SLOWMIST

Blockchain Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.02.14, the SlowMist security team received the AB(Formerly Newton) team's security audit application for AB BlockChain, 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.
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	Cryptographic Security Audit	Some Risks
2	Account and Transaction Security Audit	Some Risks
3	RPC Security Audit	Passed
4	P2P Security Audit	Passed
5	Consensus Security Audit	Some Risks

3 Project Overview

3.1 Project Introduction

AB BlockChain is based on Official golang implementation of the Ethereum protocol.

3.2 Coverage

Target Code and Revision:

https://github.com/newtonproject/newchain

Initial audit version: v1.9.18-newton

Final audit version: 391f1ae549292b67acbbaa04ecc68f93b8cc817a

Cryptographic Security Audit

(1) Insufficient entropy of private key random numbers audit

Generate encrypted accounts using the “crypto/rand” library to ensure that private keys cannot be predicted or cracked.

- accounts/keystore/keystore.go

```
import (
   "crypto/ecdsa"
   crand "crypto/rand"
//...
// NewAccount generates a new key and stores it into the key directory,
// encrypting it with the passphrase.
func (ks *KeyStore) NewAccount(passphrase string) (accounts.Account, error) {
   _, account, err := storeNewKey(ks.storage, crand.Reader, passphrase)
   if err != nil {
       return accounts.Account{}, err
   }
   // Add the account to the cache immediately rather
   // than waiting for file system notifications to pick it up.
   ks.cache.add(account)
   ks.refreshWallets()
   return account, nil
}
```

(2) Precision loss in private key seed conversion

Use ecdsa from the Go standard library to generate private keys, which are defined to be of type `big.Int`, with enough length to preserve random seeds without causing the strength of the private key to degrade.

```
// PrivateKey represents an ECDSA private key.
type PrivateKey struct {
   PublicKey
   D *big.Int
}
```

- accounts/keystore/key.go

```
func newKey(rand io.Reader) (*Key, error) {
   privateKeyECDSA, err := ecdsa.GenerateKey(crypto.S256(), rand)
   if err != nil {
       return nil, err
   }
   return newKeyFromECDSA(privateKeyECDSA), nil
}
```

(3) Theoretical reliability assessment of symmetric encryption algorithms

Scrypt key management:

- mobile/accounts.go
- accounts/keystore/passphrase.go

ECIES encryption:

- crypto/ecies/params.go

(4) Theoretical reliability assessment of hash algorithms

```
- SHA-1/224/256/384/512: for message digests
- Keccak256: Address generation
- HMAC: Message Authentication Code
```

(5) Theoretical reliability assessment of signature algorithms

```
secp256r1
```

(6) Supply chain security of symmetric crypto algorithm reference libraries

The main task is to check the supply chain security of the reference libraries of the symmetric encryption algorithms used.

AB BlockChain's cryptographic libraries use the Ethereum library, which is mature, reliable and widely used.

Reference

https://github.com/ethereum/go-ethereum/tree/master/crypto

(7) Keystore encryption strength detection

The main purpose is to check the encryption strength of the Keystore.

The password strength of passphrase is not detected and there is a risk of weak passwords.

- accounts/keystore/keystore.go

```
// NewAccount generates a new key and stores it into the key directory,
// encrypting it with the passphrase.
func (ks *KeyStore) NewAccount(passphrase string) (accounts.Account, error) {
   _, account, err := storeNewKey(ks.storage, crand.Reader, passphrase)
   if err != nil {
       return accounts.Account{}, err
   }
   // Add the account to the cache immediately rather
   // than waiting for file system notifications to pick it up.
   ks.cache.add(account)
   ks.refreshWallets()
   return account, nil
}
```

(8) Hash algorithm length extension attack

The main purpose is to check the length extension attack of the hash algorithm.

Using Keccak256 algorithm to calculate transaction hash, there is no hash length extension attack problem.

- core/types/block.go

```
func rlpHash(x interface{}) (h common.Hash) {
   sha := hasherPool.Get().(crypto.KeccakState)
   defer hasherPool.Put(sha)
   sha.Reset()
   rlp.Encode(sha, x)
   sha.Read(h[:])
   return h
}
```

(9) Merkle-tree Malleability attack

The main purpose is to check the modifiability of the Merkle Tree in the blockchain system.

In some merkle-tree algorithms, if there are an odd number of nodes, the last node is automatically copied to spell an even number, and an attacker may try to double-flush by including two identical transactions at the end of the block.

The Merkle-tree scheme used by AB BlockChain is consistent with that of Ethereum and there are no security issues.

(10) secp256k1 k-value randomness security

AB BlockChain has not used the secp256k1 signature algorithm.

(11) secp256k1 r-value reuse private key extraction attack

AB BlockChain has not used the secp256k1 signature algorithm.

(12) ECC signature malleability attack

The main task is to examine the malleability of signatures in Elliptic Curve Digital Signature (ECC signature), as well as the related security risks and potential attacks.

There is an extensibility problem in secp256r1. By changing the s value in the signature to s - N or N - s, the signature result can be changed. This led to the attack on the mt.gox in the early version of Bitcoin. However, this only affects the security of off-chain applications and does not affect the security of AB BlockChain itself.

(13) Ed25519 private key extraction attack

The main purpose is to check the risks and impacts of the private key extraction attack in the Ed25519 elliptic curve digital signature algorithm.

AB BlockChain has not used the ed25519 signature algorithm.

(14) Schnorr private key extraction attack

The main task is to check the risks and impacts of the private key extraction attack existing in the Schnorr signature algorithm.

AB BlockChain has not used the ed25519 signature algorithm.

(15) ECC twist attack

The main task is to check the risks and impacts of the elliptic curve twist attack existing in Elliptic Curve Cryptography (ECC).

AB BlockChain does not use the elliptic curve cryptography library with security issues.

Account and Transaction Security Audit

(1) Native characteristic false recharge

The main objective is to examine the risk and impact of possible native feature false top-up vulnerabilities in blockchain systems.

Native feature false recharge refers to attackers exploiting the characteristics or vulnerabilities of the blockchain system to forge recharge records on the blockchain. This kind of attack may cause losses to users, exchanges or blockchain platforms. It should be noted that AB BlockChain has a notable feature: failed transactions may still be submitted to the chain, and the status of each transaction needs to be confirmed.

(2) Contract call-based false recharge

The main check is the risks and impacts of potential false recharge vulnerabilities based on contract calls that may exist in blockchain smart contracts.

AB BlockChain adopts an EVM-compatible smart contract architecture, which brings powerful functions while also inheriting certain complex characteristics. One of the particularly notable ones is the behavior of cross-contract calls. In a complex transaction, even if an internal cross-contract call fails, the entire transaction may still be marked as successful. This feature may lead to potential security risks, such as vulnerabilities like 'false top-up'. Therefore, it is necessary to deeply verify the execution results of all internal calls in the transaction. Only when all related internal transactions are successfully executed can the validity of the entire transaction be truly confirmed.

(3) Native chain transaction replay attack

The main purpose is to assess the risks and impacts of possible local chain transaction replay attacks in the blockchain network.

Transaction replay attacks refer to a type of attack where the attacker resubmits previously valid transaction data to the blockchain network, deceiving network nodes and participants, causing the transaction to be executed repeatedly. This may result in unnecessary financial losses, transaction delays, or other negative impacts.

For each address in AB BlockChain, a Nonce is added as a parameter for transactions. After a successful transaction, the Nonce is incremented by one. Therefore, there is no problem of transaction replay.

(4) Heterogeneous chain transaction replay attack

The main task is to examine the risks and impacts of potential transaction replay attacks that may exist between heterogeneous chains.

