SLOWMIST

AnySwap CrossChain-Bridge Security Audit Report

Contents

1 Executive Summary

On August 10, 2020, the SlowMist security team received the AnySwap team's security audit application for AnySwap CrossChain-Bridge, 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 lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

SlowMist project 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 such as nodes, SDK, etc.

SlowMist project risk level:

Critical vulnerabilities	Critical vulnerabilities will have a significant impact on the security of the project, and it is strongly recommended to fix the critical vulnerabilities.
High-risk vulnerabilities	High-risk vulnerabilities will affect the normal operation of project. It is strongly recommended to fix high-risk vulnerabilities.
Medium-risk vulnerabilities	Medium vulnerability will affect the operation of project. It is recommended to fix medium-risk vulnerabilities.

Low-risk vulnerabilities	Low-risk vulnerabilities may affect the operation of project 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.

SlowMist security team audit process for AnySwap CrossChain-Bridge project includes two steps:

- Codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house automated analysis tools.
- Manual audit of the codes for security issues. The code are manually analyzed to look for any potential problems.

The following is a list of audit items that the SlowMist security team will focus on during the white box audit process:

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

- Communication Security Audit
- Golang Packages Security Audit
- Multi-threading Security Audit
- Cryptography Security Audit
- RPC Interface Security Audit
- nil Pointer Security Audit
- Business Logic Vulnerability Audit
- DoS Vulnerability Audit
- Sensitive Information Leakage Audit
- Log Processing Security Audit
- Cached Information Security Audit
- Access Control Security Audit
- Configuration file Security Audit

- Uncaught Exception Handling Audit
- Recursion and Loop Security Audit
- Database Query Security Audit

2 Project Background

2.1 Project Introduction

AnySwap CrossChain-Bridge is the Cross-chain Decentralized Swap Market Place. It operates on the Fusion blockchain and allows users to trade tokens by swapping seamlessly between pairs.

This audit conducted a security audit on the AnySwap CrossChain-Bridge code, excluding the dcrm module.

The following is the relevant file information:

1. Audit version code

https://github.com/fsn-dev/crossChain-Bridge
commit: 590ca12ed7fe5e767f6b7fa04e46df9e870de34d

