Overflow/Underflow	Info
CVF-44	Moderate	Overflow/Underflow	Info
CVF-45	Minor	Procedural	Fixed
CVF-46	Minor	Procedural	Fixed
CVF-47	Minor	Procedural	Fixed
CVF-48	Minor	Suboptimal	Info
CVF-49	Minor	Procedural	Fixed
CVF-50	Minor	Readability	Info
CVF-51	Minor	Bad datatype	Fixed
CVF-52	Minor	Bad naming	Info
CVF-53	Minor	Bad naming	Info
CVF-54	Minor	Bad naming	Info
CVF-55	Minor	Bad naming	Info
CVF-56	Minor	Bad datatype	Fixed
CVF-57	Minor	Bad datatype	Fixed

ID	Severity	Category	Status
CVF-58	Minor	Bad datatype	Fixed

Contents

1 Document properties **8**

2 Introduction **9**
2.1 About ABDK . . . 9
2.2 Disclaimer . . . 9
2.3 Methodology . . . 9

3 Detailed Results **11**
3.1 CVF-1 . . . 11
3.2 CVF-2 . . . 11
3.3 CVF-3 . . . 11
3.4 CVF-4 . . . 12
3.5 CVF-5 . . . 12
3.6 CVF-6 . . . 12
3.7 CVF-7 . . . 13
3.8 CVF-8 . . . 13
3.9 CVF-9 . . . 14
3.10 CVF-10 . . . 14
3.11 CVF-11 . . . 14
3.12 CVF-12 . . . 15
3.13 CVF-13 . . . 16
3.14 CVF-14 . . . 16
3.15 CVF-15 . . . 17
3.16 CVF-16 . . . 17
3.17 CVF-17 . . . 18
3.18 CVF-18 . . . 18
3.19 CVF-19 . . . 18
3.20 CVF-20 . . . 19
3.21 CVF-21 . . . 19
3.22 CVF-22 . . . 19
3.23 CVF-23 . . . 20
3.24 CVF-24 . . . 20
3.25 CVF-25 . . . 20
3.26 CVF-26 . . . 21
3.27 CVF-27 . . . 21
3.28 CVF-28 . . . 21
3.29 CVF-29 . . . 22
3.30 CVF-30 . . . 22
3.31 CVF-31 . . . 22
3.32 CVF-32 . . . 23
3.33 CVF-33 . . . 23
3.34 CVF-34 . . . 23
3.35 CVF-35 . . . 24
3.36 CVF-36 . . . 24
3.37 CVF-37 . . . 24

3.38 CVF-38 24
3.39 CVF-39 25
3.40 CVF-40 25
3.41 CVF-41 26
3.42 CVF-42 27
3.43 CVF-43 28
3.44 CVF-44 28
3.45 CVF-45 28
3.46 CVF-46 29
3.47 CVF-47 29
3.48 CVF-48 29
3.49 CVF-49 30
3.50 CVF-50 30
3.51 CVF-51 30
3.52 CVF-52 31
3.53 CVF-53 31
3.54 CVF-54 32
3.55 CVF-55 32
3.56 CVF-56 32
3.57 CVF-57 33
3.58 CVF-58 33

1 Document properties

Version

Version	Date	Author	Description
0.1	August 23, 2022	A. Zveryanskaya	Initial Draft
0.2	August 23, 2022	D. Khovratovich	Minor revision
1.0	August 24, 2022	A. Zveryanskaya	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.
We have reviewed the contracts at repository:

- connections/BaseLogic.sol
- interfaces/IERC20.sol
- utils/Call.sol
- utils/Create2.sol
- utils/Create2Deployer.sol
- utils/RecoverableOwnable.sol
- Identity.sol
- IVault.sol
- Vault.sol

The fixes were provided in a new commit.

2.1 About ABDK

ABDK Consulting, established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function. The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

3 Detailed Results

3.1 CVF-1

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** RecoverableOwnable.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.
Recommendation Consider specifying like "^0.8.0".

Listing 1:

```
pragma solidity 0.8.15;
```

3.2 CVF-2

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** RecoverableOwnable.sol

Description This interface should be defined in a file named "IRecoverableOwnable.sol".

