

# Secure Code Review of wormhole & pricecaster-v2

Findings and Recommendations Report Presented to:

**Rand Labs**

July 22, 2022  
Version: 1.1

Presented by:

Kudelski Security, Inc.  
5090 North 40th Street, Suite 450  
Phoenix, Arizona 85018

STRICTLY CONFIDENTIAL

## TABLE OF CONTENTS

TABLE OF CONTENTS .....	2
LIST OF FIGURES.....	3
LIST OF TABLES .....	3
EXECUTIVE SUMMARY .....	4
Overview .....	4
Key Findings .....	4
Scope and Rules of Engagement .....	5
TECHNICAL ANALYSIS & FINDINGS .....	6
Observations .....	7
Findings.....	10
KS-RL-01 – Wrong Parameter Check → Send an Invalid Transaction .....	11
KS-RL-02 – Wrong Targeted Chain Check .....	13
KS-RL-03 – Algorand smart contract not aligned with Ethereum smart contract .....	14
KS-RL-04 – Invalid Asset Check → Transaction Falsely Rejected .....	16
KS-RL-05 – Undocumented limitation of the UnitName of asset.....	17
KS-RL-06 – No Boundaries Check .....	18
KS-RL-07 – Unused application_args .....	19
KS-RL-08 – Too Many Hardcoded Values .....	20
KS-RL-09 – Code Duplication.....	22
KS-RL-10 – Algorand guidelines not followed .....	24
KS-RL-11 – Python Code Quality .....	25
KS-RL-12 – Inappropriate Content .....	27
METHODOLOGY .....	28
Tools .....	29
Vulnerability Scoring Systems .....	30
KUDELSKI SECURITY CONTACTS .....	<b>Error! Bookmark not defined.</b>

**LIST OF FIGURES**

Figure 1: Findings by Severity..... 6

**LIST OF TABLES**

Table 1: Scope ..... 5  
Table 2: Findings Overview..... 10

## EXECUTIVE SUMMARY

### Overview

Rand Labs engaged Kudelski Security to perform a secure code assessment of both wormhole and pricecaster-v2 Algorand smart contracts.

The assessment was conducted remotely by the Kudelski Security Team. Testing took place on June 20, 2022 - July 20, 2022, and focused on the following objectives:

- Provide the customer with an assessment of the security of recently added functionality and any risks that were discovered during the engagement.
- To provide a professional opinion on the maturity, efficiency, and coding practices
- To identify potential issues and include improvement recommendations based on the result of our code review

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the Kudelski Security Teams took to identify and validate each issue, as well as any applicable recommendations for remediation.

### Key Findings

The following are the major themes and issues identified during the testing period. These, along with other items, within the findings section, should be prioritized for remediation to reduce to the risk they pose.

- A theme across the codebase was functional bugs. A prerequisite for secure code is functional correctness – working code ‘as expected’ and this is an area that would benefit from attention.
- In several areas there was little or no documentation of expected behavior beyond a higher level whitepaper. As such, assessing the correctness of the logic proved challenging and could leave the code exposed to vulnerabilities where correct behavior could not be confirmed.
- Overall, the code reflected a lower quality than we would expect for the level of risk held by the bridge. Symptoms include lower level of commenting, and coding conventions like hardcoded indices and magic values (eg. `Is 18446744073709551614` a magic value?). This made it difficult to review. During our interviews we learned that some of the Algorand code was written from reverse engineering (e.g., Ethereum code), without clear explanation of the rationale or reliability of original code.

With this being said it is worth noting that some of the documentation (`MEMORY.md`, describing `tmplSig.py` for instance) was of a high quality and this should serve as a strong foundation for improving the level of documentation.

## Scope and Rules of Engagement

Kudelski performed a Secure Code Review of wormhole & pricecaster-v2 for Rand Labs. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied with the commit hashes at:

- <https://github.com/certusone/wormhole/commit/3fcb35132ceadba82c283d7e89a7174fd0317634>
  - Subfolder `algorand`
- <https://github.com/randlabs/pricecaster-v2/commit/fcc2b411de7e5199cae421c58fd59d955f059371>
  - Subfolder `teal/pyteal` (legacy excluded)

In-Scope Contracts	
Wormhole	Pricecaster-v2
algorand/ ├── sandbox-algorand ├── teal ├── test ├── admin.py ├── deploy.sh ├── Dockerfile ├── gentest.py ├── <b>globals.py</b> ├── <b>inlineasm.py</b> ├── <b>local_blob.py</b> ├── Makefile ├── MEMORY.md ├── NOTES.md ├── package.json ├── package-lock.json ├── Pipfile ├── Pipfile.lock ├── README.md ├── requirements.txt ├── runPythonUnitTests.sh ├── sandbox ├── test_contract.py ├── testnet-update ├── <b>TmplSig.py</b> ├── <b>token_bridge.py</b> ├── <b>vaa_verify.py</b> ├── <b>wormhole_core.py</b>	pyteal/ ├── legacy ├── <b>globals.py</b> ├── <b>inlineasm.py</b> ├── <b>mapper.py</b> ├── <b>pricecaster-v2.py</b>

Table 1: Scope

## TECHNICAL ANALYSIS & FINDINGS

During the Secure Code Review of wormhole & pricecaster-v2, we discovered 4 findings that had a high severity rating, as well as 2 of low severity.