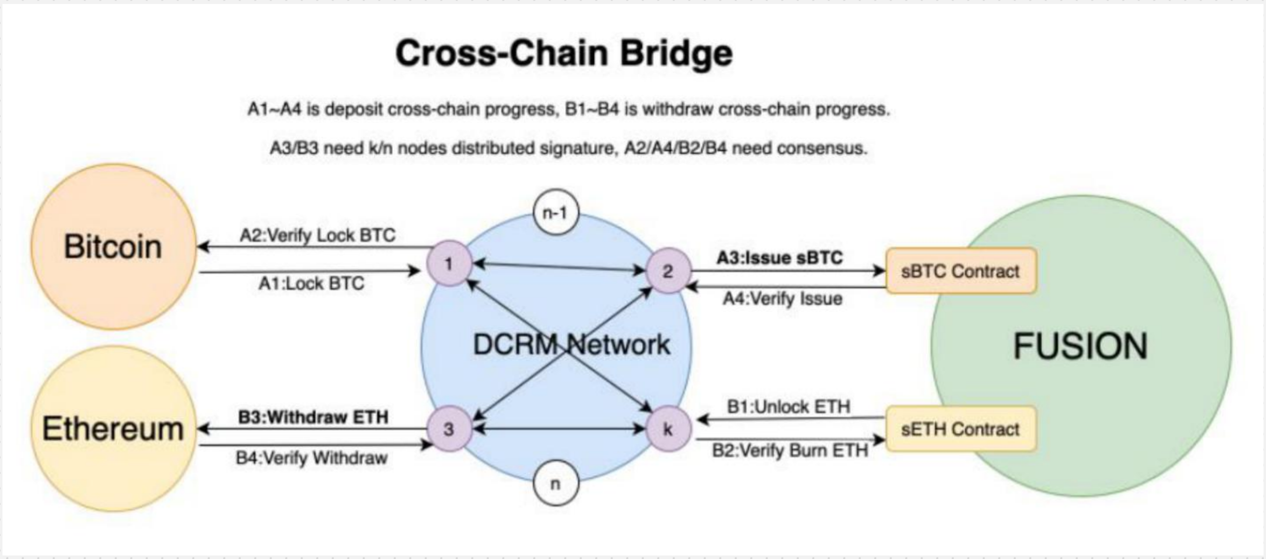
2. Fixed version code

https://github.com/fsn-dev/crossChain-Bridge
commit: 69b78a5f6f7ac0c36a3704dfcd7a52d089ac2e31

3. CrossChain Bridge design document
https://github.com/fsn-dev/crossChain-Bridge/wiki/CrossChain-Bridge-Design#%E6%96%B9%E6%A1%88%E6%A6%82%E8%BF%B0

4. Directory description
crosschain-bridge.txt
MD5: 31e2cfafdd28f6fc7bf9cba1b88a06e9

2.2 Architecture diagram



3 Code Audit

3.1 Communication Security Audit

The communication of services such as RPC and API needs to be configured in " build/bin/config.toml". Need to pay attention to the security of communication with RPC. It is recommended to use https for communication.

- build/bin/config.toml

```
# risk control config
InitialDiffValue = 81000.0
MaxAuditDiffValue = 100.0
MinWithdrawReserve = 10000.0
 [Email]
Server = "smtp.gmail.com"
Port = 25
From = "from@gmail.com"
FromName = "Risk Control"
Password = "***"
To = ["to1@gmail.com", "to2@gmail.com"]
```

```
Cc = ["cc1@gmail.com", "cc2@gmail.com"]


# source token config[SrcToken]
BlockChain = "Ethereum"
NetID = "Rinkeby"
ID = "ERC20"
Name = "USDTERC20"
Symbol = "USDT"
Decimals = 6
Description = "ERC20 USDT"
ContractAddress = "0xb09bad01684f6d47fc7dc9591889cc77eaed8d22"
DepositAddress = "0x1249E24A077A1d95254DB00ba406100fBca8003F"
DcrmAddress = "0xb09bad01684f6d47fc7dc9591889cc77eaed8d22"


# source blockchain gateway config[SrcGateway]
APIAddress = ["http://5.189.139.168:8018"]


# dest token config[DestToken]
BlockChain = "Fusion"
NetID = "Testnet"
ID = "mUSDT"
Name = "SMPC ERC20 USDT"
Symbol = "mUSDT"
Decimals = 6
Description = "cross chain bridge USDT with mUSDT"
ContractAddress = "0x13edb3a93a985d5db4b4965d38c446ed7a931c53"
DcrmAddress = "0x826197183eB1a7D71DB073216ce157886680B094"


# dest blockchain gateway config[DestGateway]
APIAddress = ["https://testnet.fsn.dev/api"]
```

The RPC service does not use https for communication. It is recommended to use Nginx for proxy forwarding and https for communication.

- rpc/server/server.go

```
func StartAPIServer() {
        router := initRouter()
        apiPort := params.GetAPIPort()
        apiServer := params.GetConfig().APIServer
        allowedOrigins := apiServer.AllowedOrigins

        corsOptions := []handlers.CORSOption{
                handlers.AllowedMethods([]string{"GET", "POST"}),
        }
        if len(allowedOrigins) != 0 {
                corsOptions = append(corsOptions,
                        handlers.AllowedHeaders([]string{"X-Requested-With", "Content-Type"}),
                        handlers.AllowedOrigins(allowedOrigins),
                )
        }

        log.Info("JSON RPC service listen and serving", "port", apiPort, "allowedOrigins", allowedOrigins)
        svr := http.Server{
                Addr:         fmt.Sprintf(":%v", apiPort),
                ReadTimeout:  60 * time.Second,
                WriteTimeout: 60 * time.Second,
                Handler:      handlers.CORS(corsOptions...)(router),
        }
        go func() {
                if err := svr.ListenAndServe(); err != nil {
                        log.Error("ListenAndServe error", "err", err)
                }
        }()
```

3.2 Golang Packages Security Audit

The project's dependent packages were audited, and no packages with known vulnerabilities were found.