Listing 2:

```
interface IRecoverableOwnable
```

3.3 CVF-3

- **Severity** Minor
- **Category** Bad naming
- **Status** Fixed
- **Source** RecoverableOwnable.sol

Description The. name is confusing. One could think that this is the time when recovery happened, while actually this is the time recovery is possible after.
Recommendation Consider renaming.
Client Comment Renamed 'recoveryTimestamp' to 'disputeDeadline'.

Listing 3:

```
function recoveryTimestamp() external view returns (uint256);
```

3.4 CVF-4

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** RecoverableOwnable.sol

Description The "previousOwner" parameter is redundant, as its value could be derived from the previous event.
Recommendation Consider renaming.
Client Comment This signatures matches Ownable.sol OwnershipTransferred https://github.com/OpenZeppelin/openzeppelin-contracts/blob/ec825d8999538f110e572605dc56ef7bf44cc574/contracts/access/Ownable.sol, we will retain it

Listing 4:

```
event OwnerTransferred(address indexed previousOwner, address
  ↪ indexed newOwner);
```

3.5 CVF-5

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** RecoverableOwnable.sol

Description This code proceeds even if the new owner is the same as the old one.
Recommendation Consider reverting in this case.
Client Comment This is an unecessary check as the consequence is benign, will not add.

Listing 5:

```
function nominateOwner(address aNewOwner) external onlyOwner
```

3.6 CVF-6

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** RecoverableOwnable.sol

Description The "previousRecoverer" parameter is redundant as its value could be derived from the previous event.
Client Comment This signatures is inspired by Ownable.sol OwnershipTransferred https://github.com/OpenZeppelin/openzeppelin-contracts/blob/ec825d8999538f110e572605dc56ef7bf44cc574/contracts/access/Ownable.sol, we will retain it

Listing 6:

```
event RecovererTransferred(address indexed previousRecoverer,
  ↪ address indexed newRecoverer);
```

3.7 CVF-7

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** RecoverableOwnable.sol

Description The parameter named "deadline" is confusing, as one could thing that the value of this parameter is the recovery deadline, i.e. the time which recovery is not possible after. while actually it is the time recovery is not possible before.
Recommendation Consider renaming.
Client Comment Will keep 'deadline' because we now call the recovery timestamp the 'disputeDeadline'.

Listing 7:

```
174 event RecoveryStarted(address indexed user, uint256 indexed
      ↪ deadline);
```

3.8 CVF-8

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** RecoverableOwnable.sol

Description This event is emitted even if there were no recovery started. Consider reverting in such a case.

Listing 8:

```
206 emit RecoveryStopped(msg.sender, recoveryTimestamp);
```

3.9 CVF-9

- **Severity** Moderate
- **Status** Info
- **Category** Unclear behavior
- **Source** RecoverableOwnable.sol

Description After this point the recoverer and the owner will be the same address, thus there will be no redundancy and the ownership will effectively become non-recoverable for some time.
Recommendation Consider allowing the recoverer to nominate during recovery a new owner other than the recoverer himself.
Client Comment The recoverer is often a lower-security, or infrequently accessed wallet. After recovery, the recoverer will need to: - Set new long-term owner - Set new recoverer, or continue itself
Plus, at deployment no recoverer is specified. Additionally, each owner must accept ownership ensuring no transfer to inaccessible addresses is possible. For these reasons we will not fix.

Listing 9:

```
_transferOwner(recoverer);
```

3.10 CVF-10

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** Vault.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.
Recommendation Consider specifying like "^0.8.0".

Listing 10:

```
pragma solidity 0.8.15;
```

3.11 CVF-11

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Vault.sol

Description We didn't review these files.

Listing 11:

```
import { SafeTransferLib } from "solmate/utils/SafeTransferLib.
    ↪ sol";
import { ERC20 } from "solmate/tokens/ERC20.sol";
```

3.12 CVF-12

- **Severity** Minor
- **Category** Documentation
- **Status** Fixed
- **Source** Vault.sol

Description This comment doesn't seem to be related to the code.
Recommendation Consider removing it.