The following chart displays the findings by severity.

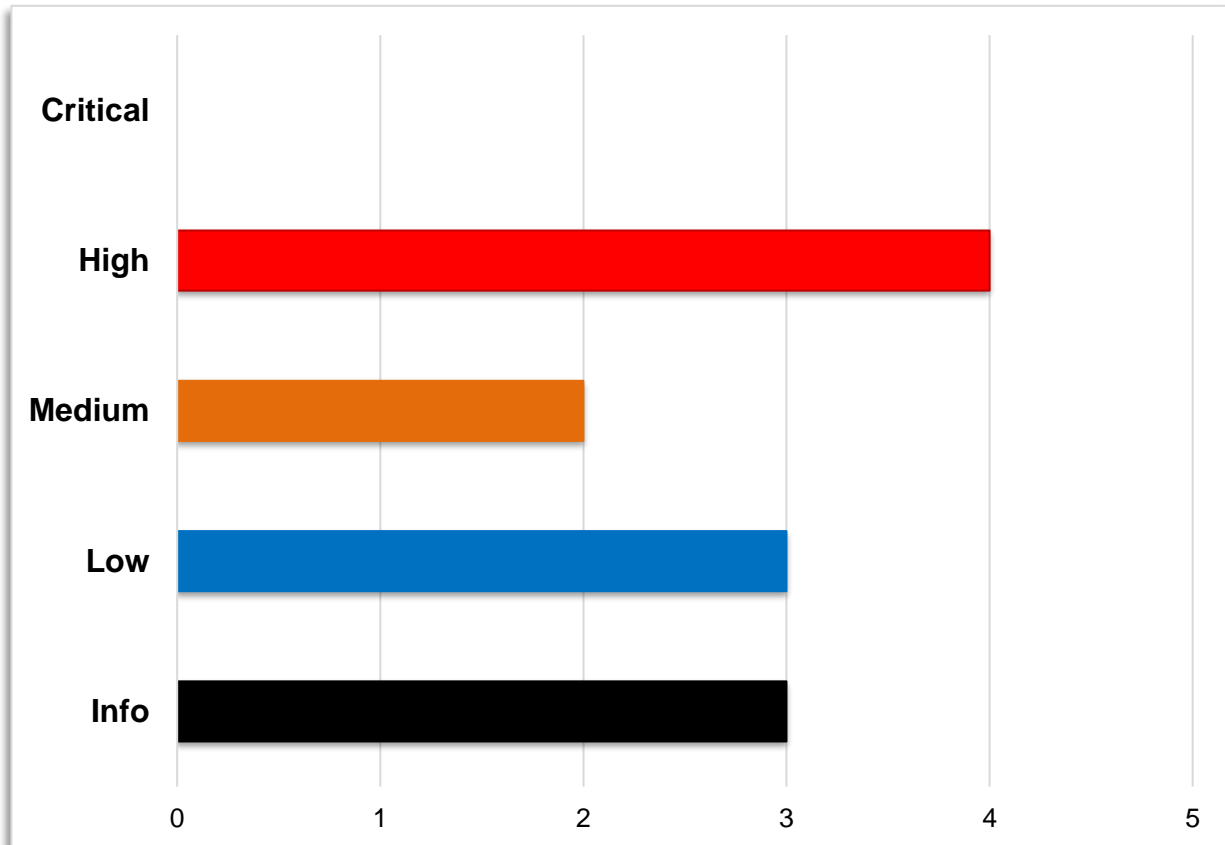


Figure 1: Findings by Severity

## Observations

As introduced previously, we observed that part of the code reflected a lack of quality, and this on different aspects of the project, with a direct impact on code auditing. Let us detail some important steps of this process.

## File tree

The file tree - discovered on the first connection to the repo - is the first source of information as to the content to be audited. The questions asked during this initial phase are:

- How is the project structured?
- What are the assets (smart contracts) to be audited? Where are they located?

