Acala Security Audit Report

Contents

1. Executive Summary

On Sep. 24, 2021, the SlowMist security team received the Acala team's security audit application for Acala, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The Acala Dollar stablecoin (ticker: aUSD) is a multi-collateral-backed cryptocurrency, with value stable against US Dollar (aka. 1:1 aUSD to USD soft peg). It is completely decentralized, that it can be created using assets from blockchains connected to the Polkadot network including Ethereum and Bitcoin as collaterals, and can be used by any chains (or digital jurisdictions) within the Polkadot network and applications on those chains.

SlowMist blockchain system test method:

Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code module through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs suck as nodes, SDK, etc.

In black box testing and gray box testing, we use methods such as fuzz testing and script testing to test the robustness of the interface or the stability of the components by feeding random data or constructing data with a specific structure, and to mine some boundaries Abnormal performance of the system under conditions such as bugs or abnormal performance. In white box testing, we use methods such as code review, combined with the relevant experience accumulated by the security team on known blockchain security vulnerabilities, to analyze the object definition and logic

implementation of the code to ensure that the code has the key components of the key logic. Realize no known vulnerabilities; at the same time, enter the vulnerability mining mode for new scenarios and new technologies, and find possible 0day errors.

SlowMist blockchain risk level:

Critical vulnerabilities	Critical vulnerabilities will have a significant impact on the security of the blockchain, and it is strongly recommended to fix the critical vulnerabilities.
High-risk vulnerabilities	High-risk vulnerabilities will affect the normal operation of blockchain. It is strongly recommended to fix high-risk vulnerabilities.
Medium-risk vulnerabilities	Medium vulnerability will affect the operation of blockchain. It is recommended to fix medium-risk vulnerabilities.
Low-risk vulnerabilities	Low-risk vulnerabilities may affect the operation of blockchain in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weaknesses	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Enhancement Suggestions	There are better practices for coding or architecture.

2. Scope of Audit

The main types of security audit include:

No.	Audit category	Subclass	Audit result
1	Code static check	State consistency	PASSED
		failure rollback	PASSED
		unit test	PASSED
		value overflow	Some Risk
		parameter verification	PASSED
		error unhandle	PASSED
		boundary check	PASSED
		coding specification	PASSED
7	DeFi logic security audit	-	PASSED

3 Coverage

3.1 Target Code and Revision

Source	https://github.com/AcalaNetwork/Acala

Version	ea45a0301539e2d4d6947178a0c910343ade1d64
Type	Blockchain base on substrate (https://github.com/paritytech/substrate)
Platform	Rust

3.2 Areas of Concern

Acala/modules/incentives/src/lib.rs

Acala/orml/rewards/src/lib.rs

Acala/modules/evm/src/lib.rs

Acala/modules/evm-accounts/src/lib.rs

Acala/modules/evm-manager/src/lib.rs

Acala/modules/evm/src/runner/stack.rs

Acala/modules/evm/src/runner/mod.rs

Acala/modules/evm/src/runner/state.rs

Acala/modules/evm/src/runner/storage_meter.rs

4 Analysis

Main analysis methods or steps:

- Use `#[pallet::call]` as the main entrance to check whether the input parameters of all functions are properly verified, and whether they are verified by permissions or signatures;
- Check whether all functions use unsafe writing, resulting in inconsistent node status;
- Check all arithmetic operations for overflow risks;
- Check all function calls for error handling;
- Use SAST to check whether the dependent library is safe;

check audit findings for more details.

Reference documents:

https://doc.rust-lang.org/std/

https://substrate.dev/docs/en/knowledgebase/runtime/macros#palletcall

5 Findings

Vulnerability distribution:

Critical vulnerabilities	0	
High-risk vulnerabilities	0	
Medium-risk vulnerabilities	1	■
Low-risk vulnerabilities	2	■■
Weaknesses	0	
Enhancement Suggestions	0	
Total	3	
■DeFi Logic ■Code static check ■Other		

5.1 Dependency library vulnerability[Low-risk]

Crate: hyper

Version: 0.12.36

Title: Integer overflow in `hyper`'s parsing of the `Transfer-Encoding` header leads to data loss

Date: 2021-07-07

ID: RUSTSEC-2021-0079

URL: https://rustsec.org/advisories/RUSTSEC-2021-0079

Solution: Upgrade to >=0.14.10

Dependency tree:

hyper 0.12.36

Crate: hyper

Version: 0.12.36

Title: Lenient `hyper` header parsing of `Content-Length` could allow request smuggling