Transaction replay attacks between heterogeneous chains refer to a type of attack where attackers exploit the interoperability between different blockchain networks to repeat a valid transaction that was successfully executed on one chain on another chain, in order to gain improper benefits or cause system damage. In this type of attack, the attacker copies previously valid transaction data across chains and submits it to another chain, deceiving nodes and participants on the chain, causing the transaction to be executed repeatedly. This may result in financial losses, transaction delays, or other negative impacts.

Although AB BlockChain is designed to be fully compatible with the EVM, a specific chainID is embedded in the signature data of each transaction to prevent the transaction from being directly re-executed on another chain.

(5) Transaction lock attack

The main task is to check the risks and impacts of possible transaction locking attacks in the blockchain system.

Transaction locking attack refers to the manipulation of blockchain transactions by malicious users or attackers to make certain funds or resources remain locked for a long period of time, in order to achieve the purpose of causing damage to the system, interfering with it, or obtaining undue benefits. In this type of attack, the attacker may take advantage of incompletely understood smart contract logic, blockchain protocol vulnerabilities, or transaction sequencing rules to make specific transactions unable to be confirmed or executed, resulting in funds or resources being locked for a long period of time, affecting the normal operation of the system and user experience.

Transactions in AB BlockChain do not have a locking function.

RPC Security Audit

(1) RPC remote key theft attack

The main task is to examine the risks and impacts of potential RPC remote key theft attacks that may exist in the blockchain system.

The interface follows the Ethernet JSON-RPC specification.

The API provides interfaces such as remote signatures, and there is a risk of remote coin theft. An attacker may remotely initiate signature operations to steal node assets during the interval when the node's account is unlocked.

- ethapi/ethapi.go

```
func (s *PublicTransactionPoolAPI) Sign(addr common.Address, data hexutil.Bytes)
(hexutil.Bytes, error) {
   // Look up the wallet containing the requested signer
   account := accounts.Account{Address: addr}


   wallet, err := s.b.AccountManager().Find(account)
   if err != nil {
       return nil, err
   }
   // Sign the requested hash with the wallet
   signature, err := wallet.SignText(account, data)
   if err == nil {
       signature[64] += 27 // Transform V from 0/1 to 27/28 according to the yellow
paper
   }
   return signature, err
}
```

But in the configuration file of deploy script, the default open modules of open api not contain the insecurity functions.

- https://github.com/newtonproject/newchain-deploy/blob/master/mainnet/conf/guard.toml#L7

```
MethodWhitelist = [
    "net_version",
    "net_peerCount",
    "net_listening",
    "eth_protocolVersion",
    "eth_syncing",
    "eth_mining",
    "eth_hashrate",
    "eth_gasPrice",
    "eth_blockNumber",
    "eth_getBalance",
    "eth_getStorageAt",
    "eth_getTransactionCount",
    "eth_getBlockTransactionCountByHash",
    "eth_getBlockTransactionCountByNumber",
    "eth_getUncleCountByBlockHash",
    "eth_getUncleCountByBlockNumber",
    "eth_getCode",
    "eth_sendRawTransaction",
    "eth_call",
    "eth_estimateGas",
    "eth_getBlockByHash",
    "eth_getBlockByNumber",
    "eth_getTransactionByHash",
    "eth_getTransactionByBlockHashAndIndex",
    "eth_getTransactionByBlockNumberAndIndex",
    "eth_getTransactionReceipt",
    "eth_getUncleByBlockHashAndIndex",
    "eth_getUncleByBlockNumberAndIndex",
    "eth_getCompilers",
    "eth_compileLLL",
    "eth_compileSolidity",
    "eth_compileSerpent",
    "eth_newFilter",
    "eth_newBlockFilter",
    "eth_newPendingTransactionFilter",
    "eth_uninstallFilter",
    "eth_getFilterChanges",
    "eth_getFilterLogs",
    "eth_getLogs",
    "eth_chainId",
    "eth_getProof",
    "eth_getRawTransactionByHash",
    "txpool_status",
    "rpc_modules",
]
```

(2) RPC port identifiability

Port 8545/8546 is used by default, and can also be specified with a command line parameter, or through a configuration file.

- node/defaults.go

```
const (
    DefaultHTTPHost    = "localhost" // Default host interface for the HTTP RPC server
    DefaultHTTPPort    = 8545        // Default TCP port for the HTTP RPC server
    DefaultWSHost      = "localhost" // Default host interface for the websocket RPC
server
    DefaultWSPort      = 8546        // Default TCP port for the websocket RPC server
    DefaultGraphQLHost = "localhost" // Default host interface for the GraphQL server
    DefaultGraphQLPort = 8547        // Default TCP port for the GraphQL server
)
```

- https://github.com/newtonproject/newchain-deploy/blob/master/mainnet/conf/node.toml#L53

```
HTTPHost = "127.0.0.1"
HTTPPort = 8808
```

(3) RPC open cross-domain vulnerability to local phishing attacks

The main check is to assess the cross-domain vulnerabilities of RPC interfaces in the blockchain system to determine whether the system is vulnerable to local phishing attacks.

The hacker tricks the victim into opening a malicious webpage, connects to the cryptocurrency wallet RPC port through a cross-domain request, and then steals crypto assets.

Test with a public RPC:

```
curl -s -v -D- 'https://global.rpc.mainnet.newtonproject.org' \
  -H 'content-type: application/json' \
  --data-raw '{
 "id":101,
 "jsonrpc":"2.0",
 "method":"eth_getTransactionByHash",
 "params":["0xb766964fd3f22d615545019de693e381b964d2bcacf8cae4f35fe5bcc4aa403f"]
}'
```

`Access-Control-Allow-Origin: *` was not found in the return header, there is no cross-domain request issue.

(4) JsonRPC malformed packet denial-of-service attack

The main check is the security of the JsonRPC interface in the blockchain system to determine whether the system is vulnerable to Denial of Service (DoS) attacks caused by maliciously constructed abnormal JSON data packets.

Constructing malformed data for testing node RPCs:

```
    data = '{"' + '}' * 0x101000 + '":' + '{"x":' * 0x10000 + '"}'
    print(post(posturl, data))
```

And

```
g_req += 'Content-Type: application/json\r\n'
g_req += 'Content-length: %d\r\n\r\n' % 0xffffffffffffffff
g_req += '{"method":"'
```

Returned results normally and did not cause the node to crash.

```
http.cli
```

(5) RPC database injection

The main check is whether there is a database injection problem.

There are no database queries using SQL statements and there are no RPC injection issues.

(6) RPC communication encryption

The main purpose is to check whether the RPC (Remote Procedure Call) communication in the blockchain system has appropriate encryption protection.

RPC does not use HTTPS encryption by default, but the node operator can encrypt the communication by adding a proxy such as Nginx in the front.

Related Code

- rpc/*

P2P Security Audit

(1) P2P communication encryption

The main check is whether the P2P (peer-to-peer) communication between nodes in the blockchain network uses an appropriate encryption mechanism to protect the privacy and security of the communication content.

The P2P encryption implementation is divided into the following main parts:

The handshake process uses the RLPx encrypted transmission protocol

- p2p/server.go

```
   // Run the RLPx handshake.
   remotePubkey, err := c.doEncHandshake(srv.PrivateKey, dialPubkey)
```

- p2p/rlpx/rlpx.go

```
// secrets represents the connection secrets
// which are negotiated during the encryption handshake.
type secrets struct {
   Remote                *ecies.PublicKey
   AES, MAC              []byte
   EgressMAC, IngressMAC hash.Hash
   Token                 []byte
}
```

The cryptographic handshake process uses ECDH for key exchange and ECIES for message encryption.

