Litentry-node Module Security Audit Report

Contents

1. Executive Summary 1

2. Project Background (Context) 2

3. Coverage 2

3.1 Target Code and Revision 2

4. Risk Analysis 3

4.1 Code static check 3

4.2 Design Logic Audit 6

5 Findings 12

5.1 `link_eth/link_btc` function `expiring_block_number` does not set the maximum allowed value[medium-risk] 13

5.2 `link_btc` does not support Bitcoin P2SH addresses[enhance] 13

5.3 Buffer overflow in `SmallVec::insert_many` [enhance] 13

6 Fix Log 13

7 Conclusion 14

8 Statement 14

1. Executive Summary

On Jan. 21, 2021, the SlowMist security team start security audit for Litentry-node module, 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 SlowMist security team adopts the strategy of "white box assists" to conduct a complete security test on the project in the way closest to the real attack.

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.

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. Project Background (Context)

The main types of security audit include:

No.	Audit category	Subclass	Audit result
1	Code static check	-	Pass
2	Design Logic Audit	-	Pass

(other unknown security vulnerabilities are not included in the scope of responsibility of this audit)

3. Coverage

3.1 Target Code and Revision

Source	https://github.com/litentry/litentry-node/blob/develop/pallets/account-linker/src/lib.rs

Version	Commit:894ddac535108f97996b7961578e816a79b88d08
Type	Module Audit
Platform	Rust

4. Risk Analysis

4.1 Code static check

```
$ cd pallets/account-linker/src/
$ cargo test
running 22 tests
test btc::base58::tests::test_to_base58_initial_zeros ... ok
test btc::base58::tests::test_to_base58_basic ... ok
test btc::legacy::tests::correct_btc_addr_from_pk_uncompressed ... ok
test btc::legacy::tests::correct_dhash160 ... ok
test btc::witness::tests::test_to_base32_basic ... ok
test btc::witness::tests::valid_address ... ok
test btc::base58::tests::test_to_base58_bitcoin_repo_cases ... ok
test tests::test_btc_link_p2pkh ... ok
test tests::test_invalid_block_number ... ok
test tests::test_btc_link_p2wpkh ... ok
test tests::test_insert_eth_address ... ok
test util_eth::tests::correct_recover ... ok
test tests::test_unexpected_address ... ok
test util_eth::tests::msg_with_expected_length ... ok
test util_eth::tests::msg_with_unexpected_length ... ok
test util_eth::tests::usize_to_u8_array_input_one_digit ... ok
test util_eth::tests::usize_to_u8_array_input_too_large ... ok
test util_eth::tests::usize_to_u8_array_input_two_digits ... ok
test util_eth::tests::sig_from_another_addr ... ok
test util_eth::tests::wrong_msg ... ok
test tests::test_eth_address_pool_overflow ... ok
test tests::test_update_eth_address ... ok

test result: ok. 22 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out
```

```
$ cargo audit

ID:           RUSTSEC-2021-0003
Crate:       smallvec
Version:    0.6.13
Date:        2021-01-08
URL:          https://rustsec.org/advisories/RUSTSEC-2021-0003
Title:       Buffer overflow in SmallVec::insert_many
Solution:    upgrade to >= 0.6.14, < 1.0.0 OR >= 1.6.1
Dependency tree:
smallvec 0.6.13

ID:           RUSTSEC-2021-0003
Crate:       smallvec
Version:    1.5.0
Date:        2021-01-08
URL:          https://rustsec.org/advisories/RUSTSEC-2021-0003
Title:       Buffer overflow in SmallVec::insert_many
Solution:    upgrade to >= 0.6.14, < 1.0.0 OR >= 1.6.1
Dependency tree:
smallvec 1.5.0

warning: 2 warnings found

Crate:    block-cipher
Title:    crate has been renamed to `cipher`
Date:     2020-10-15
URL:       https://rustsec.org/advisories/RUSTSEC-2020-0057
Dependency tree:
block-cipher 0.8.0

Crate:    block-cipher-trait
Title:    crate has been renamed to `block-cipher`
Date:     2020-05-26
URL:       https://rustsec.org/advisories/RUSTSEC-2020-0018
Dependency tree:
block-cipher-trait 0.6.2

Crate:    directories
Title:    directories is unmaintained, use directories-next instead
Date:     2020-10-16
```

URL: https://rustsec.org/advisories/RUSTSEC-2020-0054
Dependency tree:
directories 2.0.2

Crate: failure
Title: failure is officially deprecated/unmaintained
Date: 2020-05-02
URL: https://rustsec.org/advisories/RUSTSEC-2020-0036
Dependency tree:
failure 0.1.8

Crate: net2
Title: `net2` crate has been deprecated; use `socket2` instead
Date: 2020-05-01
URL: https://rustsec.org/advisories/RUSTSEC-2020-0016
Dependency tree:
net2 0.2.35

Crate: stream-cipher
Title: crate has been renamed to `cipher`
Date: 2020-10-15
URL: https://rustsec.org/advisories/RUSTSEC-2020-0058
Dependency tree:
stream-cipher 0.3.2

