ABDK CONSULTING

MAKER IMPROVEMENT
FINAL SMART CONTRACT
AUDIT

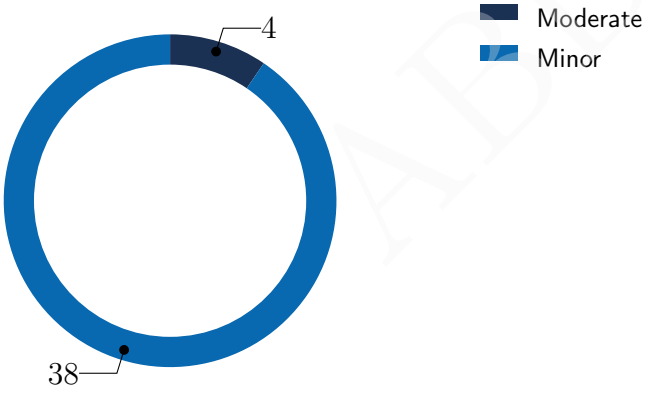
Yield Protocol

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
30th April 2021

We've been asked to review the smart contracts for the Maker Improvement Proposal in the Yield projects, given in separate files. We found several major problems, which have been addressed, and a number of issues of lesser importance.



Findings

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

ID	Severity	Category	Status
CVF-28	Minor	Documentation	Opened
CVF-29	Minor	Bad datatype	Opened
CVF-30	Minor	Bad datatype	Opened
CVF-31	Minor	Suboptimal	Opened
CVF-32	Minor	Documentation	Opened
CVF-33	Minor	Suboptimal	Opened
CVF-34	Minor	Documentation	Opened
CVF-35	Minor	Flaw	Opened
CVF-36	Minor	Suboptimal	Opened
CVF-37	Minor	Bad naming	Opened
CVF-38	Minor	Bad naming	Opened
CVF-39	Minor	Suboptimal	Opened
CVF-40	Minor	Documentation	Opened
CVF-41	Minor	Suboptimal	Opened
CVF-42	Minor	Unclear behavior	Opened

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	Mar. 23, 2021	D. Khovratovich	Initial Draft
1.0	Mar. 24, 2021	D. Khovratovich	Release
1.1	Mar. 24, 2021	D. Khovratovich	Severity and Category revision
2.0	Mar. 24, 2021	D. Khovratovich	Release
2.1	Apr. 29, 2021	D. Khovratovich	Add comment
3.0	Apr. 29, 2021	D. Khovratovich	Final 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. We have looked at certain contracts at the dss-tlm repo. We reviewed the following files

- tlm.sol;
- dss-interfaces/dd/DaiJoinAbstract.sol;
- dss-interfaces/dd/DaiAbstract.sol;
- dss-interfaces/dd/VatAbstract.sol;
- dss/lib.sol

at the commit 56054a. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations. The fixes were also reviewed.

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
- **Status** Info
- **Category** Suboptimal
- **Source** tlm.sol

Recommendation Should be "ĝ0.6.0" according to a common best practice, unless there is something special about this particular version.
Client Comment Fixed in add9bc0, we need 0.6.5 for immutable.

Listing 1:

```
1 solidity ^0.6.7;
```

3.2 CVF-2

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

Recommendation This interface should be moved to a separate file "AuthGemJoinAbstract".
Client Comment It is the MakerDAO coding style to do it like this.

Listing 2:

```
10 AuthGemJoinAbstract {
```

3.3 CVF-3

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

Description These functions are not documented which makes it hard to understand what they actually do.
Recommendation Consider adding documentation comments.
Client Comment Fixed in 8e47c0d.

Listing 3:

```
11 function ilk() external view returns (bytes32);
   function gem() external view returns (MaturingGemAbstract);
   function join(address, uint256) external;
   function exit(address, uint256) external;

19 function approve(address spender, uint256 amount) external
     ↪ returns (bool);
20 function balanceOf(address usr) external view returns (uint256);
   function maturity() external view returns (uint256);
   function transferFrom(address src, address dst, uint wad)
     ↪ external returns (bool);
   function redeem(address src, address dst, uint256 amount)
     ↪ external returns (uint256);
```

3.4 CVF-4

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

Recommendation This interface should be moved to a separate file "MaturingGemAbstract".
Client Comment It is the MakerDAO coding style to do it like this.