Listing 12:

```
// nb: code blocks extend from 4 to col 80, in-line with
   ↪ markdown standards
// nb: code block name starts at 37:
//
//   offset = 4
//   length = (80 - offset)
//   text_len = 12
//
//   start_index = offset + length / 2 - text_len / 2
//   start_index = 36
//
// Due to how editors count columns (starting at 1), this means
   ↪ the first
// character will be placed at col 37, i.e. col 36 will still
   ↪ be whitespace.
//
// If your title is less than 12 chars, it will be left of
   ↪ center.
// If your title is more than 12 chars, it will be right of
   ↪ center
// This is fine, do not attempt to center your title, just
   ↪ start typing at
// col 37 and beyond.
```

3.13 CVF-13

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** Vault.sol

Description These event declarations should be moved to the “IVault” interface.

Listing 13:

```
event KeeperUpdated(address indexed keeper, bool enabled);

event ApprovedSelectorUpdated(bytes4 indexed selector, bool enabled);

event ApprovedLogicUpdated(address indexed logic, bool enabled);

event ConnectionUpdated(string indexed name, bytes config);

event ApprovalUpdated(IERC20 indexed token, address indexed recipient, uint256 approval);
```

3.14 CVF-14

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** Vault.sol

Description It would be more efficient to split each of these events into two events, one for enabled=true and another for enabled=false.

Listing 14:

```
event KeeperUpdated(address indexed keeper, bool enabled);

event ApprovedSelectorUpdated(bytes4 indexed selector, bool enabled);

event ApprovedLogicUpdated(address indexed logic, bool enabled);
```

3.15 CVF-15

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** Vault.sol

Description The "== true" part is redundant.
Recommendation Consider removing it.
Client Comment This is a stylistic preference to verbosely extract the conditions in all control flow. Will keep this in mind for new projects with new style guides.

Listing 15:

```
if (aEnabled == true) {

require(_keepers.contains(msg.sender) == true, "VAULT:
    ↪ NOT_KEEPER");

if (aEnabled == true) {

if (aEnabled == true) {

            require(abi.decode(lData, (bool)) == true, "
                ↪ TRANSFER_FAILED");
```

3.16 CVF-16

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** Vault.sol

Description It would be more efficient to have a single array of structs with two fields, rather than two parallel arrays.
Client Comment This would affect the 'yul' code in 'execute' which was deemed not worth the risk/work. It may also increase the gas cost of accessing the bye config by index.

Listing 16:

```
bytes[] private _connections;

string[] private _connectionNames;
```

3.17 CVF-17

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** Vault.sol

Description Binary “and” would be more efficient.

Listing 17:

```
require(aConfig.length % 32 == 0, "VAULT: UNALIGNED_CONFIG");
```

3.18 CVF-18

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** Vault.sol

Description It should be “>=” instead of “ ” in this comment.

Listing 18:

```
// because this branch requires that lConnectionId != 0 and
// lConnectionId is a uint256, we can be sure that
   ↪ lConnectionId is
// > 1, ergo lConnectionId - 1 must be > 0
```

3.19 CVF-19

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** Vault.sol

Description The same event is emitted for adding a new connection and updating an existing connection.
Recommendation Consider emitting different events.

Listing 19:

```
emit ConnectionUpdated(aName, aConfig);
```

3.20 CVF-20

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** Vault.sol

Description The “lRecipient” variable is redundant.
Recommendation Consider using “lRecipients[i]” instead.

Listing 20:

```
(address lRecipient , ) = lMap.at(i);
```

3.21 CVF-21

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** Vault.sol

Description The expression “_recipients[aToken]” is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 21:

```
_recipients[aToken].remove(aRecipient);

if (_recipients[aToken].length() == 0) {
```

3.22 CVF-22

- **Severity** Minor
- **Category** Bad datatype
- **Status** Fixed
- **Source** Vault.sol

Description The type of this variable should be “Identity[]”.

Listing 22:

```
address[] private _identities;
```

3.23 CVF-23

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** Vault.sol

Description Brackets around the subtraction are redundant.
Recommendation Consider removing them.