Crate: stream-cipher
Title: crate has been renamed to `cipher`
Date: 2020-10-15
URL: https://rustsec.org/advisories/RUSTSEC-2020-0058
Dependency tree:
stream-cipher 0.7.1

Crate: miow
Version: 0.2.1
Warning: package has been yanked!
Dependency tree:
miow 0.2.1

Crate: net2
Version: 0.2.35
Warning: package has been yanked!

```
error: 2 vulnerabilities found!
warning: 2 warnings found!
```

4.2 Design Logic Audit

- litentry-node/pallets/account-linker/src/lib.rs

```
#![cfg_attr(not(feature = "std"), no_std)]

use codec::Encode;
use sp_std::prelude::*;
use sp_io::crypto::{secp256k1_ecdsa_recover, secp256k1_ecdsa_recover_compressed};
use frame_support::{decl_module, decl_storage, decl_event, decl_error, dispatch, ensure};
use frame_system::{ensure_signed};
use btc::base58::ToBase58;
use btc::witness::WitnessProgram;

#[cfg(test)]
mod mock;

#[cfg(test)]
mod tests;

mod btc;
mod util_eth;

pub const MAX_ETH_LINKS: usize = 3;
pub const MAX_BTC_LINKS: usize = 3;

pub trait Trait: frame_system::Trait {
type Event: From<Event<Self>> + Into<<Self as frame_system::Trait>::Event>;
}

enum BTCAddrType {
Legacy,
Segwit,
}
```

```
decl_storage! {
trait Store for Module<T: Trait> as AccountLinkerModule {
pub EthereumLink get(fn eth_addresses): map hasher(blake2_128_concat) T::AccountId => Vec<[u8; 20]>;
pub BitcoinLink get(fn btc_addresses): map hasher(blake2_128_concat) T::AccountId => Vec<Vec<u8>>;
}
}

decl_event!(
pub enum Event<T>
where
AccountId = <T as frame_system::Trait>::AccountId,
{
SomethingStored(u32, AccountId),
}
);

decl_error! {
pub enum Error for Module<T: Trait> {
EcdsaRecoverFailure,
LinkRequestExpired,
UnexpectedAddress,
// Unexpected ethereum message length error
UnexpectedEthMsgLength,
InvalidBTCAddress,
InvalidBTCAddressLength,
}
}

decl_module! {
pub struct Module<T: Trait> for enum Call where
origin: T::Origin {
// Errors must be initialized if they are used by the pallet.
type Error = Error<T>;

// Events must be initialized if they are used by the pallet.
fn deposit_event() = default;
```

```
/// separate sig to r, s, v because runtime only support array parameter with length <= 32
#[weight = 1]
pub fn link_eth(
origin,
account: T::AccountId,
index: u32,
addr_expected: [u8; 20],
expiring_block_number: T::BlockNumber,
r: [u8; 32],
s: [u8; 32],
v: u8,
) -> dispatch::DispatchResult {

let _ = ensure_signed(origin)?;

let current_block_number = <frame_system::Module<T>>::block_number();
ensure!(expiring_block_number > current_block_number, Error::<T>::LinkRequestExpired);
//SlowMist// expiring_block_number does not set the maximum offset value

// TODO: there is no eth prefix here
let mut bytes = b"Link Litentry: ".encode();
let mut account_vec = account.encode();
let mut expiring_block_number_vec = expiring_block_number.encode();

bytes.append(&mut account_vec);
bytes.append(&mut expiring_block_number_vec);

let hash = util_eth::eth_data_hash(bytes).map_err(|_| Error::<T>::UnexpectedEthMsgLength)?;

let mut msg = [0u8; 32];
let mut sig = [0u8; 65];

msg[..32].copy_from_slice(&hash[..32]);
sig[..32].copy_from_slice(&r[..32]);
sig[32..64].copy_from_slice(&s[..32]);
sig[64] = v;

let addr = util_eth::addr_from_sig(msg, sig)
```

```
.map_err(|_| Error::<T>::EcdsaRecoverFailure)?; //SLowMist// Recover the Ethereum address from the signature, the content of the signature is accointid and expired blockensure!(addr == addr_expected, Error::<T>::UnexpectedAddress);

let index = index as usize;
let mut addrs = Self::eth_addresses(&account);
// NOTE: allow linking `MAX_ETH_LINKS` eth addresses.
if (index >= addrs.len()) && (addrs.len() != MAX_ETH_LINKS) {
addrs.push(addr);
} else if (index >= addrs.len()) && (addrs.len() == MAX_ETH_LINKS) {
addrs[MAX_ETH_LINKS - 1] = addr;
} else {
addrs[index] = addr;
}

<EthereumLink<T>>::insert(account, addrs);

Ok(())

}

/// separate sig to r, s, v because runtime only support array parameter with length <= 32
#[weight = 1]
pub fn link_btc(
origin,
account: T::AccountId,
index: u32,
addr_expected: Vec<u8>,
expiring_block_number: T::BlockNumber,
r: [u8; 32],
s: [u8; 32],
v: u8,
) -> dispatch::DispatchResult {

let _ = ensure_signed(origin)?;

let current_block_number = <frame_system::Module<T>>::block_number();
```

```
ensure!(expiring_block_number > current_block_number, Error::<T>::LinkRequestExpired);

// TODO: we may enlarge this 2
if addr_expected.len() < 2 {
Err(Error::<T>::InvalidBTCAddressLength)?
}

let addr_type = if addr_expected[0] == b'1' {
BTCAddrType::Legacy
} else if addr_expected[0] == b'b' && addr_expected[1] == b'c' { // TODO: a better way?
BTCAddrType::Segwit
} else {
Err(Error::<T>::InvalidBTCAddress)? //SLowMist// Supports P2PKH, Bech32 addresses, but does not support P2SH addresses
};

let mut bytes = b"Link Litentry: ".encode();
let mut account_vec = account.encode();
let mut expiring_block_number_vec = expiring_block_number.encode(); //SlowMist// expiring_block_number does not set the maximum offset value

bytes.append(&mut account_vec);
bytes.append(&mut expiring_block_number_vec);

// TODO: seems btc uses sha256???
let hash = sp_io::hashing::keccak_256(&bytes);

let mut msg = [0u8; 32];
let mut sig = [0u8; 65];

msg[..32].copy_from_slice(&hash[..32]);
sig[..32].copy_from_slice(&r[..32]);
sig[32..64].copy_from_slice(&s[..32]);
sig[64] = v;

// currently both compressed and uncompressed is ok for legacy address
// need to also modify the test
```

```
// let pk_no_prefix = secp256k1_ecdsa_recover(&sig, &msg)
// .map_err(|_| Error::<T>::EcdsaRecoverFailure)?;

let pk = secp256k1_ecdsa_recover_compressed(&sig, &msg)
.map_err(|_| Error::<T>::EcdsaRecoverFailure)?;

let mut addr = Vec::new();

match addr_type {
BTCAddrType::Legacy => {
// let mut pk = [0u8; 65];

// // pk prefix = 4
// pk[0] = 4;
// pk[1..65].copy_from_slice(&pk_no_prefix);

// addr = btc::legacy::btc_addr_from_pk_uncompressed(pk).to_base58();

addr = btc::legacy::btc_addr_from_pk_compressed(pk).to_base58();
},
// Native P2WPKH is a scriptPubKey of 22 bytes.
// It starts with a OP_0, followed by a canonical push of the keyhash (i.e. 0x0014{20-byte keyhash})
// keyhash is RIPEMD160(SHA256) of a compressed public key
// https://bitcoincore.org/en/segwit_wallet_dev/
BTCAddrType::Segwit => {
let pk_hash = btc::legacy::hash160(&pk);
let mut pk = [0u8; 22];
pk[0] = 0;
pk[1] = 20;
pk[2..].copy_from_slice(&pk_hash);
let wp = WitnessProgram::from_scriptpubkey(&pk.to_vec()).map_err(|_|
Error::<T>::InvalidBTCAddress)?;
addr = wp.to_address(b"bc".to_vec()).map_err(|_| Error::<T>::InvalidBTCAddress)?;
}
}

ensure!(addr == addr_expected, Error::<T>::UnexpectedAddress);
```

```
let index = index as usize;
let mut addrs = Self::btc_addresses(&account);
// NOTE: allow linking `MAX_BTC_LINKS` btc addresses.
if (index >= addrs.len()) && (addrs.len() != MAX_BTC_LINKS) {
addrs.push(addr);
} else if (index >= addrs.len()) && (addrs.len() == MAX_BTC_LINKS) {
addrs[MAX_BTC_LINKS - 1] = addr;
} else {
addrs[index] = addr;
}

<BitcoinLink<T>>::insert(account, addrs);

Ok(())

}

}
}
```

