DeHacker

Code Security Assessment

Shentu Security Oracle

July 6th, 2023

Contents

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	Shentu Security Oracle
Platform	Ethereum
Website	https://www.shentu.technology/
Type	DeFi
Language	Solidity

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	0	0	0	0	0	0
Minor	0	0	0	0	0	0
Informational	14	0	0	2	0	12
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum

Findings

ID	Category	Severity	Status
DFE-01	Language Specific	Informational	Resolved
DFE-02	Language Specific	Informational	Resolved
GLOBAL-01	Language Specific	Informational	Resolved
SSO-01	Language Specific	Informational	Resolved
SSO-02	Optimization	Informational	Resolved
SSO-03	Coding Style	Informational	Resolved
SSO-04	Optimization	Informational	Resolved
SSO-05	Optimization	Informational	Acknowledged
SSO-06	Volatile Code	Informational	Resolved
SSO-07	Language Specific	Informational	Resolved
SSO-08	Language Specific	Informational	Resolved
SSO-09	Language Specific	Informational	Acknowledged
SSP-01	Coding Style	Informational	Resolved
SSP-02	Language Specific	Informational	Resolved

INFORMATIONAL

DFE-01| UNLOCKED COMPILER VERSIONS

Category	Severity	Location	Status
Language Specific	Informational	DeFiExample.sol (a09e939) ：2	Resolved

Description

An unlocked compiler version in the source code of the contract permits the user to compile it at or above aparticular version.This, in turn,leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs mayoccur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the full project can becompiled at.

INFORMATIONAL

DFE-02| DIFFERENT COMPILER VERSIONS

Category	Severity	Location	Status
Language Specific	Informational	DeFiExample.sol (a09e939) ：2	Resolved

Description

If the compiler version is between 0.4.21 and 0.4.26,then the contract raises a compilation error.

Recommendation

We advise that compiler versions below 0.5.0 should be avoided.

INFORMATIONAL

GLOBAL-01|UNLOCKED COMPILER VERSION

Category	Severity	Location	Status
Language Specific	Informational	ShentuSecurityOracle.sol (a09e939)	Resolved

Description

If the compiler version is between 0.4.21 and 0.4.26,then the contract raises a compilation error due to thekeyword payable.

Recommendation

We advise that compiler versions below 0.5.0 should be avoided.

INFORMATIONAL

SSO-01| UNLOCKED COMPILER VERSION

Category	Severity	Location	Status
Language Specific	Informational	ShentuSecurityOracle.sol (a09e939)	Resolved

Description

An unlocked compiler version in the source code of the contract permits the user to compile it at or above aparticular version.This,in turn,leads to differences in the generated bytecode between compilations due to differing compiler version numbers.This can lead to an ambiguity when debugging as compiler specific bugs mayoccur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the full project can becompiled at.

INFORMATIONAL

SSO-02| struct OPTIMIZATION

Category	Severity	Location	Status
Optimization	Informational	ShentuSecurityOracle.sol (a09e939) ：17~20	Resolved

Description

Each struct packs its members in 256-bit chunks. The Result struct contains the score (uint8) and the expiration(uint256) members,thus reserving two 256-bit chunks in storage.

Recommendation

We advise that the data type of the expiration member of the Result struct is changed to uint248, as the maximum bit-size that a datetime variable reserves are 64-bit,resulting in a one chuck storage reservation for the struct.

INFORMATIONAL

SSO-03| UNCOMMON NAMING CONVENTION

Category	Severity	Location	Status
Coding Style	Informational	ShentuSecurityOracle.sol (a09e939):25	Resolved

Description

The linked variable is prefixed with an underscore yet is declared as public .

Recommendation

We advise that the underscore is omitted per the Solidity style guide.

INFORMATIONAL

SSO-04| USE OF memory OVER storage

Category	Severity	Location	Status
Optimization	Informational	ShentuSecurityOracle.sol (a09e939):35	Resolved

Description

The linked variable is redundantly stored in storage,as storage look-ups make the gas price higher.