Listing 23:

```
lRecipients.set(aRecipient, (lApproval - aAmount));
```

3.24 CVF-24

- **Severity** Minor
- **Category** Flaw
- **Status** Fixed
- **Source** Vault.sol

Description Here an implicit underflow check introduced by compiler is used to enforce a business-level constraint. This is a bad practice that makes code more fragile.
Recommendation Consider using an explicit "require" statement to ensure that the current allowance is enough.

Listing 24:

```
lRecipients.set(aRecipient, (lApproval - aAmount));
```

3.25 CVF-25

- **Severity** Minor
- **Category** Readability
- **Status** Fixed
- **Source** Vault.sol

Description This makes code harder to read.
Recommendation Consider performing increment inside the "for" statement.

Listing 25:

```
unchecked {
    ++i;
}
```

3.26 CVF-26

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Vault.sol

Description This immutable variable could be turned into a constant.
Client Comment Leaving as is because Identity stores the Vault address immutably, makings its codehash dependent on the address of the deployer/vault. This creates non-trivial devops overhead, so we prefer to pay the extra gas to deploy an unused entity.

Listing 26:

```
bytes32 immutable private _identityCodehash = address(new
  ↪ Identity{salt: bytes32(type(uint256).max)}()).codehash;
```

3.27 CVF-27

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Vault.sol

Description This line does nothing.
Recommendation Consider removing it.
Client Comment Removed receive() from Vault.

Listing 27:

```
return;
```

3.28 CVF-28

- **Severity** Moderate
- **Status** Info
- **Category** Unclear behavior
- **Source** Vault.sol

Description This could load bytes after "aInput".
Client Comment We assume this to be impossible, see the following test cases.

Listing 28:

```
lSelector := calldataload(aInput.offset)
```

3.29 CVF-29

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** Vault.sol

Description Shift would be more efficient than division.

Listing 29:

```
let config_end := add(config_body, div(config_size, 32))
```

3.30 CVF-30

- **Severity** Minor
- **Category** Procedural
- **Status** Info
- **Source** Identity.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.
Recommendation Consider specifying like "^0.8.0".

Listing 30:

```
pragma solidity 0.8.15;
```

3.31 CVF-31

- **Severity** Minor
- **Category** Readability
- **Status** Info
- **Source** Identity.sol

Description Defining top-level types ane constants in files named after particular contracts, interfaces, or libraries makes it harder to navigate through the code.
Recommendation Consider moving the definitions into the library or into separate file named "types.sol" and "constants.sol" or something like this.

Listing 31:

```
enum CallType

uint256 constant VALUE_OFFSET = 21;
uint256 constant CALL_DATA_OFSSET = 53;
uint256 constant DELEGATECALL_DATA_OFFSET = 21;
```

3.32 CVF-32

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Identity.sol

Description This library should be defined in a file named “IdentityLib.sol”.
Client Comment Elected to retain all components in one file to promote cohestion around magic numbers (e.g. data offsets, enum CallType). Risk of drift if the Library and Contract are separated.

Listing 32:

```
library IdentityLib
```

3.33 CVF-33

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Identity.sol

Description This line does nothing.
Recommendation Consider removing it.
Client Comment Done to silence solhint, was cleaner/easier than using a disable declaration.

Listing 33:

```
return;
```

3.34 CVF-34

- **Severity** Moderate
- **Status** Info
- **Category** Unclear behavior
- **Source** Identity.sol

Description This branch is executed for any call type value other than CALL, not only for DELEGATE_CALL.
Recommendation Consider executing only for DELEGATE_CALL and reverting for other values.
Client Comment As we control the caller, we do not check the exact CallType as a gas optimization. Documentation added.

Listing 34:

```
case 0 { // false
```

3.35 CVF-35

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** Identity.sol

Description This code in unreachable.
Client Comment Updated documentation & reason.

Listing 35:

```
revert("ID: INVALID_CALL_TYPE");
```

3.36 CVF-36

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** Call.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.
Recommendation Consider specifying like "^0.8.0".

Listing 36:

```
pragma solidity 0.8.15;
```

3.37 CVF-37

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** Call.sol

Description This library should be defined in a file named "CallLib.sol".

Listing 37:

```
library CallLib
```

3.38 CVF-38

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** Call.sol

Description Expression "success == true" is equivalent to just "success".
Client Comment Elected to maintain verbose statement in this case.