5 Findings

Vulnerability distribution:

Critical vulnerabilities	0	
High-risk vulnerabilities	0	
Medium-risk vulnerablities	1	■
Low-risk vulnerabilities	0	
Weaknesses	0	
Enhancement Suggestions	2	■■

Total	3	
■Code static ■Design logic ■Other		

5.1 `link_eth/link_btc` function `expiring_block_number` does not set the maximum allowed value[medium-risk]

If someone sets a block expiration that is too large, the signature will almost never expire, it may cause some security issues.

5.2 `link_btc` does not support Bitcoin P2SH addresses[enhance]

It supports Bitcoin P2PKH and Bech32 addresses, but does not support P2SH addresses. It should be enhanced to fully cover the Bitcoin protocol.

5.3 Buffer overflow in `SmallVec::insert_many` [enhance]

Buffer overflow in `SmallVec::insert_many`, but function `insert_many` does not been included in this audit module.

6 Fix Log

- Patch PR: https://github.com/litentry/litentry-node/pull/30

- Patch description: Set the maximum number of `expiring_block_number`.

7 Conclusion

Audit result: PASS

Audit No. : BCA002101220001

Audit date: January 22, 2021

Audit team: SlowMist security team

Summary conclusion: After correction, all problems found have been fixed and the above risks have been eliminated by litentry-node. Comprehensive assessed, litentry-node has no risks above already.

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