Blockchain Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.10.21, the SlowMist security team received the Netmarble team's security audit application for semita-netmarble-template-bsc, 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 "black, grey box lead, white box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules 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 such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Level	Description
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for the chain includes two steps:

Chain codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The codes are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the chain:

NO.	Audit Items	Result
1	[Encryption]Insecure entropy source audit	Passed
2	[Encryption]Keystore Encryption Audit	Some Risks
3	[Encryption]Cryptographic Attack Audit	Passed
4	[Encryption]Transaction Malleability Attack	Some Risks
5	[Ledger]Transaction Replay Attack	Some Risks
6	[Off-Chain]False Top-up Audit	Some Risks
7	[RPC]The Ethereum Black Valentine's Day Vulnerability	Passed
8	[Supply Chain]Code Diff	Passed

3 Project Overview

3.1 Project Introduction

BSC Application Sidechain (BAS) is an infrastructure for developers that helps them to build large scale BSC-based apps with higher throughput and much lower or even zero transaction fees.

3.2 Coverage

Target Code and Revision:

https://github.com/node-real/semita-netmarble-template-bsc

Version: netmarble_v1.0

[Encryption]Insecure entropy source audit

The generation of the private key seed is based on the `crypto/rand` standard library, and the entropy value is secure.

crypto/crypto.go

```
func GenerateKey() (*ecdsa.PrivateKey, error) {
return ecdsa.GenerateKey(S256(), rand.Reader)
}
```

[Encryption]Keystore Encryption Audit

cmd/ethkey/generate.go

```
// Encrypt key with passphrase.
passphrase := getPassphrase(ctx, true)
scryptN, scryptP := keystore.StandardScryptN, keystore.StandardScryptP
if ctx.Bool("lightkdf") {
    scryptN, scryptP = keystore.LightScryptN, keystore.LightScryptP
}
keyjson, err := keystore.EncryptKey(key, passphrase, scryptN, scryptP)
if err != nil {
    utils.Fatalf("Error encrypting key: %v", err)
}
```

[Encryption]Cryptographic Attack Audit

Hash:

crypto/sha256

github.com/ethereum/go-ethereum/crypto/blake2b

golang.org/x/crypto/sha3

Signature:

crypto/ed25519

github.com/ethereum/go-ethereum/crypto/secp256k1

Encrypt:

golang.org/x/crypto/pbkdf2

Use well known cryptographic algorithms.

[Encryption]Transaction Malleability Attack

Allowing transactions with any s value with 0 < s < secp256k1n, as is currently the case, opens a transaction malleability concern, as one can take any transaction, flip the s value from s to secp256k1n - s, flip the v value (27 -> 28, 28 -> 27), and the resulting signature would still be valid. This is not a serious security flaw, especially since Ethereum uses addresses and not transaction hashes as the input to an ether value transfer or other transaction, but it nevertheless creates a UI inconvenience as an attacker can cause the transaction that gets confirmed in a block to have a different hash from the transaction that any user sends, interfering with user interfaces that use transaction hashes as tracking IDs.

[Ledger]Transaction Replay Attack

(a). Replay transaction on origin chain:

Each transaction is signed with a unique `nonce` value, duplicate transaction not be packed into block again.

(b). Replay transaction on forked chain:

Use `ChainID` to distinguish different chains when signing transactions, and there is no replay attack problem for transactions between different chains.

[Off-Chain]False Top-up Audit

Using EVM, an attacker can deploy evil contracts to launch attacks.

[RPC]The Ethereum Black Valentine's Day Vulnerability

If insecure account unlocking is not allowed if node's APIs are exposed to external.

cmd/geth/main.go

```
        isDevMode := false
        if ctx.GlobalIsSet(utils.NetworkIdFlag.Name) &&
ctx.GlobalUint64(utils.NetworkIdFlag.Name) == 1337 {
                isDevMode = true
        }
        if !stack.Config().InsecureUnlockAllowed && stack.Config().ExtRPCEnabled() &&
!isDevMode {
                utils.Fatalf("Account unlock with HTTP access is forbidden!")
        }
```