Date: 2021-07-07

ID: RUSTSEC-2021-0078

URL: https://rustsec.org/advisories/RUSTSEC-2021-0078

Solution: Upgrade to >=0.14.10

Crate: hyper

Version: 0.13.10

Title: Integer overflow in `hyper`'s parsing of the `Transfer-Encoding` header leads to data loss

Date: 2021-07-07

ID: RUSTSEC-2021-0079

URL: https://rustsec.org/advisories/RUSTSEC-2021-0079

Solution: Upgrade to >=0.14.10

Dependency tree:

hyper 0.13.10

Crate: hyper

Version: 0.13.10

Title: Lenient `hyper` header parsing of `Content-Length` could allow request smuggling

Date: 2021-07-07

ID: RUSTSEC-2021-0078

URL: https://rustsec.org/advisories/RUSTSEC-2021-0078

Solution: Upgrade to >=0.14.10

Crate: libsecp256k1

Version: 0.3.5

Title: libsecp256k1 allows overflowing signatures

Date: 2021-07-13

ID: RUSTSEC-2021-0076

URL: https://rustsec.org/advisories/RUSTSEC-2021-0076

Solution: Upgrade to >=0.5.0

Dependency tree:

libsecp256k1 0.3.5

Crate: prost-types

Version: 0.7.0

Title: Conversion from `prost_types::Timestamp` to `SystemTime` can cause an overflow and panic

Date: 2021-07-08

ID: RUSTSEC-2021-0073

URL: https://rustsec.org/advisories/RUSTSEC-2021-0073

Solution: Upgrade to >=0.8.0

Dependency tree:

prost-types 0.7.0

Crate: wasmtime

Version: 0.27.0

Title: Multiple Vulnerabilities in Wasmtime

Date: 2021-09-17

ID: RUSTSEC-2021-0110

URL: https://rustsec.org/advisories/RUSTSEC-2021-0110

Solution: Upgrade to >=0.30.0

Dependency tree:

wasmtime 0.27.0

Crate: zeroize_derive

Version: 1.1.0

Title: `#[zeroize(drop)]` doesn't implement `Drop` for `enum`s

Date: 2021-09-24

ID: RUSTSEC-2021-0115

URL: https://rustsec.org/advisories/RUSTSEC-2021-0115

Solution: Upgrade to >=1.2.0

Dependency tree:

zeroize_derive 1.1.0

5.2 Risk of calculation errors[Medium-risk]

`saturating_add/saturating_sub/saturating_mul` saturating at the numeric bounds instead of overflowing, there may be some unexpected behavior.

https://doc.rust-lang.org/std/primitive.i32.html#method.saturating_add

5.3 Address not verified[Low-risk]

`new_maintainer` is not verified, it can be zero-address or other addresses that have not been mastered the private key.

- Acala/modules/evm/src/lib.rs

```
/// Sets a given contract's contract info to a new maintainer.
fn do_transfer_maintainer(who: T::AccountId, contract: EvmAddress, new_maintainer: EvmAddress) ->
DispatchResult {
Accounts::<T>::get(contract).map_or(Err(Error::<T>::ContractNotFound), |account_info| {
account_info
.contract_info
.map_or(Err(Error::<T>::ContractNotFound), |_| Ok(()))
})?;

Accounts::<T>::mutate(contract, |maybe_account_info| -> DispatchResult {
let account_info = maybe_account_info.as_mut().ok_or(Error::<T>::ContractNotFound)?;
```

```
let contract_info = account_info
.contract_info
.as_mut()
.ok_or(Error::<T>::ContractNotFound)?;

let maintainer = T::AddressMapping::get_evm_address(&who).ok_or(Error::<T>::AddressNotMapped)?;
ensure!(contract_info.maintainer == maintainer, Error::<T>::NoPermission);

contract_info.maintainer = new_maintainer;
Ok(())
})?;

Ok(())
}
```

6 Fix Log

7 Conclusion

Audit result: Fixing

Audit No. : BCA002110120001

Audit date: October 12, 2021

Audit team: SlowMist security team

Summary conclusion: All problems have been reported and are waiting for the project team to fix.

8. Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility base on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance this report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

Github
https://github.com/slowmist