- go.mod

```
module github.com/anyswap/CrossChain-Bridge

go 1.14
```

```
require (
            github.com/BurntSushi/toml v0.3.1
            github.com/btcsuite/btcd v0.20.1-beta
            github.com/btcsuite/btcutil v1.0.2
            github.com/btcsuite/btcwallet/wallet/txauthor v1.0.0
            github.com/btcsuite/btcwallet/wallet/txrules v1.0.0
            github.com/btcsuite/btcwallet/wallet/txsizes v1.0.0
            github.com/dvyukov/go-fuzz v0.0.0-20200805095026-2dc2d88eb660 // indirect
            github.com/ethereum/go-ethereum v1.9.18
            github.com/fastly/go-utils v0.0.0-20180712184237-d95a45783239 // indirect
            github.com/gorilla/handlers v1.4.2
            github.com/gorilla/mux v1.7.4
            github.com/gorilla/rpc v1.2.0
            github.com/jehiah/go-strftime v0.0.0-20171201141054-1d33003b3869 // indirect
            github.com/jonboulle/clockwork v0.2.0 // indirect
            github.com/lestrrat-go/file-rotatelogs v2.3.0+incompatible
            github.com/lestrrat-go/strftime v1.0.1 // indirect
            github.com/pborman/uuid v1.2.0
            github.com/pkg/errors v0.9.1 // indirect
            github.com/sirupsen/logrus v1.5.0
            github.com/stretchr/testify v1.5.1
            github.com/tebeka/strftime v0.1.5 // indirect
            github.com/urfave/cli/v2 v2.2.0
            golang.org/x/crypto v0.0.0-20200622213623-75b288015ac9
            gopkg.in/mgo.v2 v2.0.0-20190816093944-a6b53ec6cb22)
```

3.3 Multi-threading Security Audit

During the audit, no slice security issues and map read and write security issues in the concurrent process were found.

3.4 Cryptography Security Audit

This audit does not include dcrm modules.

3.5 RPC Interface Security Audit

The APIs exposed to the outside were tested, and no security issues were found during the audit.

- rpc/rpcapi/api.go

```
GetVersionInfo(r *http.Request, args *RPCNullArgs, result *string) error

GetServerInfo(r *http.Request, args *RPCNullArgs, result *swapapi.ServerInfo) error

GetSwapStatistics(r *http.Request, args *RPCNullArgs, result *swapapi.SwapStatistics) error

GetRawSwapin(r *http.Request, txid *string, result *swapapi.Swap) error

GetRawSwapinResult(r *http.Request, txid *string, result *swapapi.SwapResult) error

GetSwapin(r *http.Request, txid *string, result *swapapi.SwapInfo) error

GetRawSwapout(r *http.Request, txid *string, result *swapapi.Swap) error

GetRawSwapoutResult(r *http.Request, txid *string, result *swapapi.SwapResult) error

GetSwapout(r *http.Request, txid *string, result *swapapi.SwapInfo) error

GetSwapinHistory(r *http.Request, args *RPCQueryHistoryArgs, result *[]*swapapi.SwapInfo) error

GetSwapoutHistory(r *http.Request, args *RPCQueryHistoryArgs, result *[]*swapapi.SwapInfo) error

Swapin(r *http.Request, txid *string, result *swapapi.PostResult) error

RetrySwapin(r *http.Request, txid *string, result *swapapi.PostResult) error

P2shSwapin(r *http.Request, args *RPCP2shSwapinArgs, result *swapapi.PostResult) error

Swapout(r *http.Request, txid *string, result *swapapi.PostResult) error

IsValidSwapinBindAddress(r *http.Request, address *string, result *bool) error

IsValidSwapoutBindAddress(r *http.Request, address *string, result *bool) error

RegisterP2shAddress(r *http.Request, bindAddress *string, result *tokens.P2shAddressInfo) error

GetP2shAddressInfo(r *http.Request, p2shAddress *string, result *tokens.P2shAddressInfo) error

GetLatestScanInfo(r *http.Request, isSrc *bool, result *swapapi.LatestScanInfo) error

RegisterAddress(r *http.Request, address *string, result *swapapi.PostResult) error

GetRegisteredAddress(r *http.Request, address *string, result *swapapi.RegisteredAddress) error

BuildSwapoutTx(r *http.Request, args *BuildSwapoutTxArgs, result *types.Transaction) error

AdminCall(r *http.Request, rawTx *string, result *string) error
```