Looking at [Table 1 : Scope](#) give us first insights:

- Wormhole ↔ not structured
  - All files are at the same level
    - \*.json, \*.lock, \*.md, \*.py, \*.sh, \*.txt, and \*.py files
    - \*.py files: how many smart contracts are there?
      - Some of them seem to be for testing...
  - First difficulty: identify assets
- Pricecaster-v2 ↔ structured
  - Smart contracts are directly identified in a dedicated “pyteal” folder
    - “pyteal” programming language
    - 4 files

## Documentation

In code audit, one of the main tasks is to check compliance with the code with the provided technical documentation:

- Any document format is welcome (presentation, specification, diagrams, ...)
- If they allow to understand – in details – your business flow

The `algorand/MEMORY.md` file is a very good example of such a documentation which includes:

- Background
- The “allocator” program: a. Instantiation, off-chain b. Instantiation, on-chain
- Many links to source code: `algorand/TmplSig.py`, `algorand/wormhole_core.py`, `algorand/admin.py`
- The `algorand/TmplSig.py`, 144 lines, is detailed almost line by line

Unfortunately, we could not find anything like that for all the other files, especially for the `algorand/token_bridge.py` file and it is 1021 lines

## Source Code

The general observation we made about the source code is that there are no rules, whether for the container or for the content.

About filenames, many conventions are used:

- PascalCase: `algorand/TmplSig.py`
  - kebab-case: `pyteal/pricecaster-v2.py`
  - snake\_case: `algorand/token_bridge.py`
  - all-lowercase: `algorand/inlineasm.py` & `pyteal/inlineasm.py`
- As reference see <https://peps.python.org/pep-0008/#package-and-module-names>

While the lack of a file naming convention is not really a big deal, it becomes much more important when it comes to their content:

- With or without shebang?
- With or without header?
  - With or without copyright notice?
  - With or without license notice?
  - With or without description?

`vaa_verify.py` and `pricecaster-v2.py` are the only ones documented here
- With or without comments?
- With or without docstrings?
  - As they are used by modern editor/IDE to show details when the cursor hovers some code, it is a very valuable tool for all readers (developer, reviewer, ...)
- Which naming convention for constants, classes, functions, variables?

- One example among many

```

35 max_keys = 15
36 max_bytes_per_key = 127
37 bits_per_byte = 8
38
39 bits_per_key = max_bytes_per_key * bits_per_byte
40 max_bytes = max_bytes_per_key * max_keys
41 max_bits = bits_per_byte * max_bytes
42
43 portal_transfer_selector = MethodSignature("portal_transfer(byte[])byte[]")
44
45 > def fullyCompileContract(genTeal, client: AlgodClient, contract: Expr, name, devmode) -> bytes: ...
62
63 > def clear_token_bridge():--
65
66 def approve_token_bridge(seed_amt: int, tmpl_sig: TmplSig, devMode: bool):
67     blob = LocalBlob()
68     tid_x = ScratchVar()
69     mfee = ScratchVar()
70
71 > def MagicAssert(a) -> Expr: --
77
78 @Subroutine(TealType.uint64)
79 > def governanceSet() -> Expr: --
82

```

- First time we observed that the very widely used rule of SCREAMING\_CASE for constants is not applied, convention used later in the same file
- 5 functions ↔ 3 conventions
- Not a single comment, nor docstring



- Hard-coded values vs constants?
  - One example among many
 

```
off.store(off.load()+Int(43)),

MagicAssert(Int(2) ==      Btoi(Extract(Txn.application_args[1], off.load(),      Int(1))),
Address.store(            Extract(Txn.application_args[1], off.load() + Int(1),  Int(32))),

FromChain.store(          Btoi(Extract(Txn.application_args[1], off.load() + Int(33), Int(2))),
Decimals.store(           Btoi(Extract(Txn.application_args[1], off.load() + Int(35), Int(1))),
Symbol.store(             Extract(Txn.application_args[1], off.load() + Int(36), Int(32))),
Name.store(               Extract(Txn.application_args[1], off.load() + Int(68), Int(32))),
```

    - Do you think it is easy to check that correct offsets and lengths are used?
    - If the structure evolves, it will be necessary to update the whole code, line by line  
→ very error prone
    - Whereas with constants like ITEM OFFSET and ITEM SIZE  
→ it is immediately readable and understandable  
→ a single update and the whole code remains correct
- Common vs. case-by-case processing?
  - Example of function parameters:
    - Number of parameters is sometimes checked, but often is not checked
    - Sometimes at the beginning, sometimes later in the processing
    - Parameter sometimes used without any check

Therefore, since all files are different from each other, from naming to processing, the main difficulty is the readability and comprehensibility of the code.

## Recommendations

Adopting best practices from the software engineering industry would help to code the right way:

- Define coding conventions
  - Standardized conventions help keep the code relevant and useful for clients, future developers, and the coders themselves
- Enforce coding conventions
  - This task can be automated with a static analyzer (e.g., [PyLint](#)) to:
    - Provide consistency in engineering (and audit) teams
    - Provide insight into code without executing it
    - Search for bugs at the early stages (although not all)
    - Find of security problems at an early stage
- Enhance code readability
  - Try to always write code that can be easily understood by others
    - Allocate appropriate names to all classes, functions, variables
    - Avoid abbreviations as much as possible to reduce ambiguity
- Define a standard/systematic control flow
  - First, check all parameters
    - Number of, value/range, consistency
  - According to the specification, process step by step
    - Abort/reject transaction at the slightest deviation

## Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

#	Severity	Description
KS-RL-01	High	Wrong Parameter Check → Send an Invalid Transaction
KS-RL-02	High	Wrong Targeted Chain Check
KS-RL-03	High	Algorand smart contract not aligned with Ethereum smart contract
KS-RL-04	High	Invalid Asset Check → Transaction Falsely Rejected
KS-RL-05	Medium	Undocumented limitation of the UnitName of asset
KS-RL-06	Medium	No Boundaries Check
KS-RL-07	Low	Unused application_args
KS-RL-08	Low	Too Many Hardcoded Values
KS-RL-09	Low	Code Duplication
KS-RL-10	Info	Algorand guidelines not followed
KS-RL-11	Info	Python Code Quality
KS-RL-12	Info	Inappropriate Content

Table 2: Findings Overview

## KS-RL-01 – Wrong Parameter Check → Send an Invalid Transaction

Severity

HIGH

Impact	Likelihood	Difficulty
High	High	Low

### Description

In `token_bridge.py`, in `sendTransfer()` method, the expected number of arguments is 6 (if no payload) or 7 (if payload). But the `sendTransfer()` never checks exactly this condition.

It checks (line 766) if the `Txn.application_args.length()` is equal to 6, and it checks (line 777 and 781) if the `Txn.application_args.length()` is equal to 7.

### Impact

If an attacker sends a “`sendTransfer`” transaction with 8 parameters, the smart contract will accept the transaction and the behavior of the smart contract will be impacted.

The smart contract will publish a message of type “`TransferWithPayload`” (line 768), with the fee value in the “`FromAddress`” field of the message (line 777) and with no payload.

Then the “`magic line`” (as it is mentioned in the code) will not be executed (line 781).

As a result, the transaction will be accepted by the smart contract and an invalid message “`TransferWithPayload`” will be sent.

It is difficult to estimate the impact as we do not know how the target chain will behave, but there is a risk of token lock in the Algorand smart contract.

### Evidence

```

765         p.store(Concat(
766             If(Txn.application_args.length() == Int(6),
767                 Bytes("base16", "01"),
768                 Bytes("base16", "03")),
769             Extract(zb.load(), Int(0), Int(24)),
770             Itob(amount.load(), # 8 bytes
771             Extract(zb.load(), Int(0), Int(32) - Len(Address.load())),
772             Address.load(),
773             FromChain.load(),
774             Extract(zb.load(), Int(0), Int(32) - Len(Txn.application_args[3])),
775             Txn.application_args[3],
776             Extract(Txn.application_args[4], Int(6), Int(2)),
777             If(Txn.application_args.length() == Int(7), Concat(Txn.sender(), Txn.application_args[6]), Concat(Extract(zb.load(), Int(0), Int(24)), Itob(fee.load())))),
778         )),
779
780         # This one magic line should protect us from overruns/underruns and trickery..
781         If(Txn.application_args.length() == Int(7),
782             MagicAssert(Len(p.load()) == Int(133) + Len(Txn.application_args[6])),
783             MagicAssert(Len(p.load()) == Int(133))),

```

### Affected Resource

- `algorand/token_bridge.py` (lines 766-783)

**Recommendation**

In `sendTransfer()`, check that `application_args` length is equal to 6 or 7.  
Do not accept any other value.

**Reference**

N/A

## KS-RL-02 – Wrong Targeted Chain Check

Severity	HIGH		
Impact	Likelihood	Difficulty	
High	High	Low	

### Description

In `wormhole_core.py`, in `hdlGovernance()` method, the smart contract checks the target of the governance message. It only accepts the target value 8 (Algorand) or 0 (all chains).

Then it checks if the governance message is a contract upgrade, an update of the guardian set, a message fee, or a Payment.

If the governance is a contract upgrade, the smart contract will accept an upgrade even if the target value is 0.

### Impact

The smart contract could accept a contract upgrade sends to all chains.

Upgrading a contract is always a sensitive part in a smart contract. It should be protected against unexpected upgrade. The impact could be high depending on the content of the new contract.

### Evidence

```

213         # What is the target of this governance message?
214         tchain.store(Extract(Txn.application_args[1], off.load() + Int(1), Int(2))),
215         # Needs to point at us or to all chains
216         MagicAssert(Or(tchain.load() == Bytes("base16", "0008"), tchain.load() == Bytes("base16", "0000"))),
217

```

### Affected Resource

- `algorand/wormhole_core.py` (line 216, 220/228)

### Recommendation

The smart contract should only accept contract upgrade with Algorand target value (8).

### Reference

N/A

## KS-RL-03 – Algorand smart contract not aligned with Ethereum smart contract

Severity	HIGH		
Impact	Likelihood	Difficulty	
High	High	Medium	

### Description

A wormhole core contract is implemented on each supported chain.

Concerning the governance messages handled by the `wormhole_core.py`, there are at least two differences with what is done on Ethereum smart contract:

1. When receiving a governance message with a Guardian Set update action, the Ethereum smart contract checks if the index is incrementing by 1.  
This is not done in the `wormhole_core.py` for Algorand.
2. When receiving a governance message with a Transfer Fee, the Ethereum smart contract accepts its own chainId and chainId 0.  
The `wormhole_core.py` only accepts its own chainId.

### Impact

Its difficult to estimate the impact as we do not know which smart contract is correctly implemented (no documentation that explains the implementation or the design choices).

### Evidence

Ethereum smart contract

<https://github.com/certusone/wormhole/blob/dev.v2/ethereum/contracts/Governance.sol>

Ethereum

```

96         // Verify that the index is incrementing via a predictable +1 pattern
97         require(upgrade.newGuardianSetIndex == getCurrentGuardianSetIndex() + 1, "index must increase in steps of 1");
98     
```

Algorand

```

242         # Make sure it is different and we can only walk forward
243         If(isBoot == Int(0), Seq(
244             MagicAssert(Txn.accounts[3] != Txn.accounts[2]),
245             MagicAssert(idx.load() > (set.load()))
246         )),
    
```

## Ethereum

```

115     function submitTransferFees(bytes memory _vm) public {
116         Structs.VM memory vm = parseVM(_vm);
117
118         // Verify the VAA is valid before processing it
119         (bool isValid, string memory reason) = verifyGovernanceVM(vm);
120         require(isValid, reason);
121
122         // Obtains the transfer from the VAA payload
123         GovernanceStructs.TransferFees memory transfer = parseTransferFees(vm.payload);
124
125         // Verify the VAA is for this module
126         require(transfer.module == module, "invalid Module");
127
128         // Verify the VAA is for this chain
129         require(transfer.chain == chainId() || transfer.chain == 0, "invalid Chain");
130     }

```

## Algorand

```

275         [a.load() == Int(4), Seq([
276             off.store(off.load() + Int(1)),
277             MagicAssert(tchain.load() == Bytes("base16", "0008")),
278             off.store(off.load() + Int(26)),

```

## Affected Resource

- algorand/wormhole\_core.py (245 and 277)

## Recommendation

Add comments in the wormhole\_core.py to explain the implementation choice and/or do the modification.

## Reference

N/A

## KS-RL-04 – Invalid Asset Check → Transaction Falsely Rejected

Severity	HIGH		
Impact	Likelihood	Difficulty	
High	High	Low	

### Description

The number of decimals is limited to 16, while an ASA can be identified with up to 19.

### Impact

Transaction with an ASA id greater than 9999999999999999 (0x2386F26FC0FFFF) is rejected.

### Evidence

```

173     @Subroutine(TealType.uint64)
174     def getFactor(dec: Expr):
175         return Cond(
176             [dec == Int(9), Int(10)],
177             [dec == Int(10), Int(100)],
178             [dec == Int(11), Int(1000)],
179             [dec == Int(12), Int(10000)],
180             [dec == Int(13), Int(100000)],
181             [dec == Int(14), Int(1000000)],
182             [dec == Int(15), Int(10000000)],
183             [dec == Int(16), Int(100000000)],
184             [dec > Int(16), Seq(Reject(), Int(1))],
185             [dec < Int(9), Int(1)]
186         )

```

### Affected Resource

algorand/token\_bridge.py (line 184)

### Recommendation

1. Document the function, especially constraints, limitations
2. Follow a logical process:
  - a. Lower limit `[dec < Int(9), Int(1)]`
  - b. In between limits `[dec == Int(9), Int(10)]`  
`[dec == Int(...), Int(...)]`
  - c. Upper limit `[dec > Int(19), Seq(Reject(), Int(1))]`
3. Maybe refactor the code with an optimized function to reduce its size.

### Reference

<https://developer.algorand.org/docs/get-details/transactions/transactions/#asset-parameters>



## KS-RL-05 – Undocumented limitation of the UnitName of asset

Severity	MEDIUM	
Impact	Likelihood	Difficulty
High	Medium	Medium

### Description

In `token_bridge.py`, in `receiveAttest()` method, the `UnitName` is limited to 7, while the max size of a unit name of an asset is 8 bytes on Algorand.

### Impact

If the `UnitName` of an asset is 8 bytes length, the smart contract will remove the last byte, and creates an asset with a modified name.