- p2p/rlpx.go

```
// doEncHandshake runs the protocol handshake using authenticated
// messages. the protocol handshake is the first authenticated message
// and also verifies whether the encryption handshake 'worked' and the
// remote side actually provided the right public key.
func (t *rlpx) doEncHandshake(prv *ecdsa.PrivateKey, dial *ecdsa.PublicKey)
(*ecdsa.PublicKey, error) {
   var (
       sec secrets
       err error
   )
   if dial == nil {
       sec, err = receiverEncHandshake(t.fd, prv)
   } else {
       sec, err = initiatorEncHandshake(t.fd, prv, dial)
   }
   if err != nil {
       return nil, err
   }
   t.wmu.Lock()
   t.rw = newRLPXFrameRW(t.fd, sec)
   t.wmu.Unlock()
   return sec.Remote.ExportECDSA(), nil
}
```

Symmetric encryption using AES-CTR mode and message integrity verification using MAC.

- p2p/rlpx.go

```
func newRLPXFrameRW(conn io.ReadWriter, s secrets) *rlpxFrameRW {
   macc, err := aes.NewCipher(s.MAC)
   if err != nil {
       panic("invalid MAC secret: " + err.Error())
   }
   encc, err := aes.NewCipher(s.AES)
   if err != nil {
       panic("invalid AES secret: " + err.Error())
   }
   // we use an all-zeroes IV for AES because the key used
   // for encryption is ephemeral.
   iv := make([]byte, encc.BlockSize())
   return &rlpxFrameRW{
       conn:       conn,
       enc:        cipher.NewCTR(encc, iv),
       dec:        cipher.NewCTR(encc, iv),
       macCipher:  macc,
       egressMAC:  s.EgressMAC,
       ingressMAC: s.IngressMAC,
    }
}
```

v4 uses ECDSA-based signatures to verify identity

v5 uses AES/GCM mode for message encryption

- p2p/discover/v5_encoding.go

```
func (c *wireCodec) encodeEncrypted(toID enode.ID, toAddr string, packet packetV5,
writeKey []byte, challenge *whoareyouV5) (enc []byte, authTag []byte, err error) {
    // ...
    enc, err = encryptGCM(headbuf, writeKey, nonce, body.Bytes(), tag[:])
    return enc, nonce, err
}
```

(2) Insufficient number of core nodes

The main check is whether the number of core nodes in the blockchain network is sufficient to ensure the security and stability of the network.

Utilizing the PoA consensus algorithm, a small number of nodes hold the rights to produce blocks, with fewer nodes, there is a higher risk of centralization.

(3) Excessive concentration of core node physical locations

The main check is whether the physical location distribution of the core nodes in the blockchain network is too concentrated.

(4) P2P node maximum connection limit

The main check is the maximum connection limit of a blockchain node to other nodes.

The `MaxPeers` parameter is defined in config to limit the maximum number of connections to peer nodes, preventing the system from experiencing performance degradation or even denial of service due to too many connections.

- p2p/server.go

```
// Config holds Server options.
type Config struct {
    //...
    // MaxPeers is the maximum number of peers that can be
    // connected. It must be greater than zero.
    MaxPeers int
//...
```

- node/defaults.go

```
    P2P: p2p.Config{
        ListenAddr: ":30303",
        MaxPeers:   50,
        NAT:        nat.Any(),
    },
```

(5) P2P node independent IP connection limit

The main check is the limit on the number of independent connections of the blockchain node to the same IP address.

Use `inboundHistory` to record connected nodes, while checking whether the same IP has an existing connection when a new connection is made, avoiding the malicious construction of a large number of nodes by one IP node to occupy the connection pool of the target node.

- p2p/server.go

```
func (srv *Server) checkInboundConn(fd net.Conn, remoteIP net.IP) error {
    if remoteIP == nil {
        return nil
    }
    // Reject connections that do not match NetRestrict.
    if srv.NetRestrict != nil && !srv.NetRestrict.Contains(remoteIP) {
        return fmt.Errorf("not whitelisted in NetRestrict")
    }
    // Reject Internet peers that try too often.
    now := srv.clock.Now()
    srv.inboundHistory.expire(now, nil)
    if !netutil.IsLAN(remoteIP) && srv.inboundHistory.contains(remoteIP.String()) {
        return fmt.Errorf("too many attempts")
    }
    srv.inboundHistory.add(remoteIP.String(), now.Add(inboundThrottleTime))
    return nil
}
```

(6) P2P inbound/outbound connection limit

The main check is the limit on the number of incoming and outgoing connections of the blockchain node.

DialRatio controls the ratio of inbound to dialed connections.

- p2p/server.go

```
type Config struct {
   //...
   DialRatio int `toml:",omitempty"`
   //...
}
//...
func (srv *Server) postHandshakeChecks(peers map[enode.ID]*Peer, inboundCount int, c
*conn) error {
   switch {
   case !c.is(trustedConn) && len(peers) >= srv.MaxPeers:
       return DiscTooManyPeers
   case !c.is(trustedConn) && c.is(inboundConn) && inboundCount >=
srv.maxInboundConns():
       return DiscTooManyPeers
   //...
   }
}
```

(7) P2P Alien attack

Both Discv4 and Discv5 are enabled on the node discovery protocol, but the protocol does not support verifying that peer nodes are on the same chain during handshaking, which may cause nodes on the current chain and the address pools of similar chains, such as Ethereum, to pollute each other, resulting in degraded network performance or even congestion.

- p2p/server.go

```
func (srv *Server) setupDiscovery() error {
        //...
       ntab, err := discover.ListenV4(conn, srv.localnode, cfg)
       //...
       if sconn != nil {
           srv.DiscV5, err = discover.ListenV5(sconn, srv.localnode, cfg)
       } else {
           srv.DiscV5, err = discover.ListenV5(conn, srv.localnode, cfg)
       }
       //...
   }
   return nil
}
```

- p2p/discover/v4wire/v4wire.go

```
   Ping struct {
       Version    uint
       From, To   Endpoint
       Expiration uint64
       // Ignore additional fields (for forward compatibility).
```

```
    Rest []rlp.RawValue `rlp:"tail"`
}
```

- p2p/discv5/udp.go

```
ping struct {
    Version    uint
    From, To   rpcEndpoint
    Expiration uint64


    // v5
    Topics []Topic


    // Ignore additional fields (for forward compatibility).
    Rest []rlp.RawValue `rlp:"tail"`
}


// pong is the reply to ping.
pong struct {
    // This field should mirror the UDP envelope address
    // of the ping packet, which provides a way to discover the
    // the external address (after NAT).
    To rpcEndpoint


    ReplyTok   []byte // This contains the hash of the ping packet.
    Expiration uint64 // Absolute timestamp at which the packet becomes invalid.


    // v5
    TopicHash    common.Hash
    TicketSerial uint32
    WaitPeriods  []uint32


    // Ignore additional fields (for forward compatibility).
    Rest []rlp.RawValue `rlp:"tail"`
}


// findnode is a query for nodes close to the given target.
findnode struct {
    Target     NodeID // doesn't need to be an actual public key
    Expiration uint64
    // Ignore additional fields (for forward compatibility).
    Rest []rlp.RawValue `rlp:"tail"`
}
```

Reference:

https://mp.weixin.qq.com/s/UmricgYGUakAlZTb0ihqdw

(8) P2P port identifiability

The main purpose is to check whether the ports used for P2P (peer-to-peer) communication between nodes in the blockchain network are easy to be identified and tracked.

Port 38311 is used by default, and can also be specified with the command line parameter --port, or through a configuration file.

- https://github.com/newtonproject/newchain-deploy/blob/master/mainnet/conf/node.toml#L67

```
ListenAddr = ":38311"
```

Consensus Security Audit

(1) Excessive administrator privileges

The main task is to check whether the system has administrator permissions or special beneficiary accounts to ensure the rationality, minimization and decentralization of permission control, thereby guaranteeing that there is no fraudulent behavior in the system.

There is not special administrator permission in AB BlockChain.

(2) Transaction fees not dynamically adjusted

The main check is whether the transaction fees in the blockchain system are dynamically adjusted according to the network conditions and demands.