Call the Ethereum API to get information.

tokens/eth/callapi.go

```
SendSignedTransaction -> eth_sendRawTransaction

ChainID       ->          eth_chainId

NetworkID     ->          net_version

GetCode       ->          eth_getCode

CallContract  ->          eth_call
```

```
tokens/eth/callcontract.go
GetErc20Balance         ->          balanceOf
GetErc20Decimals        ->          decimals
```

Call Bitcoin API to get information.

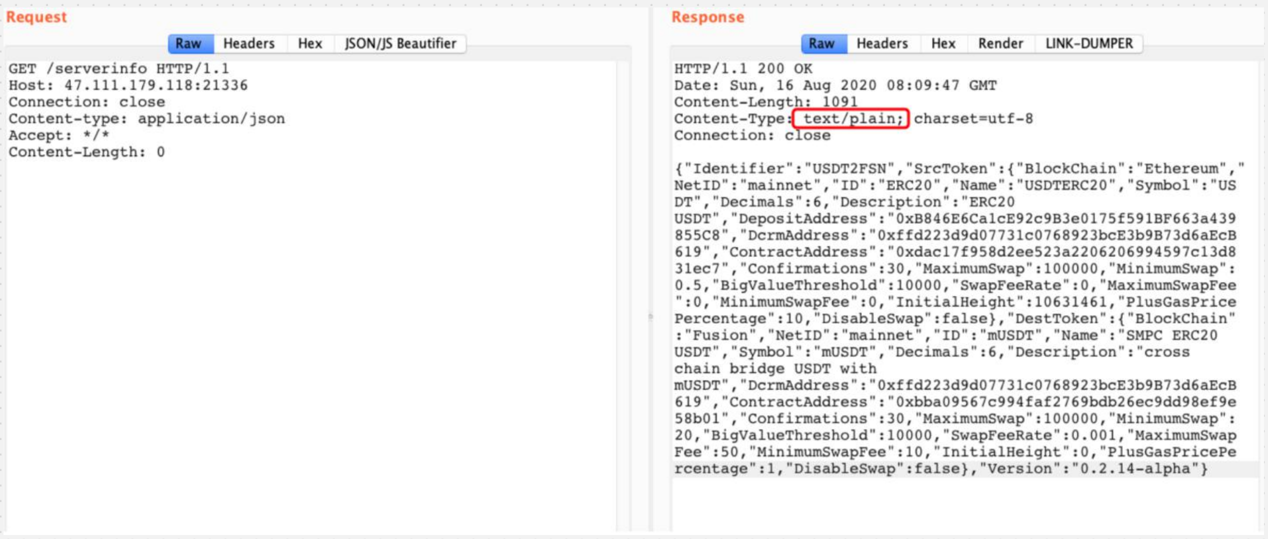
- tokens/btc/electrs/callapi.go

```
GetLatestBlockNumber    ->          apiAddress + "/blocks/tip/height"
GetTransactionByHash    ->          apiAddress + "/tx/" + txHash
GetElectTransactionStatus           ->          apiAddress + "/tx/" + txHash + "/status"
FindUtxos   ->          apiAddress + "/address/" + addr + "/utxo"
GetPoolTxidList         ->          apiAddress + "/mempool/txids"
GetPoolTransactions     ->          apiAddress + "/address/" + addr + "/txs/mempool"
GetTransactionHistory   ->          apiAddress + "/address/" + addr + "/txs/chain"
GetOutspend             ->          apiAddress + "/tx/" + txHash + "/outspend/" + fmt.Sprintf("%d", vout)
PostTransaction         ->          apiAddress + "/tx"
GetBlockHash            ->          apiAddress + "/block-height/" + fmt.Sprintf("%d", height)
GetBlockTxids           ->          apiAddress + "/block/" + blockHash + "/txids"
```

