



PlatON 安全审计报告



目录

1 前言(Executive Summary).....	3
2 项目背景(Context).....	4
2.1 项目简介.....	4
2.2 审计范围.....	4
3 代码分析(Code Overview).....	5
3.1 基础架构.....	5
3.2 静态代码检查.....	6
3.2.1 错误未处理.....	6
3.3 P2P 安全.....	6
3.3.1 节点连接数审计.....	6
3.3.2 节点性能审计.....	7
3.3.3 通信加密审计.....	8
3.3.4 “异形攻击” 审计.....	8
3.4 RPC 安全.....	8
3.4.1 远程调用权限审计.....	8
3.4.2 畸形数据请求审计.....	9
3.4.3 通信加密审计.....	9
3.4.4 同源策略审计.....	9
3.5 加密签名安全.....	9
3.5.1 随机数生成算法审计.....	9

3.5.2 密钥存储审计.....	9
3.5.3 密码学组件调用审计.....	9
3.5.4 哈希强度审计.....	10
3.5.5 交易延展性攻击审计.....	10
3.5.6 加解密模糊测试.....	10
3.6 账户与交易模型安全.....	10
3.6.1 交易权限校验审计.....	10
3.6.2 交易重放审计.....	12
3.6.3 “假充值” 审计.....	12
3.7 虚拟机安全审计.....	14
3.7.1 EVM 虚拟机安全审计.....	14
3.7.2 WASM 虚拟安全审计.....	15
3.8 共识安全.....	15
3.8.1 激励层安全审计.....	15
3.8.2 区块校验审计.....	15
3.8.3 默克尔树审计.....	16
4 审计结果(Result).....	16
4.1 增强建议.....	16
4.2 交易所安全小结.....	16
4.3 结论.....	17
5 声明(Statement).....	17

1 前言(Executive Summary)

慢雾安全团队于 2020-06-10 日，收到 PlatON 团队对 PlatON 安全审计申请，根据双方约定和项目特点制定审计方案，并最终出具安全审计报告。

慢雾安全团队采用“黑灰为主，白盒为辅”的策略，以最贴近真实攻击的方式，对项目方进行完整的安全测试。

慢雾科技区块链系统测试方法：

黑盒测试	站在外部从攻击者角度进行安全测试。
灰盒测试	通过脚本工具对代码模块进行安全测试，观察内部运行状态，挖掘弱点。
白盒测试	基于开源、未开源代码，对节点、SDK 等程序进行漏洞挖掘。

慢雾科技区块链风险等级：

严重漏洞	严重漏洞会对区块链的安全造成重大影响，强烈建议修复严重漏洞。
高危漏洞	高危漏洞会影响区块链的正常运行，强烈建议修复高危漏洞。
中危漏洞	中危漏洞会影响区块链的运行，建议修复中危漏洞。
低危漏洞	低危漏洞可能在特定场景中会影响区块链的操作，建议项目方自行评估和考虑这些问题是否需要修复。
弱点	理论上存在安全隐患，但工程上极难复现。
增强建议	编码或架构存在更好的实践方法。

2 项目背景(Context)

2.1 项目简介

PlatON 是基于密码学算法构建的可扩展的 Trustless 计算网络，可解决区块链的可扩展性和隐私性问题。

项目官网：<https://www.platon.network/>

项目源码：<https://github.com/PlatONnetwork/PlatON-Go>

审计版本：v0.13.0

2.2 审计范围

本次安全审计的主要类型包括：

序号	审计大类	审计子类	审计结果
1	代码静态检查	内置函数安全	通过
		标准库安全审计	通过
		第三方库安全审计	通过
		注入审计	通过
		序列化算法审计	通过
		内存泄露审计	通过
		算术运算审计	通过
		资源消耗审计	通过
		异常处理审计	通过
2	P2P 安全	节点连接数审计	通过
		节点性能审计	通过
		通信加密审计	通过
		“异形攻击”审计	通过
3	RPC 安全	远程调用权限审计	通过