[Supply Chain]Code Diff

File diff with bsc (https://github.com/bnb-chain/bsc/tree/v1.1.5):

```
accounts/abi/bind/backends/simulated.go
cmd/blockdump/main.go
cmd/clef/main.go
cmd/devp2p/discv4cmd.go
cmd/devp2p/nodesetcmd.go
cmd/evm/internal/t8ntool/execution.go
cmd/evm/internal/t8ntool/flags.go
cmd/evm/internal/t8ntool/transition.go
cmd/evm/disasm.go
cmd/evm/main.go
cmd/evm/runner.go
```

```
cmd/evm/staterunner.go
cmd/faucet/faucet.go
cmd/faucet/website.go
cmd/geth/accountcmd.go
cmd/geth/chaincmd.go
cmd/geth/config.go
cmd/geth/consolecmd.go
cmd/geth/dbcmd.go
cmd/geth/main.go
cmd/geth/misccmd.go
cmd/geth/snapshot.go
cmd/geth/usage.go
cmd/utils/flags.go
common/systemcontract/const.go
consensus/clique/clique.go
consensus/ethash/consensus.go
consensus/parlia/parlia.go
consensus/parlia/snapshot.go
core/rawdb/chain_iterator.go
core/rawdb/database.go
core/rawdb/freezer.go
core/rawdb/schema.go
core/rawdb/table.go
core/state/pruner/pruner.go
core/state/snapshot/difflayer.go
core/state/snapshot/disklayer.go
core/state/snapshot/journal.go
core/state/snapshot/snapshot.go
core/state/database.go
core/state/statedb.go
core/state/trie_prefetcher.go
core/types/transaction.go
core/vm/runtime/runtime.go
core/vm/access_list_tracer.go
core/vm/contracts.go
core/vm/evm.go
core/vm/instructions.go
core/vm/interpreter.go
core/vm/logger.go
core/vm/logger_json.go
core/vm/stack.go
core/block_validator.go
core/blockchain.go
core/chain_makers.go
```

```
core/error.go
core/events.go
core/gen_genesis.go
core/gen_genesis_account.go
core/genesis.go
core/state_prefetcher.go
core/state_processor.go
core/tx_pool.go
core/types.go
eth/downloader/downloader.go
eth/ethconfig/config.go
eth/ethconfig/gen_config.go
eth/protocols/diff/handshake.go
eth/protocols/snap/handler.go
eth/abi.go
eth/api_backend.go
eth/backend.go
eth/handler.go
eth/peerset.go
eth/state_accessor.go
eth/sync.go
ethdb/database.go
internal/debug/flags.go
internal/ethapi/api.go
internal/web3ext/web3ext.go
les/api_backend.go
les/client.go
les/commons.go
miner/worker.go
node/node.go
p2p/dnsdisc/sync.go
p2p/peer.go
p2p/server.go
params/config.go
params/protocol_params.go
params/version.go
rlp/safe.go
tests/fuzzers/bls12381/bls12381_fuzz.go
tests/init.go
trie/database.go
trie/proof.go
trie/trie.go
```

Mainly functional changes, no malicious code introduced, no special administrator privileges backdoor added.

3.3 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Weak passphrase	[Encryption]Keystore Encryption Audit	Suggestion	Confirming
N2	Transaction malleability risk	[Encryption]Transaction Malleability Attack	Suggestion	Confirming
N3	"False top up" risks	[Off-Chain]False Top-up Audit	Low	Confirming
N4	Default `ChainConfig` incorrect	[Ledger]Transaction Replay Attack	Suggestion	Confirming

4 Findings

4.1 Vulnerability Summary

[N1] [Suggestion] Weak passphrase

Category: [Encryption]Keystore Encryption Audit

Content

The private key is encrypted using keystore and the strength of the password is not verified. A weak passphrase like `123456` can be used in the test, which can be easily cracked.

Solution

Detect password strength.

Status

Confirming

[N2] [Suggestion] Transaction malleability risk

Category: [Encryption]Transaction Malleability Attack

Content

Preventing high `s` values reduces the risk, but `secp256k1` is still at risk of malleability.

Vulnerability reference:

https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2.md

Solution

When a withdrawal transaction fails, the original transaction needs to be rebroadcast or repackaged with the same `nonce` to prevent malicious repeated withdrawals.

Status

Confirming

[N3] [Low] "False top up" risks

Category: [Off-Chain]False Top-up Audit

Content

There are 2 kinds of "False top up" risks:

(a). When using a contract for native token top-up, the transfer process may be aborted by a malicious contract revert, but the transaction status is successful, which may result in a false top-up if no valid event judgment is performed.

(b). Failed transactions are also packed into blocks, and if the status of the transaction is not checked, the exchange will have a "false top-up" problem.

Solution

Exchanges, etc. need to check whether the status of the transaction is successful and determine whether the event

is correct, it is recommended to prohibit the use of contract top-up.

Status

Confirming

[N4] [Suggestion] Default `ChainConfig` incorrect

Category: [Ledger]Transaction Replay Attack

Content

params/config.go

```
AllEthashProtocolChanges = &ChainConfig{
    big.NewInt(1337),
//...
AllCliqueProtocolChanges = &ChainConfig{
    big.NewInt(1337),
//...
TestChainConfig = &ChainConfig{
    big.NewInt(1),
```

The default mainnet or testnet configuration is incorrect and the user should pass the configuration through the command line.

Solution

Correct default configuration.

Status

Confirming

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002210310001	SlowMist Security Team	2022.10.21 - 2022.10.31	Low Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 low risk, 3 suggestion vulnerabilities.

6 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 based 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 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