The response header set in the code is "application/json", but in fact it returns "text/plain". Please check this bug.

- rpc/restapi/api.go

```
func writeResponse(w http.ResponseWriter, resp interface{}, err error) {
        if err == nil {
                jsonData, _ := json.Marshal(resp)
                w.Header().Set("Content-Type", "application/json")
                _, _ = w.Write(jsonData)
        } else {
                fmt.Fprintln(w, err.Error())
        }}
```


Request

Raw Headers Hex JSON/JS Beautifier

```
GET /serverinfo HTTP/1.1
Host: 47.111.179.118:21336
Connection: close
Content-type: application/json
Accept: */*
Content-Length: 0
```

Response

Raw Headers Hex Render LINK-DUMPER

```
HTTP/1.1 200 OK
Date: Sun, 16 Aug 2020 08:09:47 GMT
Content-Length: 1091
Content-Type: text/plain; charset=utf-8
Connection: close

{"Identifier":"USDT2FSN","SrcToken":{"BlockChain":"Ethereum","
NetID":"mainnet","ID":"ERC20","Name":"USDTERC20","Symbol":"US
DT","Decimals":6,"Description":"ERC20
USDT","DepositAddress":"0xB846E6CalcE92c9B3e0175f591BF663a439
855C8","DcrmAddress":"0xffd223d9d07731c0768923bcE3b9B73d6aEcB
619","ContractAddress":"0xdac17f958d2ee523a2206206994597c13d8
31ec7","Confirmations":30,"MaximumSwap":100000,"MinimumSwap":
0.5,"BigValueThreshold":10000,"SwapFeeRate":0,"MaximumSwapFee
":0,"MinimumSwapFee":0,"InitialHeight":10631461,"PlusGasPrice
Percentage":10,"DisableSwap":false},"DestToken":{"BlockChain"
:"Fusion","NetID":"mainnet","ID":"mUSDT","Name":"SMPC ERC20
USDT","Symbol":"mUSDT","Decimals":6,"Description":"cross
chain bridge USDT with
mUSDT","DcrmAddress":"0xffd223d9d07731c0768923bcE3b9B73d6aEcB
619","ContractAddress":"0xbba09567c994faf2769bdb26ec9dd98ef9e
58b01","Confirmations":30,"MaximumSwap":100000,"MinimumSwap":
20,"BigValueThreshold":10000,"SwapFeeRate":0.001,"MaximumSwap
Fee":50,"MinimumSwapFee":10,"InitialHeight":0,"PlusGasPricePe
rcentage":1,"DisableSwap":false},"Version":"0.2.14-alpha"}
```

3.6 nil Pointer Security Audit

During the audit, no security issues caused by incorrect use of nil pointers were found.

3.7 Business Logic Vulnerability Audit

Bitcoin transactions use scanning blocks to obtain transaction information from the blocks, and configure and check the number of confirmations of transactions. Legal transactions will be recorded in the db, but special transactions are not verified when checking transactions. For example: CLTV, CSV these two time lock transactions need to be spent within a specified time. So attackers can set a very long time lock malicious transaction, but it can pass the transaction check, the project party needs to wait until the specified time after receiving it, In order to spend normally, there is a issue of using time locks for DoS. It is recommended to add a check on time locks to avoid affecting the flow of funds due to malicious transactions.