### Evidence

```

380         # Lets trim this... seems these are limited to 7 characters
381         Symbol.store(trim_bytes(Symbol.load())),
382         If (Len(Symbol.load()) > Int(7), Symbol.store(Extract(Symbol.load(), Int(0), Int(7)))),
383         Name.store(Concat(trim_bytes(Name.load()), Bytes(" (Wormhole)")),

```

### Affected Resource

`algorand/token_bridge.py` (line 382)

### Recommendation

Explain why it is limited to 7 or change this limitation to 8 for Algorand.

### Reference

<https://developer.algorand.org/docs/get-details/transactions/transactions/#asset-configuration-transaction>

## KS-RL-06 – No Boundaries Check

Severity	MEDIUM
----------	--------

Impact	Likelihood	Difficulty
Medium	Medium	Low

### Description

There is no check on the index for the blob read and write operations.

### Impact

Possible overflow.

### Evidence

```

36
37 def _key_and_offset(idx: Int) -> Tuple[Int, Int]:
38     return idx / page_size, idx % page_size
39

```

### Affected Resource

algorand/local\_blob.py

### Recommendation

Limit the index to the maximum available size.

### Reference

N/A

## KS-RL-07 – Unused application\_args

Severity	LOW	
Impact	Likelihood	Difficulty
Medium	Low	Low

### Description

In `token_bridge.py`, the `sendTransfer()` method accepts transaction with 6, 7 (or more) arguments. The `Txn.application_args[2]` is never used.

### Impact

The impact needs to be evaluated by the developer. As there is no information on the implementation choice, we do not know if this parameter is unneeded or if it has been forgotten in the `sendTransfer` implementation.

### Evidence

Check the code from line 661 to 679.  
The `Txn.application_args[2]` is never used.

### Affected Resource

`algorand/token_bridge.py` (lines 661-679)

### Recommendation

Use it in the implementation, or remove this argument if not needed.

### Reference

N/A

## KS-RL-08 – Too Many Hardcoded Values

Severity	LOW
----------	-----

Impact	Likelihood	Difficulty
Medium	Low	Low

### Description

Much of the code consists of reading or writing data from larger structures; arrays, byte arrays (VAA / blob). To do this, the basic operation is to read/write with 2 parameters: offset and length. We observed that in most cases these 2 parameters were hard-coded values.

### Impact

The very first impact is that such a code is unreadable/incomprehensible. Looking at the few samples below:

- Who can know what the developer intended?
  - What parameter is it?
- You constantly need to consult the description of the structure in use

Second impact is for the development team:

- Same remarks as above for a new developer
  - At the slightest change in the structure, you have to review all the code to apply the changes in the right places
- Very error prone, especially if the code is not fully covered by automated tests

### Evidence

```

918         return Seq(
919             # VM only is version 1
920             MagicAssert(Btoi(Extract(Txn.application_args[1], Int(0), Int(1))) == Int(1)),
921
922             off.store(Btoi(Extract(Txn.application_args[1], Int(5), Int(1))) * Int(66) + Int(14)),
923
924             # emitter is chain/contract-address
925             emitter.store(Extract(Txn.application_args[1], off.load(), Int(34))),
926             sequence.store(Btoi(Extract(Txn.application_args[1], off.load() + Int(34), Int(8)))),
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214

```

```
166         packed_price_data.store(Concat(  
167             Extract(attestation_data.load(), Int(64), Int(41)),  
168             Extract(attestation_data.load(), Int(109), Int(8)),  
169             Extract(attestation_data.load(), Int(133), Int(8)),  
170         )),
```

**Affected Resource**

algorand/token\_brigde.py  
algorand/vaa\_verify.py  
algorand/wormhole\_core.py  
pyteal/pricecaster-v2.py

**Recommendation**

1. Think twice before hardcoding a value
2. Define constants with appropriate names and constants reflecting the structure:  
ITEM\_OFFSET = X  
ITEM\_LENGTH = Y  
OTHER\_OFFSET = ITEM\_OFFSET + ITEM\_LENGTH  
OTHER\_LENGTH = Z

**Reference**

N/A

## KS-RL-09 – Code Duplication

Severity	LOW
----------	-----

Impact	Likelihood	Difficulty
Medium	Low	Low

### Description

Duplication of code in different files.

### Impact

Duplicated code is used as a code maturity metric in the industry to point out how maintainable the code base is. It is also a possible entry point for new bugs as code duplication leads to mistakes when updating/rewriting the codebase.

### Evidence

- Some constants

```

27  _max_keys = 15
28  _page_size = 128 - 1 # need 1 byte for key
29  _max_bytes = _max_keys * _page_size
30  _max_bits = _max_bytes * 8
31
32  max_keys = Int(_max_keys)
33  page_size = Int(_page_size)
34  max_bytes = Int(_max_bytes)

```

```

35  max_keys = 15
36  max_bytes_per_key = 127
37  bits_per_byte = 8
38
39  bits_per_key = max_bytes_per_key * bits_per_byte
40  max_bytes = max_bytes_per_key * max_keys
41  max_bits = bits_per_byte * max_bytes

```

- Some methods
  - get\_sig\_address
    - calling the method encode\_uvarint also duplicated
  - encode\_uvarint
  - checkForDuplicate
    - 30 duplicated lines
    - Excepted 1 extra line in wormhole\_core.py

```

131 @Subroutine(TealType.bytes)
132 def encode_uvarint(val: Expr, b: Expr):
133     buff = ScratchVar()
134     return Seq(
135         buff.store(b),
136         Concat(
137             buff.load(),
138             If(
139                 val >= Int(128),
140                 encode_uvarint(
141                     val >> Int(7),
142                     Extract(Itob((val & Int(255)) | Int(128)), Int(7), Int(1)),
143                 ),
144                 Extract(Itob(val & Int(255)), Int(7), Int(1)),
145             ),
146         ),
147     )

```

```

71 @Subroutine(TealType.bytes)
72 def encode_uvarint(val: Expr, b: Expr):
73     buff = ScratchVar()
74     return Seq(
75         buff.store(b),
76         Concat(
77             buff.load(),
78             If(
79                 val >= Int(128),
80                 encode_uvarint(
81                     val >> Int(7),
82                     Extract(Itob((val & Int(255)) | Int(128)), Int(7), Int(1)),
83                 ),
84                 Extract(Itob(val & Int(255)), Int(7), Int(1)),
85             ),
86         ),
87     )

```

### Affected Resource

- Constants
  - algorand/local\_blob.py (lines 27-34)
  - algorand/token\_bridge.py (lines 35-41)
- Methods
  - algorand/token\_bridge.py (encode\_uvarint lines 131-147 / get\_sig\_address lines 189-216)
  - algorand/wormhole\_core.py (encode\_uvarint lines 71-87 / get\_sig\_address lines 90-118)

### Recommendation

Refactor duplicated code, e.g., in global.py.

### Reference

N/A

## KS-RL-10 – Algorand guidelines not followed

Severity

INFORMATIONAL

### Description

One of the first Algorand recommendation is to “*Always verify that the `RekeyTo` property of any transaction is set to the `ZeroAddress` unless the contract is specifically involved in a rekeying operation*”.

Another recommendation is “*CloseRemainderTo or AssetCloseTo should be the intended recipient or equal to global `ZeroAddress`*”.

### Impact

Code should validate as many `txn` fields as possible.

If not done, things not checked could become set to anything.

### Evidence

No `Txn.rekey_to()`, `Txn.close_remainder_to()`, `Txn.asset_close_to()` check.

### Affected Resource

`pyteal/pricecaster-v2.py`

### Recommendation

Follow Algorand guidelines.

Even if other checks are implemented to validate the transactions, basic checks should be systematically applied to embody best practices.

### Reference

<https://developer.algorand.org/docs/get-details/dapps/avm/teal/guidelines/>



## KS-RL-11 – Python Code Quality

Severity	INFO
----------	------

### Description

One way of trying to define code quality is to look at one end of the spectrum: high-quality code. Hopefully, you can agree on the following high-quality code identifiers:

- It does what it is supposed to do, without any defects or problems
- It is easy to read & review, maintain, and extend

### Impact

What happens when code does not meet above requirements:

- It does **not** do what it is supposed to do
  - Meeting requirements is the basis of any product, software, including security ones.
  - Software is developed to do something. If in the end, it does not do it... what about security impacts?
- It is **difficult** to read, maintain, or extend
  - Imagine the person who wrote the original code is gone. Now it is up to you to replace that person, and to make sense of the code that is already there
  - If the code is easy to comprehend, you will be able to analyze the problem and come up with a solution much quicker. If the code is complex and convoluted, you will probably take longer and possibly make some wrong assumptions → even applied for security
  - If the code is not easy to extend, your new feature could break other things.
  - No one wants to be in the position where they have to read, maintain, or extend low-quality code.

### Evidence

One way to evaluate code is to use standard tools.

E.g., Pylint (default rcfile + disable `unused-wildcard-import`)

- pyteal/
  - mapper.py 3.90/10
  - pricecaster-v2.py 3.86/10
- algorand/
  - local\_blob.py 7.55/10
  - TmplSig.py 3.19/10
  - token\_bridge.py **0.00/10**
  - vaa\_ferify.py 1.54/10
  - wormhole\_core.py **0.00/10**

Here are samples of “unused-variable” warnings:

- `algorand/admin.py:357:18: W0612: Unused variable 'clear'`
- `algorand/admin.py:784:8: W0612: Unused variable 'bits_set'`
- `algorand/admin.py:1324:18: W0612: Unused variable 'clear'`
- `algorand/test_contract.py:47:4: W0612: Unused variable 'me'`
- `algorand/TmplSig.py:113:12: W0612: Unused variable 'admin_app_id'`

- `algorand/TmplSig.py:114:12: W0612: Unused variable 'admin_address'`
- `algorand/token_bridge.py:308:8: W0612: Unused variable 'c'`
- `algorand/token_bridge.py:309:8: W0612: Unused variable 'a'`
- `algorand/token_bridge.py:443:8: W0612: Unused variable 'me'`
- `algorand/token_bridge.py:458:8: W0612: Unused variable 'd'`
- `algorand/token_bridge.py:664:8: W0612: Unused variable 'd'`
- `algorand/token_bridge.py:825:8: W0612: Unused variable 'asset'`
- `algorand/token_bridge.py:833:8: W0612: Unused variable 'Address'`
- `algorand/token_bridge.py:834:8: W0612: Unused variable 'FromChain'`
- `algorand/wormhole_core.py:184:12: W0612: Unused variable 'emitter'`
- `algorand/wormhole_core.py:555:8: W0612: Unused variable 'progHash'`
- `algorand/wormhole_core.py:556:8: W0612: Unused variable 'progSet'`
- `algorand/wormhole_core.py:557:8: W0612: Unused variable 'clearHash'`
- `algorand/wormhole_core.py:558:8: W0612: Unused variable 'clearSet'`

→ Every time someone reads the code, they will need to understand what everything does

- Is that variable ever used for anything?

### **Affected Resource**

All python files

### **Recommendation**

Define a quality policy and enforce it using standard tools.

### **Reference**

Code audit tool for Python: <https://pypi.org/project/pylama/>

## KS-RL-12 – Inappropriate Content

Severity	INFO
----------	------

### Description

Raising an exception with the message `"you suck"` is not very helpful, nor meaningful.

## Impact

Whether for a developer, reviewer, or anyone else, what can they understand/learn from such a message?

## Evidence

```
311         if type == 'uint256' or type == 'bytes32':
312             return encode_single(type, val).hex()[64-(64):64]
313         raise Exception("you suck")

168     if chain == 5:
169         return "00000000000000000000000005a58505a96d1dbf8df91cb21b54419fc36e93fde"
170     raise Exception("you suck")
```

### **Affected Resource**

algorand/admin.py (line 313)  
algorand/gentest.py (line 170)

### Recommendation

Out of respect for the people who will read/update/use your code, please do not use inappropriate messages, comments, or any type of irrelevant information. Always remember to give useful information that could help them better understand your code.

## Reference

---

N/A

## METHODOLOGY

During this source code review, the Kudelski Security Services team reviewed code within the project within an appropriate IDE. During every review, the team spends considerable time working with the client to determine correct and expected functionality, business logic, and content to ensure that findings incorporate this business logic into each description and impact. Following this discovery phase the team works through the following categories:

- Authentication
- Authorization and Access Control
- Auditing and Logging
- Injection and Tampering
- Configuration Issues
- Logic Flaws
- Cryptography

These categories incorporate common vulnerabilities such as the OWASP Top 10

## Tools

The following tools were used during this portion of the test. A link for more information about the tool is provided as well.

- Visual Studio Code
- Pylint

## Vulnerability Scoring Systems

Kudelski Security utilizes a vulnerability scoring system based on impact of the vulnerability, likelihood of an attack against the vulnerability, and the difficulty of executing an attack against the vulnerability based on a high, medium, and low rating system

### Impact

The overall effect of the vulnerability against the system or organization based on the areas of concern or affected components discussed with the client during the scoping of the engagement.

#### High:

The vulnerability has a severe affect on the company and systems or has an affect within one of the primary areas of concern noted by the client

#### Medium:

It is reasonable to assume that the vulnerability would have a measurable affect on the company and systems that may cause minor financial or reputational damage.

#### Low:

There is little to no affect from the vulnerability being compromised. These vulnerabilities could lead to complex attacks or create footholds used in more severe attacks.

### Likelihood

The likelihood of an attacker discovering a vulnerability, exploiting it, and obtaining a foothold varies based on a variety of factors including compensating controls, location of the application, availability of commonly used exploits, and institutional knowledge

#### High:

It is extremely likely that this vulnerability will be discovered and abused

#### Medium:

It is likely that this vulnerability will be discovered and abused by a skilled attacker

#### Low:

It is unlikely that this vulnerability will be discovered or abused when discovered.

### Difficulty

Difficulty is measured according to the ease of exploit by an attacker based on availability of readily available exploits, knowledge of the system, and complexity of attack. It should be noted that a LOW difficulty results in a HIGHER severity.

#### Low:

The vulnerability is easy to exploit or has readily available techniques for exploit

#### Medium:

The vulnerability is partially defended against, difficult to exploit, or requires a skilled attacker to exploit.

#### High:

The vulnerability is difficult to exploit and requires advanced knowledge from a skilled attacker to write an exploit

### Severity

Severity is the overall score of the weakness or vulnerability as it is measured from Impact, Likelihood, and Difficulty