Listing 38:

```
if (aSuccess == true) {
```

3.39 CVF-39

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** Call.sol

Description This code could be simplified as: if (!success) { assembly { ... } }
Client Comment Elected to maintain for two reasons:
1. Success case is more common & so we optimize for this. 2. Early return reduces amount of nesting, making code easier to read.

Listing 39:

```
if (aSuccess == true) {
    return;
}

assembly {
    // SAFETY: okay to trash memory as we are reverting
    returndatacopy(0x00, 0x00, returndatasize())

    revert(0x00, returndatasize())
}
```

3.40 CVF-40

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** BaseLogic.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.
Recommendation Consider specifying like "^0.8.0".

Listing 40:

```
pragma solidity 0.8.15;
```

3.41 CVF-41

- **Severity** Minor
- **Category** Unclear behavior
- **Status** Info
- **Source** BaseLogic.sol

Description This function should return “bytes calldata” rather than “bytes memory” for efficiency.
Client Comment Elected to return memory as assembly efficiently copies the config to memory (which is needed by abi.decode that is inevitably on the receiving side of this function).

Listing 41:

```
function _readConfig() internal pure returns (bytes memory
    ↪ rConfig)
```

3.42 CVF-42

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** BaseLogic.sol

Description This code could be written in plain Solidity like this: return msg.data [msg.data.length - abi.decode (msg.data [msg.data.length - 0x20:], (uint)) - 0x20: msg.data.length - 0x20];
Client Comment Elected to maintain solidity version to save 500+ gas / call. Identity is on the hot path for all connection interactins.

Listing 42:

```
assembly ("memory-safe") {
    // SAFETY: this block respects solidity's memory model by
       ↪ writing
    //         to free memory given by mload(0x40). it then
       ↪ moves the
    //         free memory pointer by the amount of bytes
       ↪ written

    // start writing rConfig at the next free word memory
    rConfig := mload(0x40)

    // vault stores the config_size in the last word of
       ↪ calldata
    let config_size := calldataload(sub(calldatasize(), 0x20))

    // [length, ...data]
    // the first word of a byte array is the data length
    mstore(rConfig, config_size)

    // we can now right the data, remembering that the very
       ↪ last word
    // is the config_size and so we shouldn't copy it
    calldatacopy(add(rConfig, 0x20), sub(sub(calldatasize(),
       ↪ config_size), 0x20), config_size)

    // we've only written rConfig, so we can simply increase
       ↪ free mem by
    // rConfig + 0x20 (the length word for byte[])
    mstore(0x40, add(rConfig, add(config_size, 0x20)))
}

return rConfig;
```

3.43 CVF-43

- **Severity** Moderate
- **Category** Overflow/Underflow
- **Status** Info
- **Source** BaseLogic.sol

Description Underflow is possible here.
Client Comment Given the caller is trusted (the Vault), we assume proper encoding & values. Also, using the Vault setConnection it would be prohibitively expensive to push the connection byte size to overflow point.
The Vault will never send < 0x20 bytes of calldata so underflow should be avoided.

Listing 43:

```
let config_size := calldataload(sub(calldatasize(), 0x20))

calldatacopy(add(rConfig, 0x20), sub(sub(calldatasize(), config_size), 0x20), config_size)
```

3.44 CVF-44

- **Severity** Moderate
- **Category** Overflow/Underflow
- **Status** Info
- **Source** BaseLogic.sol

Description Overflow is possible here.
Client Comment Given the caller is trusted (the Vault), we assume proper encoding & values. Also, using the Vault setConnection it would be prohibitively expensive to push the connection byte size to overflow point.
The Vault will never send < 0x20 bytes of calldata so underflow should be avoided.

Listing 44:

```
mstore(0x40, add(rConfig, add(config_size, 0x20)))
```

3.45 CVF-45

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** Create2Deployer.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions. Consider specifying like "^0.8.0".

Listing 45:

```
pragma solidity 0.8.15;
```

3.46 CVF-46

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** Create2.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions. Consider specifying like "^0.8.0".

Listing 46:

```
pragma solidity 0.8.15;
```

3.47 CVF-47

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** Create2.sol

Description This library should be defined in a file named "Create2Lib.sol".

Listing 47:

```
library Create2Lib
```

3.48 CVF-48

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** Create2.sol

Description The value "keccak256(aInitCode)" could be precomputed in most scenarios.
Recommendation Consider accepting as an argument the hash of an init code instead of the init code itself.
Client Comment Decided against it as it hurts type-safety & usability of the helper. Added natspec warning against deploying the contract.

Listing 48:

```
bytes32 lAddress = keccak256(abi.encodePacked(hex"ff",
    ↪ aDeployer, aSalt, keccak256(aInitCode)));
```

3.49 CVF-49

- **Severity** Minor
- **Category** Procedural
- **Status** Fixed
- **Source** IVault.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.
Recommendation Consider specifying like "^0.8.0".

Listing 49:

```
pragma solidity 0.8.15;
```

3.50 CVF-50

- **Severity** Minor
- **Category** Readability
- **Status** Info
- **Source** IVault.sol

Description Defining top-level types in files named after particular contracts, interfaces, or libraries makes it harder to navigate through the code.
Recommendation Consider moving the definitions into the library or into a separate file named "types.sol" or something like this.
Client Comment These types are unqiue to/owned by the Vault. Hence we feel it appropriate to leave these in IVault.sol

Listing 50:

```
struct ConnectionTuple

struct RawCall
```

3.51 CVF-51

- **Severity** Minor
- **Category** Bad datatype
- **Status** Fixed
- **Source** IVault.sol

Description The type of this field should be "Identity".

Listing 51:

```
address identity;
```

3.52 CVF-52

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** IVault.sol

Description The name is confusing, as one could think that this function sets the only keeper for the contract, while actually it sets the keeper status for a particular address.
Recommendation Consider choosing a better name.
Client Comment Not clear how to easily improve naming. This function sets the status for a single element within a set of elements. API is small so should be easy for readers to understand the full consequences.

Listing 52:

```
function setKeeper(address keeper, bool enabled) external;
```

3.53 CVF-53

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** IVault.sol

Description The name is confusing, as one could think that this function sets the only approved selector, while actually it sets the approval status for a particular selector.
Recommendation Consider choosing a better name.
Client Comment Not clear how to easily improve naming. This function sets the status for a single element within a set of elements. API is small so should be easy for readers to understand the full consequences.

Listing 53:

```
function setApprovedSelector(bytes4 selector, bool safe)
    ↪ external;
```

3.54 CVF-54

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** IVault.sol

Description The name is confusing, as one could think that this function sets the only approved logic, while actually it sets the approval status for a particular logic.
Recommendation Consider choosing a better name.
Client Comment Not clear how to easily improve naming. This function sets the status for a single element within a set of elements. API is small so should be easy for readers to understand the full consequences.

Listing 54:

```
function setApprovedLogic(address logic, bool enabled) external
    ↪ ;
```

3.55 CVF-55

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** IVault.sol

Description The name is confusing, as one could think that this function sets the only connection, while it actually adds a new connection or updates an existing connection.
Recommendation Consider choosing a better name.
Client Comment Alternative names would be too verbose, acknowledge that this function behaves different from other setMethods because it the array is append-only

Listing 55:

```
function setConnection(string calldata name, bytes calldata
    ↪ config) external returns (uint256 connectionIndex);
```

3.56 CVF-56

- **Severity** Minor
- **Category** Bad datatype
- **Status** Fixed
- **Source** IVault.sol

Description The return type should be “Identity [] memory”.

Listing 56:

```
function identities() external returns (address[] memory);
```

3.57 CVF-57

- **Severity** Minor
- **Category** Bad datatype
- **Status** Fixed
- **Source** IVault.sol

Description The return type should be “Identity”.

Listing 57:

```
function createIdentity() external returns (address identity);
```

3.58 CVF-58

- **Severity** Minor
- **Category** Bad datatype
- **Status** Fixed
- **Source** IVault.sol

Description This function should be payable.

Listing 58:

```
function rawCall(RawCall[] calldata calls) external returns (
  ↪ bytes[] memory);
```