Recommendation

We advise the team to store the result variable in memory instead of the storage .

INFORMATIONAL

SSO-05| POTENTIAL HIGH gas OPERATION

Category	Severity	Location	Status
Optimization	Informational	ShentuSecurityOracle.sol (a09e939):72~74,105~112	Acknowledged

Description

The linked functions iteratively assign values to a mapping in storage,based on the length of an input array.

Recommendation

We advise the team to set upper boundary to the input array length.

INFORMATIONAL

SSO-06| CHECK AGAINST THE ZERO ADDRESS

Category	Severity	Location	Status
Volatile Code	Informational	ShentuSecurityOracle.sol (a09e939):31~42,79~88	Resolved

Description

The linked functions should check the value of their respective contractAddress parameter.

Recommendation

We advise the team to add a require statement to check against the zero address。

```
require(contractAddress != address(0), "Error Message");
```

INFORMATIONAL

SSO-07|DIFFERENT COMPILER VERSIONS

Category	Severity	Location	Status
Language Specific	Informational	ShentuSecurityOracle.sol (a09e939)：2	Resolved

Description

If the compiler version is between 0.4.21 and 0.4.26,then the contract raises a compilation error due to thekeyword payable.

Recommendation

We advise that compiler versions below 0.5.0 should be avoided.

INFORMATIONAL

SSO-08|MALICIOUS HASH COLLISION

Category	Severity	Location	Status
Language Specific	Informational	ShentuSecurityOracle.sol (a09e939):44~53	Resolved

Description

Since an empty bytes4 variable,i.e.0,points to the default score of a contract it is possible to have the samescore applied to a function of the contract as well.The "identifier" of a contract is simply the first 4 bytes of the keccak256 hash of the signature,meaning that an attacker would simply need to generate a function signature that results in a keccak256 hash with leading zeroes which is not an impossible achievement.

Recommendation

We advise that the default grade of a contract is either stored in a different data structure or a sanity check is put in place.

INFORMATIONAL

SSO-09|initialize PARADIGM

Category	Severity	Location	Status
Language Specific	Informational	ShentuSecurityOracle.sol (a09e939):117~121	Acknowledged

Description

The initialize function of a contract should be invokable only once via sanity checks.Here,it is possible tosubsequently call it multiple times.

Recommendation

We advise that a sanity check is imposed whereby the value of _defaultScore is ensured to be 0. Additionally we would advise a sanity check on the updateDefaultScore function that ensures the new score is not 0.

INFORMATIONAL

SSP-01| UNCOMMON NAMING CONVENTION

Category	Severity	Location	Status
Coding Style	Informational	ShentuSecurityOracleproxy.sol(a09e939):8	Resolved

Description

The linked variable is prefixed with an underscore yet is declared as public.

Recommendation

We advise that the underscore is omitted per the Solidity style guide.

INFORMATIONAL

SSP-02| DIFFERENT COMPILER VERSIONS

Category	Severity	Location	Status
Language Specific	Informational	ShentuSecurityOracleproxy.sol(a09e939):2	Resolved

Description

If the compiler version is above 0.6.0,then the contract raises a compilation error due to the fallback() function in the Proxysol file.

Recommendation

We advise that compiler versions above 0.6.0 should be avoided or change the following function of the Proxy.sol file:

```
function() external payable {
_fallback();
}
```