- tokens/btc/verifyp2shtx.go

```
func (b *Bridge) VerifyP2shTransaction(txHash, bindAddress string, allowUnstable bool) (*tokens.TxSwapInfo, error) {
        swapInfo := &tokens.TxSwapInfo{}
        swapInfo.Hash = txHash // Hash
        if !b.IsSrc {
                return swapInfo, tokens.ErrBridgeDestinationNotSupported
        }
        p2shAddress, _, err := b.GetP2shAddress(bindAddress)
        if err != nil {
```

```
			return swapInfo, fmt.Errorf("verify p2sh tx, wrong bind address %v", bindAddress)
		}
		if !allowUnstable && !b.checkStable(txHash) {
			return swapInfo, tokens.ErrTxNotStable
		}
		tx, err := b.GetTransactionByHash(txHash)
		if err != nil {
			log.Debug(b.TokenConfig.BlockChain+" Bridge::GetTransaction fail", "tx", txHash, "err", err)
			return swapInfo, tokens.ErrTxNotFound
		}
		txStatus := tx.Status
		if txStatus.BlockHeight != nil {
			swapInfo.Height = *txStatus.BlockHeight // Height
		}
		if txStatus.BlockTime != nil {
			swapInfo.Timestamp = *txStatus.BlockTime // Timestamp
		}
		value, _, rightReceiver := b.getReceivedValue(tx.Vout, p2shAddress, p2shType)
		if !rightReceiver {
			return swapInfo, tokens.ErrTxWithWrongReceiver
		}
		swapInfo.To = p2shAddress                              // To
		swapInfo.Bind = bindAddress                            // Bind
		swapInfo.Value = common.BigFromUint64(value) // Value

		swapInfo.From = getTxFrom(tx.Vin, p2shAddress) // From

		// check sender
		if swapInfo.From == swapInfo.To {
			return swapInfo, tokens.ErrTxWithWrongSender
		}

		if !tokens.CheckSwapValue(swapInfo.Value, b.IsSrc) {
			return swapInfo, tokens.ErrTxWithWrongValue
		}

		if !allowUnstable {
			log.Debug("verify p2sh swapin pass", "from", swapInfo.From, "to", swapInfo.To, "bind", swapInfo.Bind,
"value", swapInfo.Value, "txid", swapInfo.Hash, "height", swapInfo.Height, "timestamp", swapInfo.Timestamp)
		}
		return swapInfo, nil}
```

- tokens/btc/verifytx.go

```
func (b *Bridge) verifySwapinTx(txHash string, allowUnstable bool) (*tokens.TxSwapInfo, error) {
		swapInfo := &tokens.TxSwapInfo{}
		swapInfo.Hash = txHash // Hash
		if !allowUnstable && !b.checkStable(txHash) {
			return swapInfo, tokens.ErrTxNotStable
		}
		tx, err := b.GetTransactionByHash(txHash)
		if err != nil {
			log.Debug(b.TokenConfig.BlockChain+" Bridge::GetTransaction fail", "tx", txHash, "err", err)
			return swapInfo, tokens.ErrTxNotFound
		}
		txStatus := tx.Status
		if txStatus.BlockHeight != nil {
			swapInfo.Height = *txStatus.BlockHeight // Height
		}
		if txStatus.BlockTime != nil {
			swapInfo.Timestamp = *txStatus.BlockTime // Timestamp
		}
		depositAddress := b.TokenConfig.DepositAddress
		value, memoScript, rightReceiver := b.getReceivedValue(tx.Vout, depositAddress, anyType)
		if !rightReceiver {
			return swapInfo, tokens.ErrTxWithWrongReceiver
		}
		swapInfo.To = depositAddress                          // To
		swapInfo.Value = common.BigFromUint64(value) // Value

		memoStr, bindAddress, bindOk := getBindAddressFromMemoScipt(memoScript)
		if memoStr == aggregateMemo {
			return swapInfo, tokens.ErrTxIsAggregateTx
		}
		swapInfo.Bind = bindAddress // Bind


		swapInfo.From = getTxFrom(tx.Vin, depositAddress) // From


		// check sender
		if swapInfo.From == swapInfo.To {
			return swapInfo, tokens.ErrTxWithWrongSender
		}


		if !tokens.CheckSwapValue(swapInfo.Value, b.IsSrc) {
			return swapInfo, tokens.ErrTxWithWrongValue
		}


		if !bindOk {
			log.Debug("wrong memo", "memo", memoScript)
			return swapInfo, tokens.ErrTxWithWrongMemo
		} else if !tokens.DstBridge.IsValidAddress(swapInfo.Bind) {
			log.Debug("wrong bind address in memo", "bind", swapInfo.Bind)
			return swapInfo, tokens.ErrTxWithWrongMemo
		}


		if !allowUnstable {
			log.Debug("verify swapin pass", "from", swapInfo.From, "to", swapInfo.To, "bind", swapInfo.Bind, "value",
swapInfo.Value, "txid", swapInfo.Hash, "height", swapInfo.Height, "timestamp", swapInfo.Timestamp)
		}
		return swapInfo, nil}
```