In the AB BlockChain, transaction fees are dynamically adjusted based on the network congestion situation to ensure the efficiency and fairness of transaction processing.

(3) Miner grinding attack

The main purpose is to assess the potential risk of grinding attacks by miners in the blockchain system.

The PoA consensus algorithm will not have the grinding block attack problem.

(4) Final confirmation conditions

Block time is 3 seconds, so block finalized time is 75 hours

- params/network_params.go

```
    // FullImmutabilityThreshold is the number of blocks after which a chain segment is
    // considered immutable (i.e. soft finality). It is used by the downloader as a
    // hard limit against deep ancestors, by the blockchain against deep reorgs, by
    // the freezer as the cutoff threshold and by clique as the snapshot trust limit.
    FullImmutabilityThreshold = 90000
```

(5) Double-signing penalty

Using the PoA consensus algorithm, the security of the network is guaranteed by the credibility of authoritative nodes. If the authoritative node can double sign to construct another fork chain, then the fork will be inevitable.

(6) Block refusal penalty

Using the PoA consensus algorithm, there is no node punishment mechanism.

(7) Block time offset attack

The time deviation of the current block must not exceed `maxTimeFutureBlocks`, which is set to 30s on the mainnet.

- core/blockchain.go

```
func (bc *BlockChain) addFutureBlock(block *types.Block) error {
    max := uint64(time.Now().Unix() + maxTimeFutureBlocks)
    if block.Time() > max {
        return fmt.Errorf("future block timestamp %v > allowed %v", block.Time(), max)
    }
    bc.futureBlocks.Add(block.Hash(), block)
    return nil
}
```

(8) Consensus algorithm potential risk assessment

- consensus/clique/clique.go

  This file contains the implementation of the Clique PoA consensus algorithm. Key Points of the Clique PoA Algorithm:

```
Signers: Authorized accounts that can sign and create new blocks.
Extra Data: The header extra data field is split into the vanity section and the seal
section.
Nonces: Special nonce values are used to vote on adding or removing signers.
Difficulty: The difficulty field is used to encode whether the signer is in-turn or
out-of-turn.
Epochs: Periodic checkpoints reset the pending votes and signer set.
Signing and Sealing: Blocks are signed by authorized signers and are verified by other
nodes.
```

Utilizing the PoA consensus algorithm, a small number of nodes hold the rights to produce blocks, with fewer nodes, there is a higher risk of centralization.

3.3 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	False top-up risk of exchanges	Account and Transaction Security Audit	Information	Acknowledged
N2	Secp256r1 signature risks	Cryptographic Security Audit	Low	Acknowledged
N3	Errors unhandled	SAST	Low	Acknowledged
N4	Use of net/http serve function that has no support for setting timeouts	SAST	Low	Acknowledged
N5	Potential Slowloris Attack because ReadHeaderTimeout is not configured in the http.Server	SAST	Low	Acknowledged
N6	Implicit memory aliasing in for loop.	SAST	Medium	Fixed
N7	High-risk vulnerabilities found in Ethereum over the years have not been fixed	Others	High	Fixed
N8	comparePublicKey check is not strict	SAST	Low	Fixed
N9	The centralization risk of the PoA consensus algorithm	Consensus Security Audit	Information	Acknowledged

4 Findings

4.1 Vulnerability Summary

[N1] [Information] False top-up risk of exchanges

Category: Account and Transaction Security Audit

Content

In a complex transaction, even if an internal cross-contract call fails, the entire transaction may still be marked as successful. This feature may lead to potential security risks like 'false top-up'

Solution

Exchanges need to pay attention to the risk of fake top-up and strictly check deposit transactions.

Status

Acknowledged

[N2] [Low] Secp256r1 signature risks

Category: Cryptographic Security Audit

Content

- Signature malleability

  ECDSA signatures have a double validity issue, where a legitimate signature may have a variant (for example, s value takes n-s), which may lead to replay or confusion risks. In the code, the s value is adjusted to reduce this risk.

- Parameter dispute

  The parameters for secp256r1 were set by NIST, and there has been controversy over whether these parameters have potential backdoors (although there is currently no conclusive evidence).

Solution

Status

Acknowledged

[N3] [Low] Errors unhandled

Category: SAST

Content

The following code does not process the returned error information during the calling process, which may cause the program to not terminate in time when an error occurs, resulting in a logical error.

```
- whisper/whisperv6/whisper.go (line 782)
- whisper/whisperv6/whisper.go (line 609)
- whisper/whisperv6/api.go (line 424)
- whisper/whisperv6/api.go (line 421)
- whisper/mailserver/mailserver.go (line 98)
- trie/sync.go (line 211)
- trie/sync.go (line 194)
- trie/secure_trie.go (line 182)
- trie/secure_trie.go (line 181)
- trie/proof.go (line 469)
- trie/proof.go (line 427)
- trie/proof.go (line 86)
- trie/hasher.go (line 187)
- trie/hasher.go (line 186)
- trie/database.go (line 801)
- trie/database.go (line 748)
- trie/committer.go (line 246)
- trie/committer.go (line 245)
- trie/committer.go (line 234)
- trie/committer.go (line 229)
- tests/state_test_util.go (line 298)
- tests/state_test_util.go (line 196)
- tests/fuzzers/trie/trie-fuzzer.go (line 95)
- tests/fuzzers/rlp/rlp_fuzzer.go (line 55)
- tests/fuzzers/rlp/rlp_fuzzer.go (line 49)
- tests/fuzzers/rlp/rlp_fuzzer.go (line 43)
- tests/fuzzers/keystore/keystore-fuzzer.go (line 35)
- signer/rules/rules.go (line 107)
- signer/rules/rules.go (line 94-98)
- signer/rules/rules.go (line 85-93)
- signer/rules/rules.go (line 81)
- signer/rules/rules.go (line 80)
- signer/fourbyte/validation.go (line 102)
- signer/core/stdioui.go (line 41)
- signer/core/api.go (line 365)
- rpc/websocket.go (line 229)
- rpc/websocket.go (line 228)
- rpc/subscription.go (line 255)
- rpc/subscription.go (line 66)
- rpc/server.go (line 107)
- rpc/server.go (line 57)
- rpc/json.go (line 255)
- rpc/json.go (line 251)
- rpc/json.go (line 247)
- rpc/json.go (line 231)
- rpc/json.go (line 224)
- rpc/ipc_unix.go (line 47)
- rpc/ipc_unix.go (line 42)
- rpc/handler.go (line 145)
- rpc/handler.go (line 142)
- rpc/handler.go (line 128)
- rpc/handler.go (line 125)
- rpc/handler.go (line 100)
- rpc/client.go (line 623)
- p2p/simulations/test.go (line 51)
- p2p/simulations/pipes/pipes.go (line 51)
- p2p/simulations/network.go (line 250)
- p2p/simulations/network.go (line 247)
- p2p/simulations/network.go (line 244)
- p2p/simulations/network.go (line 241)
- p2p/simulations/http.go (line 698)
- p2p/simulations/http.go (line 665)
- p2p/simulations/http.go (line 115)
- p2p/simulations/adapters/types.go (line 219)
- p2p/simulations/adapters/exec.go (line 427)
- p2p/simulations/adapters/exec.go (line 403)
- p2p/simulations/adapters/exec.go (line 325)
- p2p/simulations/adapters/exec.go (line 308)
- p2p/simulations/adapters/exec.go (line 237)
- p2p/simulations/adapters/exec.go (line 172)
- p2p/server.go (line 889)
- p2p/server.go (line 876)
- p2p/server.go (line 402)
- p2p/rlpx.go (line 158)
- p2p/rlpx.go (line 121)
- p2p/rlpx.go (line 117)
- p2p/rlpx.go (line 101)
- p2p/rlpx.go (line 94)
- p2p/rlpx.go (line 87)
- p2p/peer.go (line 293)
- p2p/peer.go (line 287)
- p2p/nodestate/nodestate.go (line 558)
- p2p/nat/natupnp.go (line 65)
- p2p/nat/nat.go (line 106)
- p2p/enr/enr.go (line 250)
- p2p/enode/urlv4.go (line 176)
- p2p/enode/urlv4.go (line 175)
- p2p/enode/nodedb.go (line 467)
- p2p/enode/nodedb.go (line 429)
- p2p/enode/nodedb.go (line 409)
- p2p/enode/nodedb.go (line 288)
- p2p/enode/nodedb.go (line 120)
- p2p/enode/nodedb.go (line 113)
- p2p/enode/node.go (line 128)
- p2p/enode/node.go (line 121)
- p2p/enode/idscheme.go (line 149)
- p2p/enode/idscheme.go (line 73)
- p2p/enode/idscheme.go (line 52)
- p2p/dnsdisc/tree.go (line 234)
- p2p/discv5/udp.go (line 374)
- p2p/discv5/udp.go (line 310)
- p2p/discv5/udp.go (line 304)
- p2p/discv5/udp.go (line 289-293)
- p2p/discv5/udp.go (line 282-285)
- p2p/discv5/udp.go (line 275)
- p2p/discv5/udp.go (line 249)
- p2p/discv5/topic.go (line 149)
- p2p/discv5/table.go (line 163)
- p2p/discv5/table.go (line 154)
- p2p/discv5/table.go (line 109)
- p2p/discv5/net.go (line 1236)
- p2p/discv5/database.go (line 395)
- p2p/discv5/database.go (line 330)
- p2p/discv5/database.go (line 277)
- p2p/discv5/database.go (line 120)
- p2p/discv5/database.go (line 113)
- p2p/discover/v5_udp.go (line 832)
- p2p/discover/v5_udp.go (line 812)
- p2p/discover/v5_udp.go (line 791)
- p2p/discover/v5_udp.go (line 783)
- p2p/discover/v5_udp.go (line 727-732)
- p2p/discover/v5_udp.go (line 680)
- p2p/discover/v5_udp.go (line 675)
- p2p/discover/v5_udp.go (line 490)
- p2p/discover/v5_udp.go (line 429)
- p2p/discover/v5_udp.go (line 269)
- p2p/discover/v5_udp.go (line 187)
- p2p/discover/v5_session.go (line 63)
- p2p/discover/v5_encoding.go (line 555)
- p2p/discover/v5_encoding.go (line 394)
- p2p/discover/v5_encoding.go (line 393)
- p2p/discover/v5_encoding.go (line 392)
- p2p/discover/v5_encoding.go (line 263)
- p2p/discover/v4wire/v4wire.go (line 195)
- p2p/discover/v4_udp.go (line 770-773)
- p2p/discover/v4_udp.go (line 737)
- p2p/discover/v4_udp.go (line 731)
- p2p/discover/v4_udp.go (line 689)
- p2p/discover/v4_udp.go (line 673)
- p2p/discover/v4_udp.go (line 655-660)
- p2p/discover/v4_udp.go (line 350)
- p2p/discover/v4_udp.go (line 323-326)
- p2p/discover/v4_udp.go (line 288)
- p2p/discover/v4_udp.go (line 246)
- p2p/discover/v4_udp.go (line 168)
- p2p/discover/table.go (line 391)
- p2p/discover/table.go (line 136)
- p2p/discover/ntp.go (line 92)
- p2p/discover/lookup.go (line 160)
- p2p/discover/lookup.go (line 152)
- p2p/dial.go (line 493)
- p2p/dial.go (line 156)
- node/node.go (line 449)
- node/node.go (line 409)
- node/node.go (line 358)
- node/node.go (line 245)
- node/node.go (line 233)
- node/defaults.go (line 101)
- node/api.go (line 131)
- mobile/interface.go (line 237)
- mobile/interface.go (line 229)
- mobile/interface.go (line 221)
- mobile/interface.go (line 213)
- mobile/interface.go (line 193)
- mobile/interface.go (line 185)
- mobile/interface.go (line 177)
- mobile/interface.go (line 169)
- mobile/geth.go (line 187)
- miner/worker.go (line 970)
- miner/worker.go (line 934)
- miner/worker.go (line 624)
- miner/worker.go (line 467)
- metrics/timer.go (line 61)
- metrics/syslog.go (line 57-74)
- metrics/syslog.go (line 45-53)
- metrics/syslog.go (line 29-42)
- metrics/syslog.go (line 25)
- metrics/syslog.go (line 22)
- metrics/syslog.go (line 20)
- metrics/syslog.go (line 18)
- metrics/runtime.go (line 211)
- metrics/runtime.go (line 210)
- metrics/runtime.go (line 209)
- metrics/runtime.go (line 208)
- metrics/runtime.go (line 207)
- metrics/runtime.go (line 206)
- metrics/runtime.go (line 205)
- metrics/runtime.go (line 204)
- metrics/runtime.go (line 203)
- metrics/runtime.go (line 202)
- metrics/runtime.go (line 201)
- metrics/runtime.go (line 200)
- metrics/runtime.go (line 199)
- metrics/runtime.go (line 198)
- metrics/runtime.go (line 197)
- metrics/runtime.go (line 196)
- metrics/runtime.go (line 195)
- metrics/runtime.go (line 194)
- metrics/runtime.go (line 193)
- metrics/runtime.go (line 192)
- metrics/runtime.go (line 191)
- metrics/runtime.go (line 190)
- metrics/runtime.go (line 189)
- metrics/runtime.go (line 188)
- metrics/runtime.go (line 187)
- metrics/runtime.go (line 186)
- metrics/runtime.go (line 185)
- metrics/runtime.go (line 184)
- metrics/runtime.go (line 183)
- metrics/runtime.go (line 182)
- metrics/runtime.go (line 181)
- metrics/resetting_timer.go (line 39)
- metrics/registry.go (line 92)
- metrics/prometheus/prometheus.go (line 66)
- metrics/opentsdb.go (line 116)
- metrics/meter.go (line 97)
- metrics/meter.go (line 84)
- metrics/json.go (line 26)
- metrics/histogram.go (line 44)
- metrics/graphite.go (line 110)
- metrics/gauge_float64.go (line 55)
- metrics/gauge_float64.go (line 37)
- metrics/gauge.go (line 55)
- metrics/gauge.go (line 37)
- metrics/debug.go (line 69)
- metrics/debug.go (line 68)
- metrics/debug.go (line 66)
- metrics/debug.go (line 65)
- metrics/debug.go (line 64)
- metrics/counter.go (line 69)
- metrics/counter.go (line 56)
- log/logger.go (line 134-146)
- log/handler.go (line 209)
- light/txpool.go (line 531)
- light/txpool.go (line 521)
- light/txpool.go (line 518)
- light/txpool.go (line 445)
- light/txpool.go (line 213)
- light/postprocess.go (line 486)
- light/postprocess.go (line 478)
- light/postprocess.go (line 465)
- light/postprocess.go (line 318)
- light/postprocess.go (line 249)
- light/postprocess.go (line 241)
- light/postprocess.go (line 228)
- light/postprocess.go (line 128)
- light/odr.go (line 104)
- light/nodeset.go (line 133)
- light/nodeset.go (line 123)
- light/lightchain.go (line 197)
- light/lightchain.go (line 114)
- 1)
- les/txrelay.go (line 147)
- les/test_helper.go (line 579)
- les/test_helper.go (line 578)
- les/test_helper.go (line 577)
- les/test_helper.go (line 576)
- les/test_helper.go (line 575)
- les/test_helper.go (line 497)
- les/test_helper.go (line 496)
- les/test_helper.go (line 494)
- les/test_helper.go (line 358)
- les/test_helper.go (line 349)
- les/test_helper.go (line 313)
- les/test_helper.go (line 154)
- les/test_helper.go (line 148)
- les/test_helper.go (line 143)
- les/test_helper.go (line 138)
- les/test_helper.go (line 133)
- les/test_helper.go (line 129)
- les/test_helper.go (line 125)
- les/test_helper.go (line 118)
- les/serverpool.go (line 460)
- les/serverpool.go (line 459)
- les/serverpool.go (line 453)
- les/serverpool.go (line 392)
- les/serverpool.go (line 390)
- les/serverpool.go (line 387)
- les/serverpool.go (line 284)
- les/serverpool.go (line 275)
- les/server_handler.go (line 955)
- les/server_handler.go (line 946)
- les/server_handler.go (line 723)
- les/server_handler.go (line 155)
- les/server.go (line 207)
- les/retrieve.go (line 433)
- les/retrieve.go (line 336)
- les/pruner.go (line 88)
- les/peer.go (line 909)
- les/peer.go (line 898)
- les/peer.go (line 880)
- les/peer.go (line 788)
- les/peer.go (line 776)
- les/peer.go (line 613)
- les/peer.go (line 612)
- les/odr.go (line 122)
- les/fetcher.go (line 558)
- les/costtracker.go (line 268)
- les/clientpool.go (line 863)
- les/clientpool.go (line 826)
- les/clientpool.go (line 820)
- les/clientpool.go (line 761)
- les/clientpool.go (line 755)
- les/clientpool.go (line 725)
- les/clientpool.go (line 583)
- les/clientpool.go (line 252)
- les/client_handler.go (line 407)
- les/client_handler.go (line 402)
- les/client_handler.go (line 383)
- les/client_handler.go (line 359)
- les/client_handler.go (line 333)
- les/client_handler.go (line 120)
- les/client.go (line 311)
- les/client.go (line 308)
- les/client.go (line 304)
- les/client.go (line 303)
- les/client.go (line 159)
- les/benchmark.go (line 351)
- les/benchmark.go (line 350)
- les/benchmark.go (line 342)
- les/benchmark.go (line 341)
- les/benchmark.go (line 329)
- les/benchmark.go (line 284)
- les/benchmark.go (line 206)
- les/benchmark.go (line 182)
- les/benchmark.go (line 119)
- les/api_backend.go (line 57)
- les/api.go (line 89-92)
- les/api.go (line 68-71)
- internal/jsre/jsre.go (line 191)
- internal/jsre/jsre.go (line 160-165)
- internal/jsre/jsre.go (line 154-159)
- internal/jsre/jsre.go (line 81)
- internal/jsre/jsre.go (line 80)
- internal/debug/trace.go (line 59)
- internal/debug/trace.go (line 41)
- internal/debug/flags.go (line 224)
- internal/debug/flags.go (line 223)
- internal/debug/flags.go (line 149)
- internal/debug/flags.go (line 148)
- internal/debug/api.go (line 195)
- internal/debug/api.go (line 140)
- internal/debug/api.go (line 127)
- internal/debug/api.go (line 109)
- internal/debug/api.go (line 93)
- internal/cmdtest/test_cmd.go (line 223)
- internal/cmdtest/test_cmd.go (line 219)
- internal/cmdtest/test_cmd.go (line 126)
- internal/cmdtest/test_cmd.go (line 85)
- internal/build/util.go (line 152)
- internal/build/download.go (line 101)
- internal/build/archive.go (line 227)
- internal/build/archive.go (line 87)
- internal/build/archive.go (line 84)
- graphql/service.go (line 110)
- graphql/graphiql.go (line 56)
- ethstats/ethstats.go (line 285)
- ethstats/ethstats.go (line 258)
- ethstats/ethstats.go (line 246)
- ethstats/ethstats.go (line 237)
- ethdb/dbtest/testsuite.go (line 275)
- ethdb/dbtest/testsuite.go (line 274)
- ethdb/dbtest/testsuite.go (line 273)
- ethdb/dbtest/testsuite.go (line 272)
- ethdb/dbtest/testsuite.go (line 271)
- ethdb/dbtest/testsuite.go (line 127)
- ethclient/signer.go (line 39)
- eth/tracers/tracer.go (line 406-422)
- eth/tracers/tracer.go (line 401-405)
- eth/tracers/tracer.go (line 379-400)
- eth/tracers/tracer.go (line 365-378)
- eth/tracers/tracer.go (line 354-364)
- eth/tracers/tracer.go (line 343-353)
- eth/tracers/tracer.go (line 339-342)
- eth/handler.go (line 801)
- eth/handler.go (line 746)
- eth/handler.go (line 718)
- eth/handler.go (line 693)
- eth/handler.go (line 242)
- eth/handler.go (line 241)
- eth/filters/api.go (line 259)
- eth/filters/api.go (line 222)
- eth/filters/api.go (line 156)
- eth/fetcher/tx_fetcher.go (line 803)
- eth/fetcher/block_fetcher.go (line 418)
- eth/downloader/statesync.go (line 423)
- eth/downloader/peer.go (line 204)
- eth/downloader/peer.go (line 180)
- eth/downloader/fakepeer.go (line 159)
- eth/downloader/fakepeer.go (line 146)
- eth/downloader/fakepeer.go (line 135)
- eth/downloader/fakepeer.go (line 118)
- eth/downloader/fakepeer.go (line 91)
- eth/downloader/downloader.go (line 1771)
- eth/downloader/downloader.go (line 1666)
- eth/downloader/downloader.go (line 1640)
- eth/downloader/downloader.go (line 601)
- eth/downloader/downloader.go (line 431)
- eth/downloader/downloader.go (line 428)
- eth/downloader/downloader.go (line 424)
- eth/downloader/downloader.go (line 370)
- eth/downloader/api.go (line 108)
- eth/bloombits.go (line 121)
- eth/backend.go (line 577)
- eth/backend.go (line 576)
- eth/backend.go (line 571)
- eth/backend.go (line 354)
- eth/backend.go (line 220)
- eth/backend.go (line 200)
- eth/api_tracer.go (line 604)
- eth/api_tracer.go (line 601)
- eth/api_tracer.go (line 343)
- eth/api_backend.go (line 58)
- crypto/crypto.go (line 81)
- crypto/crypto.go (line 79)
- crypto/crypto.go (line 70)
- crypto/crypto.go (line 68)
- crypto/blake2b/blake2x.go (line 163)
- crypto/blake2b/blake2x.go (line 147)
- core/vm/interpreter.go (line 282)
- core/vm/interpreter.go (line 200)
- core/vm/interpreter.go (line 198)
- core/vm/instructions.go (line 244)
- core/vm/instructions.go (line 243)
- core/vm/evm.go (line 478)
- core/vm/evm.go (line 443)
- core/vm/evm.go (line 227)
- core/vm/evm.go (line 225)
- core/vm/evm.go (line 215)
- core/vm/evm.go (line 214)
- core/vm/contracts.go (line 800)
- core/vm/contracts.go (line 669)
- core/types/transaction.go (line 216)
- core/types/derive_sha.go (line 37)
- core/types/block.go (line 355)
- core/types/block.go (line 146)
```

- core/types/block.go (line 145)
- core/tx_pool.go (line 1419)
- core/tx_pool.go (line 1407)
- core/tx_pool.go (line 906)
- core/tx_pool.go (line 398)
- core/tx_journal.go (line 155)
- core/tx_journal.go (line 149)
- core/tx_cacher.go (line 65)
- core/state/snapshot/wipe.go (line 106)
- core/state/snapshot/snapshot.go (line 444)
- core/state/snapshot/generate.go (line 249)
- core/state/snapshot/generate.go (line 225)
- core/state/snapshot/generate.go (line 187)
- core/state/snapshot/conversion.go (line 272)
- core/state/dump.go (line 109-111)
- core/state/dump.go (line 104)
- core/rawdb/freezer_table.go (line 642)
- core/rawdb/freezer_table.go (line 534)
- core/rawdb/freezer_table.go (line 528)
- core/rawdb/freezer_table.go (line 510)
- core/rawdb/freezer_table.go (line 490)
- core/rawdb/freezer_table.go (line 445)
- core/rawdb/freezer_table.go (line 443)
- core/rawdb/freezer_table.go (line 434)
- core/rawdb/freezer_table.go (line 352)
- core/rawdb/freezer_table.go (line 270)
- core/rawdb/freezer_table.go (line 268)
- core/rawdb/freezer_table.go (line 239)
- core/rawdb/freezer_table.go (line 238)
- core/rawdb/freezer_table.go (line 233)
- core/rawdb/freezer_table.go (line 232)
- core/rawdb/freezer_table.go (line 215)
- core/rawdb/freezer_table.go (line 189)
- core/rawdb/freezer_table.go (line 183)
- core/rawdb/freezer.go (line 122)
- core/rawdb/freezer.go (line 120)
- core/rawdb/freezer.go (line 113)
- core/rawdb/freezer.go (line 111)
- core/rawdb/database.go (line 218)
- core/rawdb/accessors_indexes.go (line 170)
- core/genesis.go (line 288)
- core/genesis.go (line 287)
- core/chain_indexer.go (line 521)
- core/chain_indexer.go (line 512)
- core/chain_indexer.go (line 483)
- core/blockchain.go (line 1479)
- core/blockchain.go (line 1460)
- ethdb.IdealBatchSize)
- core/blockchain.go (line 950)
- core/blockchain.go (line 316)
- core/blockchain.go (line 273)
- core/asm/compiler.go (line 180)
- console/prompt/prompter.go (line 155)
- console/prompt/prompter.go (line 127)

```
- console/prompt/prompter.go (line 109)
- console/prompt/prompter.go (line 97)
- console/console.go (line 248)
- console/console.go (line 247)
- console/console.go (line 246)
- console/console.go (line 245)
- console/console.go (line 244)
- console/console.go (line 243)
- console/console.go (line 242)
- console/console.go (line 241)
- console/console.go (line 224)
- console/console.go (line 223)
- console/console.go (line 222)
- console/console.go (line 180)
- console/console.go (line 179)
- console/console.go (line 159)
- console/console.go (line 158)
- console/bridge.go (line 469)
- console/bridge.go (line 456)
- console/bridge.go (line 431)
- console/bridge.go (line 420)
- console/bridge.go (line 413)
- console/bridge.go (line 412)
- console/bridge.go (line 405)
- console/bridge.go (line 401)
- consensus/ethash/sealer.go (line 386)
- consensus/ethash/ethash.go (line 374)
- consensus/ethash/ethash.go (line 373)
- consensus/ethash/ethash.go (line 358)
- consensus/ethash/ethash.go (line 278)
- consensus/ethash/ethash.go (line 277)
- consensus/ethash/ethash.go (line 269)
- consensus/ethash/ethash.go (line 135)
- consensus/ethash/ethash.go (line 88)
- consensus/ethash/ethash.go (line 87)
- consensus/ethash/ethash.go (line 81)
- consensus/ethash/ethash.go (line 80)
- consensus/ethash/ethash.go (line 75)
- consensus/ethash/consensus.go (line 593-607)
- consensus/ethash/algorithm.go (line 115)
- consensus/ethash/algorithm.go (line 114)
- common/hexutil/json.go (line 142)
- common/hexutil/json.go (line 121)
- common/compiler/solidity.go (line 156)
- common/compiler/solidity.go (line 155)
- cmd/wnode/main.go (line 695)
- cmd/wnode/main.go (line 436)
- cmd/utils/flags.go (line 1924)
- cmd/utils/flags.go (line 1718)
- cmd/utils/flags.go (line 1715)
- cmd/utils/flags.go (line 1650)
- cmd/utils/flags.go (line 97)
- cmd/utils/customflags.go (line 73)
- cmd/rlpdump/main.go (line 109)
- cmd/puppeth/wizard_network.go (line 50)
- cmd/puppeth/wizard.go (line 64)
- cmd/puppeth/ssh.go (line 252)
- cmd/puppeth/ssh.go (line 168)
- cmd/puppeth/puppeth.go (line 54)
- cmd/puppeth/module_wallet.go (line 103-111)
- cmd/puppeth/module_wallet.go (line 90-99)
- cmd/puppeth/module_node.go (line 117-131)
- cmd/puppeth/module_node.go (line 100-113)
- cmd/puppeth/module_nginx.go (line 71-74)
- cmd/puppeth/module_nginx.go (line 67)
- cmd/puppeth/module_faucet.go (line 110-123)
- cmd/puppeth/module_faucet.go (line 94-106)
- cmd/puppeth/module_explorer.go (line 100-110)
- cmd/puppeth/module_explorer.go (line 87-92)
- cmd/puppeth/module_ethstats.go (line 87-93)
- cmd/puppeth/module_ethstats.go (line 80-83)
- cmd/puppeth/module_dashboard.go (line 610-636)
- cmd/puppeth/module_dashboard.go (line 582-590)
- cmd/puppeth/module_dashboard.go (line 576-578)
- cmd/puppeth/module.go (line 150)
- cmd/geth/retesteth.go (line 917)
- cmd/geth/retesteth.go (line 481)
- cmd/geth/retesteth.go (line 284)
- cmd/geth/retesteth.go (line 281)
- cmd/geth/main.go (line 452)
- cmd/geth/main.go (line 429)
- cmd/geth/main.go (line 327)
- cmd/geth/main.go (line 315)
- cmd/geth/main.go (line 309)
- cmd/geth/main.go (line 266)
- cmd/geth/consolecmd.go (line 229)
- cmd/geth/config.go (line 201)
- cmd/geth/config.go (line 200)
- cmd/geth/chaincmd.go (line 547)
- cmd/geth/chaincmd.go (line 540-551)
- cmd/geth/chaincmd.go (line 255)
- cmd/faucet/faucet.go (line 671)
- cmd/faucet/faucet.go (line 657)
- cmd/faucet/faucet.go (line 635)
- cmd/faucet/faucet.go (line 630)
- cmd/faucet/faucet.go (line 427)
- cmd/faucet/faucet.go (line 306)
- cmd/faucet/faucet.go (line 271)
- cmd/faucet/faucet.go (line 252)
- cmd/evm/runner.go (line 286)
- cmd/evm/runner.go (line 271)
- cmd/evm/internal/t8ntool/transition.go (line 274)
- cmd/evm/internal/t8ntool/transition.go (line 267)
- cmd/evm/internal/t8ntool/transition.go (line 129)
- cmd/evm/internal/t8ntool/transition.go (line 100)
- cmd/evm/internal/t8ntool/transition.go (line 95)
- cmd/evm/internal/t8ntool/execution.go (line 252)
- cmd/devp2p/nodeset.go (line 66)
- cmd/devp2p/internal/v4test/framework.go (line 76)
- cmd/devp2p/internal/v4test/framework.go (line 75)
- cmd/devp2p/internal/v4test/discv4tests.go (line 442)
- cmd/devp2p/internal/v4test/discv4tests.go (line 419)
- cmd/devp2p/internal/v4test/discv4tests.go (line 334)
- cmd/devp2p/internal/v4test/discv4tests.go (line 273)
- cmd/devp2p/internal/v4test/discv4tests.go (line 256)
- cmd/devp2p/dnscmd.go (line 379)
- cmd/clef/main.go (line 1147)
- cmd/clef/main.go (line 993)
- cmd/clef/main.go (line 940)
- cmd/clef/main.go (line 712)
- cmd/clef/main.go (line 699)
- cmd/clef/main.go (line 624)
- cmd/checkpoint-admin/main.go (line 98)
- cmd/checkpoint-admin/exec.go (line 254)
- cmd/bootnode/main.go (line 57)
- accounts/usbwallet/wallet.go (line 273)
- accounts/usbwallet/wallet.go (line 211)
- accounts/scwallet/securechannel.go (line 268)
- accounts/scwallet/hub.go (line 251)
- accounts/scwallet/hub.go (line 241)
- accounts/scwallet/hub.go (line 228)
- accounts/keystore/watch.go (line 104)
- accounts/keystore/key.go (line 203)
- accounts/keystore/key.go (line 200)
- accounts/keystore/key.go (line 199)
- accounts/keystore/account_cache.go (line 215)
```

Solution

Check the return value of a function call.

Status

Acknowledged

[N4] [Low] Use of net/http serve function that has no support for setting timeouts

Category: SAST

Content

- p2p/simulations/examples/ping-pong.go (line 82)

```
81:      })
82:      if err := http.ListenAndServe(":8888", simulations.NewServer(network)); err
!= nil {
83:              log.Crit("error starting simulation server", "err", err)
```

- metrics/exp/exp.go (line 64)

```
63:     go func() {
64:             if err := http.ListenAndServe(address, m); err != nil {
65:                     log.Error("Failure in running metrics server", "err", err)
```

- internal/debug/flags.go (line 214)

```
213:    go func() {
214:            if err := http.ListenAndServe(address, nil); err != nil {
215:                    log.Error("Failure in running pprof server", "err", err)
```

- cmd/faucet/faucet.go (line 300)

```
299:    http.HandleFunc("/api", f.apiHandler)
300:    return http.ListenAndServe(fmt.Sprintf(":%d", port), nil)
301: }
```

Solution

Support for setting timeouts.

Status

Acknowledged

[N5] [Low] Potential Slowloris Attack because ReadHeaderTimeout is not configured in the http.Server

Category: SAST

Content

- node/endpoints.go (line 61)

```
60:     }
61:     wsSrv := &http.Server{Handler: handler}
62:     go wsSrv.Serve(listener)
```

- p2p/simulations/http.go (line 444)

```
443:            for _, conn := range snap.Conns {
444:                    event := NewEvent(&conn)
445:                    if err := writeEvent(event); err != nil {
```

- p2p/simulations/http.go (line 437)

```
436:            for _, node := range snap.Nodes {
437:                    event := NewEvent(&node.Node)
```

```
438:                     if err := writeEvent(event); err != nil {
```

- eth/filters/api.go (line 259)

```
258:                             for _, log := range logs {
259:                                     notifier.Notify(rpcSub.ID, &log)
260:                             }
```

- core/rawdb/freezer.go (line 399)

```
398:     for _, table := range f.tables {
399:             items := atomic.LoadUint64(&table.items)
400:             if min > items {
```

- cmd/geth/accountcmd.go (line 244)

```
243:             if err := ks.Unlock(a, auth); err == nil {
244:                     match = &a
245:                     break
```

- cmd/geth/accountcmd.go (line 199)

```
198:             for _, account := range wallet.Accounts() {
199:                     fmt.Printf("Account #%d: {%x} %s\n", index, account.Address,
&account.URL)
200:                     index++
```

Solution

Status

Acknowledged

[N6] [Medium] Implicit memory aliasing in for loop.

Category: SAST

Content

- p2p/simulations/http.go (line 444)

```
443:             for _, conn := range snap.Conns {
444:                     event := NewEvent(&conn)
445:                     if err := writeEvent(event); err != nil {
```

- p2p/simulations/http.go (line 437)

```
436:            for _, node := range snap.Nodes {
437:                    event := NewEvent(&node.Node)
438:                    if err := writeEvent(event); err != nil {
```

- eth/filters/api.go (line 259)

```
258:                            for _, log := range logs {
259:                                    notifier.Notify(rpcSub.ID, &log)
260:                            }
```

- core/rawdb/freezer.go (line 399)

```
398:    for _, table := range f.tables {
399:            items := atomic.LoadUint64(&table.items)
400:            if min > items {
```

- cmd/geth/accountcmd.go (line 244)

```
243:            if err := ks.Unlock(a, auth); err == nil {
244:                    match = &a
245:                    break
```

- cmd/geth/accountcmd.go (line 199)

```
198:            for _, account := range wallet.Accounts() {
199:                    fmt.Printf("Account #%d: {%x} %s\n", index, account.Address,
&account.URL)
200:                    index++
```

Solution

By creating a copy of the variable, you ensure that each iteration receives an independent copy of the current variable, thus avoiding implicit memory aliasing problems.

Status

Fixed

[N7] [High] High-risk vulnerabilities found in Ethereum over the years have not been fixed

Category: Others

Content

- Mining nodes will generate erroneous PoW on epochs > `385`.

- A denial-of-service issue can be used to crash Geth nodes during block processing, due to an underlying bug in Go (CVE-2020-28362) versions < `1.15.5`, or `<1.14.12`
- A consensus flaw in Geth, related to `datacopy` precompile
- A denial-of-service issue can be used to crash Geth nodes during block processing
- A DoS vulnerability can make a LES server crash.
- A consensus-vulnerability in Geth could cause a chain split, where vulnerable versions refuse to accept the canonical chain.
- The client is not ready for the 'London' technical upgrade, and will deviate from the canonical chain when the London upgrade occurs (at block '12965000' around August 4, 2021.
- A consensus-flaw in the Geth EVM could cause a node to deviate from the canonical chain.
- A vulnerable node is susceptible to crash when processing a maliciously crafted message from a peer, via the snap/1 protocol. The crash can be triggered by sending a malicious snap/1 GetTrieNodes package.
- A vulnerable node can crash via p2p messages sent from an attacker node, if running with non-default log options.
- A vulnerable node can be made to consume unbounded amounts of memory when handling specially crafted p2p messages sent from an attacker node.
- A vulnerable node can be made to consume very large amounts of memory when handling specially crafted p2p messages sent from an attacker node.

Reference: https://geth.ethereum.org/docs/vulnerabilities/vulnerabilities.json

Solution

Upgrade to the latest version of go-ethereum or fix vulnerabilities one by one.

Status

Fixed

[N8] [Low] comparePublicKey check is not strict

Category: SAST

Content

`comparePublicKey` did not compare `elliptic.Curve`, attackers may bypass checks by constructing public keys.

- crypto/signature_r1.go

```
func comparePublicKey(key1, key2 *ecdsa.PublicKey) bool {
   // TODO: compare curve
   x := key1.X.Cmp(key2.X)
   y := key2.Y.Cmp(key2.Y)
   if x == 0 && y == 0 {
       return true
   } else {
       return false
   }
}
```

Solution

Compare the curve.

Status

Fixed

[N9] [Information] The centralization risk of the PoA consensus algorithm

Category: Consensus Security Audit

Content

1.Centralized control: Only a few pre-selected nodes are responsible for block generation and verification, which may lead to excessive centralized control over network decision-making and operations.

2.Trust risk: Network security relies on the trustworthiness of authoritative nodes. If some nodes have security vulnerabilities or are attacked, the entire network may face serious security threats.

3.Single point of failure: If an authoritative node fails or is attacked, it may have a significant impact on block generation, leading to service interruptions or data tampering risks.

4.Insider threats: The centralized management authority of nodes may lead to internal collusion, abuse of power, and other issues, making the network no longer fully decentralized and transparent.

Solution

Introducing hybrid consensus solutions that incorporate other consensus mechanisms (such as PoS, PBFT, etc.), further dispersing risks.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002502210001	SlowMist Security Team	2025.02.14 - 2025.02.21	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 high risk, 1 medium risk, 5 low risk 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