CERTIK

Certik Final Report For HDAC.io

Contents

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and Hdac.io (the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, CertiK's mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

For more information: https://certik.org.

Executive Summary

This report has been prepared for Hdac.io to examine issues and vulnerabilities in the source code of their system in scope. A comprehensive examination has been performed, utilizing CertiK's Static Analysis, Manual and Dynamic Review techniques.

The auditing process pays special attention to the following considerations:

- Review the security and implementation soundness of the Swap-install module.
- Review the security and implementation soundness of the Swap-logic module.
- Review the security and implementation soundness of the Swap-proxy module.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring systems logic meets the specifications and intentions of the client.

Testing Summary

SECURITY LEVEL

This report has been prepared as a product of the Audit request by The Hdac.io team.
This audit was conducted to discover issues and vulnerabilities in the source code of The Hdac.io system.

TYPE	Swap Module
SOURCE CODE	https://github.com/hdac-io/swap/releases/tag/v0.1-rc Fixes: https://github.com/hdac-io/swap/tree/v0.1-rc2
PLATFORM	Casper
LANGUAGE	Rust
REQUEST DATE	July 09, 2020
DELIVERY DATE	August 24, 2020
METHODS	A comprehensive examination has been performed using Dynamic Analysis, Static Analysis, and Manual Review.

Review Notes

Introduction

CertiK team was contracted by the Hdac.io team to audit the design and implementations of the to be released Casper based Swap module system. The audited modules:

- Swap-Install
- Swap-Logic
- Swap-Proxy

In the repository:

- https://github.com/hdac-io/swap/releases/tag/v0.1-rc

The goal of this audit is to review Hdac.io implementation for its business model, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

Summary

CertiK team has examined the code base's scope and have found **to implement the FFI functionality, language specifics, and secure design properly** with some minor exceptions regarding the panic handling in one file swap_logic/src/swap_control/ver1/rs.
More specifically there were numerous occasions that the code panicked with non optimal short error messages. This was not optimal as it would be better to handle those issues gracefully so that allocated resources could be freed with the addition of a proper error message.

Additionally the code could be more readable by eliminating repeated numeric representation of value around the code base.

All the issues that have been reported to the Hdac team were mitigated on the RC2 version.

To summarize we do believe that those minor issues did not pose a threat to the system, this view does not include the underlying framework or the total security evaluation of the project.

Recommendations

The code base uses numeric values to represent items to be fetched in collections(arguments as an example), and as arguments on functionality in many files over the code base. It would be better for readability issues that those ones are represented by variables, as the overhead is minimal but the readability upgrade of the code would be more significant.

Example swap-logic/src/lib.rs L16-22-29...

```
// Line 16
let method_name: String = runtime::get_arg(0)
// Line 22
let kyc_allowance: U512 = runtime::get_arg(1)
// Line 32
let prev_balance: U512 = runtime::get_arg(2)
// Line 70
let signature_hex_arr: Vec<String> = runtime::get_arg(3)
```

Example swap-logic/src/swap_control/mod.rs L34-55-88...

```
let new_data = UnitSnapshotData {
    prev_balance,
    is_swapped: U512::from(0),
};
```

Example swap-logic/src/swap_control/swap_storage.rs L28-34-65-68...

```
let prev_balance = U512::from_str_radix(
    unit_tree.get(keys::KEY_PREV_BALANCE_KEY)
    .unwrap_or_revert(),10,)
    .unwrap_or_default();
```

Example swap-logic/src/swap_proxy/client_api/mod.rs L53-59-66-69...

```
let method_name: String = runtime::get_arg(0)
    .unwrap_or_revert_with(ApiError::MissingArgument)
    .unwrap_or_revert_with(ApiError::InvalidArgument);
```

Additionally a lot of the functionality could return a custom Result<T, E> that implements the different errors so that we can eliminate the repeated usage of expect that leads to a panic, and make the code more readable by using the language to its full(?). This will also allow us to handle the case of an error properly instead of constantly panicking and returning short messages that will lead to maintainability issues in the future. In production systems we want to avoid panicking at all costs, as services are running it's always preferred to gracefully exit making sure that all allocations are properly freed.

Finally, although we have found testing of the modules we do believe that more code coverage and fuzzing would be beneficial for the project. Especially since we have not found fuzzing results for the underlying framework.

Findings

Exhibit 1

TITLE	TYPE	SEVERITY	LOCATION
Usage of panicking .expected	Panic	Minor	swap_logic/src/swap_control/ver1.rs L13

Description:

Usage of panicking .expected.

Recommendations:

Handle the case of failure and revert with a proper error message

Update from HDAC Team:

 Fixed in “rc2”.

Exhibit 2

TITLE	TYPE	SEVERITY	LOCATION
Usage of panicking .expected	Panic	Minor	swap_logic/src/swap_control/ver1.rs L16

Description:

Usage of panicking .expected.

Recommendations:

Handle the case of failure and revert with a proper error message

Update from HDAC Team:

 Fixed in “rc2”.

Exhibit 3

TITLE	TYPE	SEVERITY	LOCATION
Usage of panicking .expected	Panic	Minor	swap_logic/src/swap_control/ver1.rs L20

Description:

Usage of panicking .expected.

Recommendations:

Handle the case of failure and revert with a proper error message

Update from HDAC Team:

 Fixed in “rc2”.

Exhibit 4

TITLE	TYPE	SEVERITY	LOCATION
Usage of panicking .expected	Panic	Minor	swap_logic/src/swap_control/ver1.rs L26

Description:

Usage of panicking .expected.

Recommendations:

Handle the case of failure and revert with a proper error message

Update from HDAC Team:

 Fixed in “rc2”.

Exhibit 5

TITLE	TYPE	SEVERITY	LOCATION
Usage of panicking .expected	Panic	Minor	swap_logic/src/swap_control/ver1.rs L28

Description:

Usage of panicking .expected.

Recommendations:

Handle the case of failure and revert with a proper error message

Update from HDAC Team:

 Fixed in “rc2”.

Exhibit 6

TITLE	TYPE	SEVERITY	LOCATION
Usage of panicking .expected	Panic	Minor	swap_logic/src/swap_control/ver1.rs L34

Description:

Usage of panicking .expected.

Recommendations:

Handle the case of failure and revert with a proper error message

Update from HDAC Team:

 Fixed in “rc2”.

Exhibit 7

TITLE	TYPE	SEVERITY	LOCATION
Usage of panicking .expected	Panic	Minor	swap_logic/src/swap_control/ver1.rs L64

Description:

Usage of panicking .expected.

Recommendations:

Handle the case of failure and revert with a proper error message

Update from HDAC Team:

 Fixed in “rc2”.

Exhibit 8

TITLE	TYPE	SEVERITY	LOCATION
Value should be a variable	Readability	Informational	swap_logic/src/swap_control/ver1.rs L47

Description:

Value 0x28 should be declared as a variable.

Recommendations:

Handle the case of failure and revert with a proper error message

Update from HDAC Team:

 Fixed in “rc2”.