to

```
fallback () external payable {
_fallback();
}
```

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

 Ethereum

 Cosmos

 Eos

 Substrate

TECH STACK

 Python

 Solidity

 Rust

 c++

CONTACTS

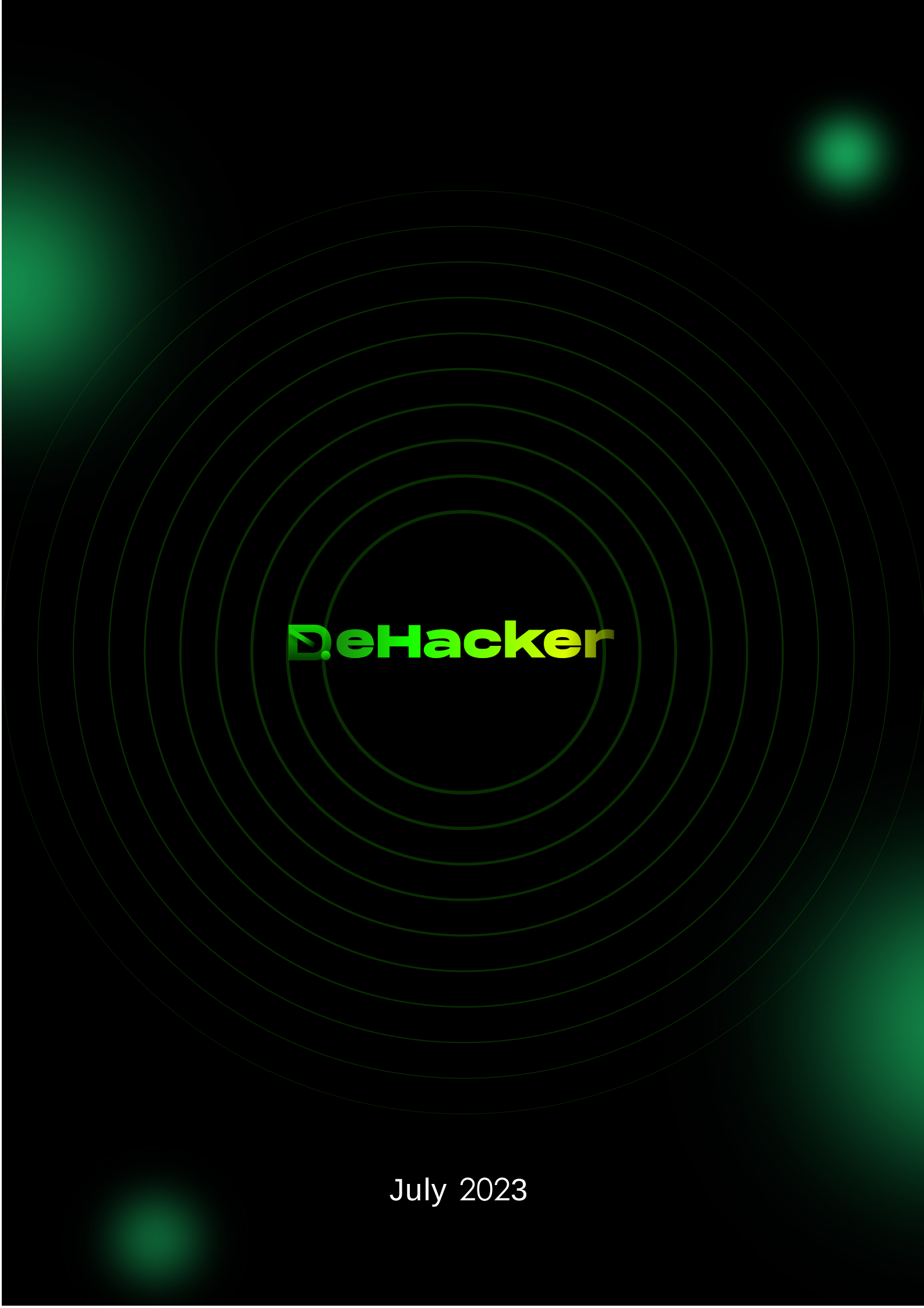
 https://dehacker.io

 https://twitter.com/dehackerio

 https://github.com/dehacker/audits_public

 https://t.me/dehackerio

 https://blog.dehacker.io/



July 2023