3.8 DoS Vulnerability Audit

When parsing the input data of ERC20, the length of the input data is not checked, which leads to a panic issue. Attackers can construct malicious input data. When the node scans the transaction and parse it, DoS will occur. It is recommended to check the length of input data before processing input data.

- tokens/eth/verifyerc20tx.go

```
func parseErc20EncodedData(encData []byte, isTransferFrom bool) (from, to string, value *big.Int, err error) {
		if isTransferFrom {
```

```
                from = common.BytesToAddress(common.GetData(encData, 0, 32)).String()
                encData = encData[32:]
        }
        if len(encData) != 64 {
                return "", "", nil, fmt.Errorf("wrong length of encoded data")
        }
        to = common.BytesToAddress(common.GetData(encData, 0, 32)).String()
        value = common.GetBigInt(encData, 32, 32)
        return from, to, value, nil}
```

3.9 Sensitive Information Leakage Audit

During the audit, no security issues related to the leakage of sensitive information were found.

3.10 Log Processing Security Audit

During the audit, no security issues in log processing were found.

3.11 Cached Information Security Audit

The scanned data is stored in MongoDB. No security issues were found during the audit. There are enhancement points. It is recommended to add salt to the data when storing the data in db to hash the data. Before obtaining the data, you must verify that the data is consistent with the hash to pass. Verification to avoid the loss of funds due to the tampering of data in the database.

3.12 Access Control Security Audit

The operation of the admin needs to sign the transaction locally with the private key, and then send and verify it through RPC.

- cmd/swapadmin/utils.go

```
func adminCall(method string, params []string) (result interface{}, err error) {
        rawTx, err := admin.Sign(method, params)
        if err != nil {
                return "", err
        }
        err = client.RPCPost(&result, swapServer, "swap.AdminCall", rawTx)
```

```
	return result, err}
```

- admin/sign.go

```
func Sign(method string, params []string) (rawTx string, err error) {
	log.Info("admin Sign", "method", method, "params", params)
	payload, err := encodeCallArgs(method, params)
	if err != nil {
		return "", err
	}

	tx := types.NewTransaction(
		0,            // nonce
		adminToAddr,     // to address
		big.NewInt(0), // value
		0,            // gasLimit
		big.NewInt(0), // gasPrice
		payload,       // data
	)

	signedTx, err := types.SignTx(tx, adminSigner, keyWrapper.PrivateKey)
	if err != nil {
		return "", err
	}

	txdata, err := rlp.EncodeToBytes(signedTx)
	if err != nil {
		return "", err
	}

	return common.ToHex(txdata), nil}
```