		畸形数据请求审计	通过
		通信加密审计	通过
		同源策略审计	通过
4	加密签名安全	随机数生成算法审计	通过
		密钥存储审计	通过
		密码学组件调用审计	通过
		哈希强度审计	通过
		交易延展性攻击审计	通过
		加解密模糊测试	通过
5	账户与交易模型安全	权限校验审计	通过
		交易重放审计	通过
		“假充值”审计	通过
6	虚拟机安全审计	EVM 虚拟机安全	通过
		WASM 虚拟机安全	通过
7	共识账本安全	激励层安全审计	通过
		区块校验审计	通过
		默克尔树审计	通过

(其他未知安全漏洞不包含在本次审计责任范围)

3 代码分析(Code Overview)

3.1 基础架构

基于开源的 go-ethereum 开发，采用 CBFT+PPoS 共识算法。

3.2 静态代码检查

3.2.1 错误未处理

错误未处理可能造成对象解引用或执行错误逻辑造成节点崩溃，代码扫描发现 876 项错误未处理，经分析，未发现可利用的漏洞。

3.3 P2P 安全

3.3.1 节点连接数审计

限制了最大连接数，限制了进站出站连接数，可有效防范日蚀攻击、女巫攻击。

- p2p/server.go

```
const (  
    defaultDialTimeout = 15 * time.Second  
  
    // Connectivity defaults.  
    maxActiveDialTasks = 16  
    defaultMaxPendingPeers = 50  
    defaultDialRatio = 3  
    maxActiveNonconsensusPeers = 5  
  
    // Maximum time allowed for reading a complete message.  
    // This is effectively the amount of time a connection can be idle.  
    frameReadTimeout = 30 * time.Second  
  
    // Maximum amount of time allowed for writing a complete message.  
    frameWriteTimeout = 20 * time.Second  
)  
...  
func (srv *Server) maxInboundConns() int {  
    return srv.MaxPeers - srv.maxDialedConns()  
}
```

```
func (srv *Server) maxDialedConns() int {  
    if srv.NoDiscovery || srv.NoDial {  
        return 0  
    }  
    r := srv.DialRatio  
    if r == 0 {  
        r = defaultDialRatio  
    }  
    return srv.MaxPeers / r  
}
```

3.3.2 节点性能审计

未发现慢函数。

限制区块接收速率，避免恶意节点大量推送数据导致节点性能不足，造成拒绝服务。

- eth/fetcher/fetcher.go

```
// enqueue schedules a new future import operation, if the block to be imported  
// has not yet been seen.  
func (f *Fetcher) enqueue(peer string, block *types.Block) {  
    hash := block.Hash()  
  
    // Ensure the peer isn't DOSing us  
    count := f.queues[peer] + 1  
    if count > blockLimit {  
        log.Debug("Discarded propagated block, exceeded allowance", "peer", peer, "number", block.Number(),  
"hash", hash, "limit", blockLimit)  
        propBroadcastDOSMeter.Mark(1)  
        f.forgetHash(hash)  
        return  
    }  
}
```


3.3.3 通信加密审计

TCP 连接使用 RLPx 加密。

3.3.4 “异形攻击” 审计

不同链的节点在握手过程中会判断 chainID，不会互相连接，不会导致地址池污染。

- p2p/discover/udp.go

```
func ListenUDP(c conn, cfg Config) (*Table, error) {
    tab, _, err := newUDP(c, cfg)
    if err != nil {
        return nil, err
    }

    if cfg.ChainID != nil {
        bytes_ChainId, _ := rlp.EncodeToBytes(cfg.ChainID)
        log.Info("UDP listener up", "chainId", cfg.ChainID, "bytes_ChainId", bytes_ChainId)
        cRest = []rlp.RawValue{bytes_ChainId, bytes_ChainId}
    }

    log.Info("UDP listener up", "self", tab.self)
    return tab, nil
}
```

3.4 RPC 安全

3.4.1 远程调用权限审计

RPC 有敏感权限，严禁开放在公网环境，谨防“黑色情人节”漏洞。

参考文档：<https://mp.weixin.qq.com/s/Kk2IsoQ1679Gda56Ec-zJg>

3.4.2 畸形数据请求审计

测试发送超大请求、超大层级 JSON、异常数据包未出现崩溃。

3.4.3 通信加密审计

非加密通信会给网络参与者带来隐私，安全和完整性的风险。

3.4.4 同源策略审计

钱包 RPC 默认不开放。

节点 RPC 默认不开启跨域。

3.5 加密签名安全

3.5.1 随机数生成算法审计

私钥种子的生成基于 crypto/rand 标准库，熵值安全。

- crypto/crypto.go

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

3.5.2 密钥存储审计

钱包未进行密码强度检测，可弱口令存储私钥。

3.5.3 密码学组件调用审计

未发现错误调用。

3.5.4 哈希强度审计

未发现 md5、sha1 等弱哈希函数用于加密。

3.5.5 交易延展性攻击审计

未发现交易延展性漏洞。

3.5.6 加解密模糊测试

基于广泛使用的 Ethereum 密钥体系，无需单独测试。

3.6 账户与交易模型安全

3.6.1 交易权限校验审计

对交易结构中的各个字段均严格校验类型和值。

- core/tx_validate.go

```
// validateSemantics checks if the transactions 'makes sense', and generate warnings for a couple of
// typical scenarios
func (v *Validator) validate(msgs *ValidationMessages, txargs *SendTxArgs, methodSelector *string)
error {
    // Prevent accidental erroneous usage of both 'input' and 'data'
    if txargs.Data != nil && txargs.Input != nil && !bytes.Equal(*txargs.Data, *txargs.Input) {
        // This is a showstopper
        return errors.New(`Ambiguous request: both "data" and "input" are set and are not identical`)
    }
    var (
        data []byte
    )
    // Place data on 'data', and nil 'input'
    if txargs.Input != nil {
        txargs.Data = txargs.Input
        txargs.Input = nil
    }
}
```

```
}  
  
if txargs.Data != nil {  
    data = *txargs.Data  
}  
  
if txargs.To == nil {  
    //Contract creation should contain sufficient data to deploy a contract  
    // A typical error is omitting sender due to some quirk in the javascript call  
    // e.g. https://github.com/ethereum/go-ethereum/issues/16106  
    if len(data) == 0 {  
        if txargs.Value.ToInt().Cmp(big.NewInt(0)) > 0 {  
            // Sending ether into black hole  
            return errors.New("Tx will create contract with value but empty code!")  
        }  
        // No value submitted at least  
        msgsg.crit("Tx will create contract with empty code!")  
    } else if len(data) < 40 { //Arbitrary limit  
        msgsg.warn(fmt.Sprintf("Tx will will create contract, but payload is suspiciously small (%db)", len(data)))  
    }  
    // methodSelector should be nil for contract creation  
    if methodSelector != nil {  
        msgsg.warn("Tx will create contract, but method selector supplied; indicating intent to call a method.")  
    }  
  
} else {  
    if !txargs.To.ValidChecksum() {  
        msgsg.warn("Invalid checksum on to-address")  
    }  
    // Normal transaction  
    if bytes.Equal(txargs.To.Address().Bytes(), common.Address{}.Bytes()) {  
        // Sending to 0  
        msgsg.crit("Tx destination is the zero address!")  
    }  
    // Validate calldata  
    v.validateCallData(msgsg, data, methodSelector)
```

```
}  
  
    return nil  
  
}
```

3.6.2 交易重放审计

参与签名的字段包含 Nonce，同一链上的交易无法重放；

参与签名的字段包含 chainID，避免了同类链或者主网与测试网之间重放攻击。

- core/types/transaction_signing.go

```
// Hash returns the hash to be signed by the sender.  
// It does not uniquely identify the transaction.  
func (s EIP155Signer) Hash(tx *Transaction) common.Hash {  
    return rlpHash([]interface{}{  
        tx.data.AccountNonce,  
        tx.data.Price,  
        tx.data.GasLimit,  
        tx.data.Recipient,  
        tx.data.Amount,  
        tx.data.Payload,  
        s.chainId, uint(0), uint(0),  
    })  
}
```

3.6.3 “假充值” 审计

尝试从创世区块的钱包地址 lax1gmhwcglky3zes3cksx8g36hpzquumg34tg5w 中向测试地址 Lax1tawnme3fhxq4ckqqksmthq82mlxp8nc4fkev4a 转账：

首先 `platon attach http://localhost:6789` 打开 console 在控制台中先解锁 input 账户：
`web3.currentProvider.unlockAccount("lax1gmhwcglky3zes3cksx8g36hpzquumg34tg5w","admin")`

然后构造 transaction object：

```
{  
  from: "lax1gmhwcglky3zes3cksx8g36hpzquumg34tg5w",  
  to: "lax1tawnme3fhxq4ckqqksmthq82mlxp8nc4fkev4a",  
}
```

```
value: "0x38d7ea4c68000"
}
```

调用函数转账: `web3.platon.sendTransaction(tx,console)`

成功后返回交易哈希:

```
"0x940f0b419328a85a0c16d24be8fc9e52d930b112381f2be0509d882c62e15778"
```

\$ curl -X POST --data

```
'{"jsonrpc":"2.0","method":"platon_getTransactionByHash","params":["0x940f0b419328a85a0c16d24be8fc9e52d930b112381f2be0509d882c62e15778"],"id":1}'
-H "accept: application/json" -H "Content-Type: application/json" http://192.168.50.120:6789
```

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "blockHash": "0x0af6fcb504813866944a37f740d92ccb00fdf3833b5623dd338b59415a3c732",
    "blockNumber": "0x3712",
    "from": "lax1gmhwcg1ky3xzes3cksx8g36hpzquumg34tg5w",
    "gas": "0x15f90",
    "gasPrice": "0x3b9aca00",
    "hash": "0x940f0b419328a85a0c16d24be8fc9e52d930b112381f2be0509d882c62e15778",
    "input": "0x",
    "nonce": "0x2",
    "to": "lax1tawnme3fhxq4ckqqksmthq82mlxp8nc4fkev4a",
    "transactionIndex": "0x0",
    "value": "0x8ac7230489e80000",
    "v": "0xed",
    "r": "0xc5c06d29c17232c3a61d92df1fa5ddd7aeb7bd03d78c2eb9c61859ed0e313dff",
    "s": "0x6dbbc89223220780df80cdd71cf26330f62d3ae67146cfebb420550f1873849"
  }
}
```

构造超过余额的转账, 并发送, 返回错误信息

```
web3.platon.personal.signTransaction({from: "lax1tawnme3fhxq4ckqqksmthq82mlxp8nc4fkev4a",to:
"lax1gmhwcg1ky3xzes3cksx8g36hpzquumg34tg5w",value: "0x38d7ea4c680000000000",gas: 210000,gasPrice:
"0x3b9aca00",nonce: 40},'123456')
truffle(development)>
```

```
web3.platon.sendSignedTransaction("0xf86f28843b9aca00830334509446eeec23f6244c2cc238b40c74475708800e73688a038d7ea4c680000000008081eda0a01c38dd0b2d88e8ae527b49e24dc415bdbc6e7d671fa267ecadeaab255511d2a04b38a315cedd22c2b9df52216e395bc7258f5c34326a1c9871c6589ad8c082e5")
```

Thrown:

Error: Returned error: insufficient funds for gas * price + value

无效的转账不会存在区块中，交易所等在校验用户充值时需要校验交易中的`from` to` `value`等字段，并查询交易所在区块是否可逆。

3.7 虚拟机安全审计

3.7.1 EVM 虚拟机安全审计

按官方提供的文档进行合约部署测试，未发现虚拟机异常情况。

特殊构造合约内联交易可造成交易所假充值，我们编写合约进行测试：

部署两个合约 Contract A 、Contract B，通过 A 调用 B，_to 是 B 合约，_payload 是 B 合约 sendEth_re 方法，_arg1 为交易所地址，_value 为转账金额。

Contract A 核心函数：

```
function single_sendEth(address _to, uint256 _value, string memory _payload, address _arg1) public  
onlyOwner {  
    _to.call.value(_value * 1 finney)(abi.encodeWithSignature(_payload,_arg1));  
}
```

Contract B 核心函数：

```
function sendEth_re(address payable _to) payable public {  
    _to.send(msg.value);  
    if (isrevert){  
        revert();  
    }  
}
```

由于 B 合约通过 send 向交易所地址转账，随后调用 revert()使交易失败了，转账回滚，但由于存在内联调用记录，交易所入账时如果没有做好状态判断将会导致“假充值”。

3.7.2 WASM 虚拟安全审计

基于成熟开源方案 (<https://github.com/go-interpreter/wagon>) 开发, 在 EOS 区块链系统上 wasm 虚拟机曾导致“交易排挤攻击”, 我们在官方示例合约中加入攻击代码进行测试:

```
CONST uint8_t get_message_size(){
while(true){ //SlowMist// payload
platon_block_number();
}
return info.self().size();
}

CONST std::string get_message_body(const uint8_t index){
while(true){} //SlowMist// payload
return info.self()[index].body;
}
```

由于测试环境所限, 我们咨询了开发团队, 得知这种情况在 Gas 消耗完了会结束运行, 攻击成本较高。

3.8 共识安全

3.8.1 激励层安全审计

在测试网 SlowMist 节点为期几个月的实际测试中, 节点激励机制准确可靠, 符合预期设计。

3.8.2 区块校验审计

已对区块进行签名校验。

- consensus/cbft/validator/validator.go

```
// VerifyHeader verify block's header.
func (vp *ValidatorPool) VerifyHeader(header *types.Header) error {
_, err := crypto.Ecrecover(header.SealHash().Bytes(), header.Signature())
if err != nil {
return err
}
```



```
}  
  
// todo: need confirmed.  
return vp.agency.VerifyHeader(header, nil)  
}
```

3.8.3 默克尔树审计

基于以太坊的默克尔树证明，未发现安全问题。

4 审计结果(Result)

4.1 增强建议

- 建议钱包进行密码强度检测，避免使用弱口令。
- RPC 非加密通信会给网络参与者带来隐私，安全和完整性的风险，建议增强。

4.2 交易所安全小结

- 严禁开放 RPC 端口到外网，避免将资金托管在节点上。
- 充值入账需要检测所有相关字段，防范“假充值攻击”。
- 提现失败时，需要等待交易过期后再重新发起，或者使用相同的 Nonce 重新发起交易，防范恶意重复提现。
- 禁止提币到合约地址，防范来自合约的恶意攻击。
- 针对合约地址充值时，需要判断内联交易中是否有 revert 的交易，如果存在 revert 的交易，则拒绝入账。

4.3 结论

审计结果：通过

审计编号：BCA002006180001

审计日期：2020 年 06 月 18 日

审计团队：慢雾安全团队

综合结论：经反馈修正后，所有发现问题均已修复，综合评估 PlatON 已无上述风险。

5 声明(Statement)

慢雾仅就本报告出具前已经发生或存在的事实出具本报告，并就此承担相应责任。对于出具以后发生或存在的事实，慢雾无法判断该项目安全状况，亦不对此承担责任。本报告所作的安全审计分析及其他内容，仅基于信息提供者截至本报告出具时向慢雾提供的文件和资料（简称“已提供资料”）。慢雾假设：已提供资料不存在缺失、被篡改、删减或隐瞒的情形。如已提供资料信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符的，慢雾对由此而导致的损失和不利影响不承担任何责任。慢雾仅对该项目的安全情况进行约定内的安全审计并出具了本报告，慢雾不对该项目背景及其他情况进行负责。



官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号