Listing 4:

```
18 MaturingGemAbstract {
```

3.5 CVF-5

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

Description It is a good practice to name Solidity files after the contracts, interfaces, or libraries contained in them.
Recommendation So this contract should be in a file named "DssTim.sol".
Client Comment It is the MakerDAO coding style to do it like this.

Listing 5:

```
29 DssTlm is LibNote {
```

3.6 CVF-6

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

Description The event is emitted even if wards[usr] was already 1.
Client Comment This is the expected behaviour across the MakerDAO platform.

Listing 6:

```
33 function rely(address usr) external auth { wards[usr] = 1; emit
   ↪ Rely(usr); }
```

3.7 CVF-7

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

Description The event is emitted even if wards[usr] was already 0.
Client Comment This is the expected behaviour across the MakerDAO platform.

Listing 7:

```
34 function deny(address usr) external auth { wards[usr] = 0; emit
   ↪ Deny(usr); }
```

3.8 CVF-8

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

Description The name is confusing. What does it mean?
Client Comment In MakerDAO an Ilk is a collateral type, usually indexed by an ilk bytes 32 identifier. https://docs.makerdao.com/smart-contract-modules/core-module/vat-detailed-documentation

Listing 8:

```
42 struct Ilk {
```

3.9 CVF-9

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

Recommendation The type should be "AuthGemJoinAbstract".
Client Comment It is the MakerDAO coding style to do it like this.

Listing 9:

```
43 address gemJoin;
```

3.10 CVF-10

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

Description The name is confusing, what does it mean?
Client Comment The ‘vow‘ is a component of MakerDAO that represents the balance sheet. https://docs.makerdao.com/smart-contract-modules/system-stabilizer-module/vow-detailed-documentation

Listing 10:

```
50 address immutable public vow;
```

3.11 CVF-11

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** tlm.sol

Description The meaning of the keys in this mapping is unclear.
Recommendation Consider adding a documentation comment.
Client Comment In MakerDAO an Ilk is a collateral type, usually indexed by an ilk bytes 32 identifier. https://docs.makerdao.com/smart-contract-modules/core-module/vat-detailed-documentation

Listing 11:

```
52 mapping (bytes32 => Ilk) public ilks; // Registered maturing
    ↪ gems
```

3.12 CVF-12

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** tlm.sol

Recommendation The type of the "daiJoin_" argument should be "DaiJoinAbstract".
Client Comment It is the MakerDAO coding style to do it like this.

Listing 12:

```
55 constructor(address daiJoin_, address vow_) public {
```

3.13 CVF-13

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

Description This code replicates the logic of "rely" function.
Recommendation Consider avoiding code duplication by extracting the common code into an utility function.
Client Comment It is the MakerDAO coding style to do it like this.

Listing 13:

```
56 wards[msg.sender] = 1;
   emit Rely(msg.sender);
```

3.14 CVF-14

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** tlm.sol

Recommendation "1e18" would be shorter and more readable.
Client Comment It is the MakerDAO coding style to do it like this.

Listing 14:

```
68 uint256 constant WAD = 10 ** 18;
```

3.15 CVF-15

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** tlm.sol

Recommendation "1e27" would be shorter and more readable.
Client Comment It is the MakerDAO coding style to do it like this.

Listing 15:

```
69 uint256 constant RAY = 10 ** 27;
```

3.16 CVF-16

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

Description No access levels are specified for these constants so internal access will be used by default.
Recommendation Consider specifying access levels explicitly.
Client Comment It is the MakerDAO coding style to do it like this. Internal is fine.

Listing 16:

```
68 uint256 constant WAD = 10 ** 18;
   uint256 constant RAY = 10 ** 27;
70 uint256 constant MAXINT256 =
     ↪ 57896044618658097711785492504343953926634992332820282019
   728792003956564819967;
```

3.17 CVF-17

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

Description Hexadecimal literal would be more appropriate here. Also, starting from Solidity 0.6.8, the following syntax is supported: "type(int256).max".
Client Comment Removed in c371691

Listing 17:

```
70 uint256 constant MAXINT256 =
   ↪ 57896044618658097711785492504343953926634992332820282019728
   ↪
792003956564819967;
```

3.18 CVF-18

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

Recommendation This could be rewritten as: require ((z = int256 (x)) >= 0);
Client Comment Fixed in c371691

Listing 18:

```
99  require(x <= MAXINT256, "DssTlm/int256-overflow");
100 return(int256(x));
```

3.19 CVF-19

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

Description In fact the answer is 0 if x*RAY <y/2 , which is not the same.
Client Comment Fixed in 509ba41

Listing 19:

```
114 /// @dev x / y, where x is a decimal of base RAY. Rounds to zero
    ↪  if x*y < RAY / 2
```

3.20 CVF-20

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

Description Even though a zero address won't respond to the 'approve' below, it might still make sense to add a check that the 'gemJoin' is not 0, for readability.
Client Comment Fixed in efe26a8

Listing 20:

```
123 require(ilks[ilk].gemJoin == address(0), "DssTlm/ilk-already-
    ↪ init");
```

3.21 CVF-21

- **Severity** Moderate
- **Status** Fixed
- **Category** Documentation
- **Source** tlm.sol

Description The function actually allows setting target yield only, there is no logic to set the ceiling debt.
Client Comment Fixed in 30183b7

Listing 21:

```
130 /// @dev Set up the ceiling debt or target yield for a maturing
    ↪ gem.
```

3.22 CVF-22

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** tlm.sol

Description It is unclear what this function does.
Recommendation Consider adding a documentation comment.
Client Comment Fixed in a19b5fd

Listing 22:

```
143 function nope(address usr) external note auth {
```

3.23 CVF-23

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** tlm.sol

Description The expression "ilks[ilk]" is calculated several times.
Recommendation Consider calculating once and caching in a local variable.
Client Comment The expression is calculated several times, but the struct fields are different, so we won't save any SLOAD. Is this really worth it?
Further recommendation Calculating it once would not save SLOAD, but would save calculating the storage address of the mapping element, i.e. would save a keccak256 calculation.

Listing 23:

```
150 AuthGemJoinAbstract gemJoin = AuthGemJoinAbstract(ilks[ilk].
    ↪ gemJoin);

153 uint256 price = rpow(ilks[ilk].yield, time, RAY);
```

3.24 CVF-24

- **Severity** Moderate
- **Status** Opened
- **Category** Flaw
- **Source** tlm.sol

Recommendation This value should be rounded down, i.e. towards the benefit of the system.
Client Comment Does this justify using a different ‘rpow‘ formula?
Further recommendation The rest of the code tries to round properly, and certain math functions exist in two version for this reason. So having two version of ‘rpow‘ or just a single version but with an additional parameter seems logical.

Listing 24:

```
153 uint256 price = rpow(ilks[ilk].yield, time, RAY);
```

3.25 CVF-25

- **Severity** Moderate
- **Status** Fixed
- **Category** Flaw
- **Source** tlm.sol

Recommendation This value should be rounded down, i.e. towards the benefit of the system.
Client Comment Fixed in 509ba41

Listing 25:

```
154 uint256 daiAmt = rdiv(gemAmt, price);
```

3.26 CVF-26

- **Severity** Moderate
- **Status** Fixed
- **Category** Unclear behavior
- **Source** tlm.sol

Description The returned value is ignored.
Client Comment Fixed in 2506563

Listing 26:

```
156 gem.transferFrom(msg.sender, address(this), gemAmt);

168 dai.transferFrom(msg.sender, address(this), amt);
```

3.27 CVF-27

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DaiJoinAbstract.sol

Recommendation Should be "ô.5.0" according to a common best practice, unless there is something special about this particular version.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 27:

```
2 solidity >=0.5.12;
```

3.28 CVF-28

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** DaiJoinAbstract.sol

Description The functions are not documented which makes it hard to understand what they actually do.
Recommendation Consider adding documentation comments.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 28:

```
6  function wards(address) external view returns (uint256);
   function rely(address usr) external;
   function deny(address usr) external;
   function vat() external view returns (address);
10 function dai() external view returns (address);
   function live() external view returns (uint256);
   function cage() external;
   function join(address, uint256) external;
   function exit(address, uint256) external;
```

3.29 CVF-29

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** DaiJoinAbstract.sol

Recommendation The return type of this function should be "VatAbstract".
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 29:

```
9  function vat() external view returns (address);
```

3.30 CVF-30

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** DaiJoinAbstract.sol

Recommendation The return type of this function should be "DaiAbstract".
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 30:

```
10 function dai() external view returns (address);
```

3.31 CVF-31

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** VatAbstract.sol

Recommendation Should be "ô.5.0" according to a common best practice, unless there is something special about this particular version.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 31:

```
2 solidity >=0.5.12;
```

3.32 CVF-32

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** VatAbstract.sol

Description The functions are not documented which makes it hard to understand what they actually do.
Recommendation Consider adding documentation comments.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 32:

```
6  function wards(address) external view returns (uint256);
   function rely(address) external;
   function deny(address) external;
   function can(address, address) external view returns (uint256);
10 function hope(address) external;
   function nope(address) external;
   function ilks(bytes32) external view returns (uint256, uint256,
      ↪ uint256, uint256, uint256);
   function urns(bytes32, address) external view returns (uint256,
      ↪ uint256);
   function gem(bytes32, address) external view returns (uint256);
   function dai(address) external view returns (uint256);
   function sin(address) external view returns (uint256);
   function debt() external view returns (uint256);
   function vice() external view returns (uint256);
   function Line() external view returns (uint256);
20 function live() external view returns (uint256);
   function init(bytes32) external;
   function file(bytes32, uint256) external;
   function file(bytes32, bytes32, uint256) external;
   function cage() external;
   function slip(bytes32, address, int256) external;
   function flux(bytes32, address, address, uint256) external;
   function move(address, address, uint256) external;
   function frob(bytes32, address, address, address, int256, int256
      ↪ ) external;
   function fork(bytes32, address, address, int256, int256)
      ↪ external;
30 function grab(bytes32, address, address, address, int256, int256
      ↪ ) external;
   function heal(uint256) external;
   function suck(address, address, uint256) external;
   function fold(bytes32, address, int256) external;
```

3.33 CVF-33

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DaiAbstract.sol

Recommendation Should be "ô.5.0" according to a common best practice, unless there is something special about this particular version.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 33:

```
2 solidity >=0.5.12;
```

3.34 CVF-34

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** DaiAbstract.sol

Description The functions are not documented which makes it hard to understand what they actually do.
Recommendation Consider adding documentation comments.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 34:

```
6  function wards(address) external view returns (uint256);
   function rely(address) external;
   function deny(address) external;
   function name() external view returns (string memory);
10 function symbol() external view returns (string memory);
   function version() external view returns (string memory);
   function decimals() external view returns (uint8);
   function totalSupply() external view returns (uint256);
   function balanceOf(address) external view returns (uint256);
   function allowance(address, address) external view returns (
      ↪ uint256);
   function nonces(address) external view returns (uint256);
   function DOMAIN_SEPARATOR() external view returns (bytes32);
   function PERMIT_TYPEHASH() external view returns (bytes32);
   function transfer(address, uint256) external;
20 function transferFrom(address, address, uint256) external
      ↪ returns (bool);
   function mint(address, uint256) external;
   function burn(address, uint256) external;
   function approve(address, uint256) external returns (bool);
   function push(address, uint256) external;
   function pull(address, uint256) external;
   function move(address, address, uint256) external;
   function permit(address, address, uint256, uint256, bool, uint8,
      ↪  bytes32, bytes32) external;
```

3.35 CVF-35

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** DaiAbstract.sol

Description The return type is missing, which violates ERC20 standard.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 35:

```
19 function transfer(address, uint256) external;
```

3.36 CVF-36

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** lib.sol

Recommendation Should be "0̂.5.0" according to a common best practice, unless there is something special about this particular version.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 36:

```
14 solidity >=0.5.12;
```

3.37 CVF-37

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** lib.sol

Recommendation It is good practice to name Solidity files after the contracts, interfaces, or libraries contained in them, so this file should be named "LibNote.sol" to simplify code navigation.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 37:

```
16 LibNote {
```

3.38 CVF-38

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** lib.sol

Recommendation The prefix "Log" is redundant in even name as all events are logged.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 38:

```
17 event LogNote(
```

3.39 CVF-39

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** lib.sol

Description Logging an event after executing the function body means, that in case of several nested calls, the order or event will be the reversal of the order of function calls, which would be confusing.
Recommendation Consider logging the event before the function body is executed.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 39:

```
26 _;
assembly {
```

3.40 CVF-40

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** lib.sol

Description It actually contains 7 words (224 bytes) of calldata.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 40:

```
28 // log an 'anonymous' event with a constant 6 words of calldata
```

3.41 CVF-41

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** lib.sol

Recommendation This is redundant, as the allocated memory is not used after the modifier.
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 41:

```
31 mstore(0x40, add(mark, 288))                  // update free memory
   ↪ pointer
```

3.42 CVF-42

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** lib.sol

Description The first 68 bytes of calldata are already encoded as indexed parameters. Is it necessary to encode them again in data field?
Client Comment This is MakerDAO code out of my control, I'll pass the issue along.

Listing 42:

```
34 calldatacopy(add(mark, 0x40), 0, 224)         // bytes payload
```