- tools/loadkeystore.go

```
// LoadKeyStore load keystore from keyfile and passfile
func LoadKeyStore(keyfile, passfile string) (*keystore.Key, error) {
	keyjson, err := ioutil.ReadFile(keyfile)
	if err != nil {
```

```
                    return nil, fmt.Errorf("read keystore fail %v", err)
        }
        passdata, err := ioutil.ReadFile(passfile)
        if err != nil {
                    return nil, fmt.Errorf("read password fail %v", err)
        }
        passwd := strings.TrimSpace(string(passdata))
        key, err := keystore.DecryptKey(keyjson, passwd)
        if err != nil {
                    return nil, fmt.Errorf("decrypt key fail %v", err)
        }
        return key, nil}
```

3.13 Configuration file Security Audit

The config.toml configuration file will store the smtp password. It is recommended to put the account password in the environment variable to avoid the disclosure of the configuration file and the account password.

3.14 Uncaught Exception Handling Audit

During the audit, it was discovered that the ERC20 input data was not detected, and the panic abnormality was not captured and caused the DoS problem.
Reference: 3.8 DoS Vulnerability Audit.

3.15 Recursion and Loop Security Audit

During the audit, no security issues caused by recursive errors or loop errors were found.

3.16 Database Query Security Audit

When querying historical data, there is no check for negative numbers. When the amount of data is large, querying with a negative limit will exhaust the resources of the database service. It is recommended to check the range of limit, and it is forbidden to enter negative numbers.

- internal/swapapi/api.go

```
func processHistoryLimit(limit int) int {
        if limit == 0 {
                limit = 20
        } else if limit > 100 {
                limit = 100
        }
        return limit}
```

4 Audit Result

4.1 Critical vulnerabilities

Critical severity issues can have a major impact on the security of project, and it is highly recommended to fix critical severity vulnerability.

The audit has shown no critical severity vulnerability.

4.2 High-risk vulnerabilities

High severity issues can affect the normal operation of project, and it is highly recommended to fix high severity vulnerability.

During the audit, two high-risk vulnerabilities were discovered:

1. Business Logic Vulnerability
2. DoS Vulnerability

These two high-risk vulnerabilities will cause loss of funds, it is strongly recommended to fix high-risk vulnerabilities.

Reference: 3.7 Business Logic Vulnerability Audit, 3.8 DoS Vulnerability Audit.

4.3 Medium-risk vulnerabilities

Medium severity issues can affect the operation of a project, and it is recommended to fix medium severity vulnerability.

The audit has shown no medium severity vulnerability.

4.4 Low-risk vulnerabilities

Low severity issues can affect project operation in future versions of code. We recommend the project party to evaluate and consider whether these problems need to be fixed.

During the audit, a low-risk vulnerability was discovered:

1. When querying historical data, it is not judged that the value of limit is negative. As data accumulates, node resources will be exhausted due to this issue.

Reference: 3.16 Database Query Security Audit。

4.5 Enhancement Suggestions

The enhancement suggestion is the optimization proposal for the project, and the project party self-assess and considers whether these problems need to be optimized.

During the audit, 4 enhancement points found:

1. The security of communication with RPC must use https for communication.
2. The response header set in the code is "application/json", but in fact it returns "text/plain". Please check this bug.
3. It is recommended to hash the data with salt when storing the data in db. Before obtaining the data, you must verify that the data is consistent with the hash to pass the verification, avoiding the loss of funds due to the data in the database being tampered with.
4. The config.toml configuration file will store the smtp password. It is recommended to put the account password in the environment variable to avoid the disclosure of the configuration file and the account password.

Reference： 3.1 Communication Security Audit, 3.5 RPC Interface Security Audit, 3.11 Cached Information Security Audit, 3.13 Configuration file Security Audit.

4.6 Audit conclusion

Summary: 7 issues were found in this audit, including 2 high-risk vulnerabilities, 1 low-risk vulnerabilities, and four enhancement suggestions. After communicating with the project team, both high-risk and low-risk vulnerabilities have been fixed, and the enhancement points will be enhanced in subsequent versions of the code.

5 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 the project, and is not responsible for it. 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 this report (referred to as "the provided information"). If the provided information is missing, tampered, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom.