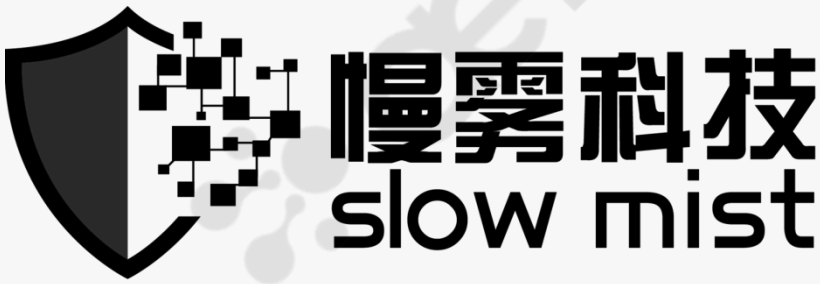


aelf 安全审计报告

目录

1 审计概要(Executive Summary)

慢雾安全团队于 2020-10-23 日，根据双方约定和项目特点制定审计方案，开始对 aelf 安全审计，并最终出具安全审计报告。

aelf 是专为新经济打造的区块链开放网络。aelf 一主链+多侧链的结构可支持开发者基于一条侧链独立部署或运行分布式应用，实现资源的有效隔离；aelf 通过采用并行运算以及 AEDPos 共识机制实现在性能上的极大突破，可承载高频交易需求；aelf 基于主链索引和验证机制的跨链技术实现了主链与侧链、以及侧链之间的信息交互，方便价值的高效传递。

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	基于源代码，对节点、SDK 等程序进行漏洞挖掘。

在黑盒测试和灰盒测试中，我们采用 fuzz 测试、脚本测试等方法，通过投喂随机数据或者构造特定结构的数据等方法，测试接口的稳健性或者组件的稳定性，并用于挖掘一些边界条件下系统的异常表现如 bug 或性能异常等。在白盒测试中，我们通过 code review 等方法，结合安全团队积累的关于已知区块链安全漏洞的相关经验对代码的对象定义、逻辑实现等作出分析，确保代码中对于关键逻辑关键组件的实现无已知漏洞；同时针对新场景、新技术进入漏洞挖掘模式，发现可能存在的 0day 错误。

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 审计约定(Engagement Goals & Scope)

本次安全审计的主要类型包括：

审计大类	审计子类
P2P 安全	节点连接数审计
	通信加密审计
	“异形攻击”审计
RPC 安全	远程调用权限审计
	畸形数据请求审计
	通信加密审计
	同源策略审计
加密签名安全	随机数生成算法审计
	密钥存储审计
	密码学组件调用审计
	哈希强度审计

	长度扩展攻击审计
	交易延展性攻击审计
账户与交易模型安全	交易校验审计
	交易重放审计
	“假充值”审计

3 审计覆盖(Coverage)

3.1 目标代码(Target Code and Revision)

源码仓库	https://github.com/aelfProject/aelf
版本	release/1.0.0-rc1
类型	基于 DPoS 共识协议的区块链
平台	C#

4 风险分析(Risk Analysis)

4.1 P2P 安全

4.1.1 节点连接数审计

- aelf/src/aelf.OS.Core/Network/NetworkConstants.cs

```
public const int DefaultMaxPeers = 25;
public const int DefaultMaxPeersPerIpAddress = 0;
```

aelf/src/aelf.OS.Core/Network/NetworkOptions.cs

```
/// <summary>
/// The maximum number of peers accepted by this node (0 for no limit).
/// </summary>
public int MaxPeers { get; set; } = NetworkConstants.DefaultMaxPeers;
/// <summary>
/// The maximum number of connection from a given host (0 for no limit).
/// </summary>
public int MaxPeersPerIpAddress { get; set; } = NetworkConstants.DefaultMaxPeersPerIpAddress;
```

- aelf/src/aelf.OS.Core/Network/Infrastructure/PeerPool.cs

```
public bool IsFull()
{
var peerCount = Peers.Where(p => !p.Value.IsInvalid).ToList().Count;
return NetworkOptions.MaxPeers != 0 && peerCount >= NetworkOptions.MaxPeers;
}
//...code snip...
private bool IsOverIpLimit(string host)
{
if (NetworkOptions.MaxPeersPerIpAddress == 0 || host.Equals(IPAddress.Loopback.ToString()))
return false;
int initiatedHandshakes = 0;
if (HandshakingPeers.TryGetValue(host, out var handshakes))
initiatedHandshakes = handshakes.Count;
int peersFromIpCount = GetPeersByHost(host).Count;
if (peersFromIpCount + initiatedHandshakes >= NetworkOptions.MaxPeersPerIpAddress)
{
Logger.LogWarning($"Max peers from {host} exceeded, current count {peersFromIpCount} " +
$"(max. per ip {NetworkOptions.MaxPeersPerIpAddress}).");

return true;
}

return false;
}
```

默认最大连接数 25，默认不限制单 IP 最大连接数，女巫攻击成本较低，也可能导致拒绝服务攻击。

4.1.2 通信加密审计

使用 gRPC 协议通信，支持 TLS 加密通信

4.1.3 “异形攻击”审计

- aelf/src/aelf.OS.Core/Network/Protocol/HandshakeProvider.cs

```
public async Task<HandshakeValidationResult> ValidateHandshakeAsync(Handshake handshake)
{
var pubkey = handshake.HandshakeData.Pubkey.ToHex();
if (_networkOptions.AuthorizedPeers == AuthorizedPeers.Authorized &&
!_networkOptions.AuthorizedKeys.Contains(pubkey))
{
return HandshakeValidationResult.Unauthorized;
}

var chainId = _blockchainService.GetChainId();
if (handshake.HandshakeData.ChainId != chainId)
{
Logger.LogDebug($"Chain is is incorrect: {handshake.HandshakeData.ChainId}.");
return HandshakeValidationResult.InvalidChainId;
}
```

4.2 RPC 安全

4.2.1 远程调用权限审计

RPC 没有敏感权限，不存在“黑色情人节”漏洞。

参考文档：https://mp.weixin.qq.com/s/Kk2lsoQ1679Gda56Ec-zJg

4.2.2 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。

测试 Python PoC 示例如下：

```
posturl = "http://127.0.0.1:8001/api/blockChain/executeRawTransaction"
data = '{"' + '}' * 0x101000 + '":' + '{"Body":' * 0x10000 + '"}'
print post(posturl, data)
```

4.2.3 通信加密审计

使用 http 协议，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。

4.2.4 同源策略审计

节点 RPC 默认不开启跨域。

4.3 加密签名安全

4.3.1 随机数生成算法审计

RandomNumberGenerator 加密随机数生成器创建加密型强随机值。

- aelf/src/aelf.Cryptography/CryptoHelper.cs

```
public static ECKeyPair GenerateKeyPair()
{
try
{
Lock.AcquireWriterLock(Timeout.Infinite);
var privateKey = new byte[32];
var secp256K1PubKey = new byte[64];

// Generate a private key.
var rnd = RandomNumberGenerator.Create();
do
```

```
{
rnd.GetBytes(privateKey);
} while (!Secp256K1.SecretKeyVerify(privateKey));

if(!Secp256K1.PublicKeyCreate(secp256K1PubKey, privateKey))
throw new InvalidPrivateKeyException("Create public key failed.");
var pubKey = new byte[Secp256k1.SERIALIZED_UNCOMPRESSED_PUBKEY_LENGTH];
if(!Secp256K1.PublicKeySerialize(pubKey, secp256K1PubKey))
throw new PublicKeyOperationException("Serialize public key failed.");
return new ECKeyPair(privateKey, pubKey);
}
finally
{
Lock.ReleaseWriterLock();
}
}
```

参考：

https://docs.microsoft.com/zh-cn/dotnet/api/system.security.cryptography.randomnumbergenerator?view=netframework-4.7.2

4.3.2 密钥存储审计

aelf 中没有私钥管理模块。

4.3.3 密码学组件调用审计

Hash 算法：sha256

签名算法：secp256k1

签名模块: Secp256k1Net

```
public static byte[] SignWithPrivateKey(byte[] privateKey, byte[] hash)
{
try
{
Lock.AcquireWriterLock(Timeout.Infinite);
var recSig = new byte[65];
```

```
var compactSig = new byte[65];
if(!Secp256K1.SignRecoverable(recSig, hash, privateKey))
throw new SignatureOperationException("Create a recoverable ECDSA signature failed.");
if(!Secp256K1.RecoverableSignatureSerializeCompact(compactSig, out var recoverId, recSig))
throw new SignatureOperationException("Serialize an ECDSA signature failed.");
compactSig[64] = (byte) recoverId; // put recover id at the last slot
return compactSig;
}
finally
{
Lock.ReleaseWriterLock();
}
}

public static bool VerifySignature(byte[] signature, byte[] data, byte[] publicKey)
{
var recoverResult = RecoverPublicKey(signature, data, out var recoverPublicKey);
return recoverResult && publicKey.BytesEqual(recoverPublicKey);
}
```

未发现错误调用。

4.3.4 哈希强度审计

在密码学和计算机安全中，长度扩展攻击是一种攻击类型，攻击者可以使用哈希（message1）和 message1 的长度来计算攻击者控制的 message2 的 Hash（message1 ‖ message2），而无需知道 message1 内容。基于 Merkle－Damgard 构造的 MD5，SHA-1 和 SHA-2 等算法容易受到此类攻击。 SHA-3 算法不易受影响。

未发现错误调用。

4.3.5 交易延展性攻击审计

ECDSA 算法生成两个大整数 r 和 s 组合起来作为签名，可以用来验证交易。而 r 和 BN-s 也同样可以作为签名来验证交易。 这样，攻击者拿到一个交易，将其中 inputSig 的 r, s 提取出来， 使用 r, BN-s 生成新的 inputSig， 然后组成新的交易，拥有同样的 input 和 output，但是不同的 TXID. 攻击者能在不掌握私钥的情

况下几乎无成本地成功地生成了合法的交易。

aelf 使用 Secp256k1 算法签名，但交易结构没有使用脚本输入，且在计算 transactionId 之前将 signature 清空，确保签名的改变不会影响 transactionId。

- aelf/src/aelf.Types/Types/Transaction.cs

```
private byte[] GetSignatureData()
{
if (!VerifyFields())
throw new InvalidOperationException($"Invalid transaction: {this}");

if (Signature.IsEmpty)
return this.ToByteArray();

var transaction = Clone();
transaction.Signature = ByteString.Empty;
return transaction.ToByteArray();
}
```

漏洞参考：https://en.bitcoinwiki.org/wiki/Transaction_Malleability

4.4 账户与交易模型安全

4.4.1 交易校验审计

交易结构

- aelf/src/aelf.WebApp.Application.Chain/Dto/TransactionDto.cs

```
public string From { get; set; }
public string To { get; set; }
public long RefBlockNumber { get; set; }
public string RefBlockPrefix { get; set; }
public string MethodName { get; set; }
public string Params { get; set; }
```

- aelf/src/aelf.Kernel.TransactionPool/Infrastructure/TxHub.cs

交易签名及交易结构各字段的值均已严格校验。

4.4.2 交易重放审计

检测了交易 ID 是否存在，同一链上的交易无法重放；

- aelf/src/aelf.Kernel.TransactionPool/Infrastructure/TxHub.cs

```
await _blockchainService.HasTransactionAsync(queuedTransaction.TransactionId)
```

当存在 aelf folk 链时，RefBlockNumber、RefBlockPrefix 在不同的链上不可能相同，所以不同的链上不存在完全相同的交易，无法进行重放攻击。

4.4.3“假充值”审计

发起一笔转账测试：

```
aelf.ContractNames.Token Transfer
? Enter the required param <to>: 2mtJh6excBxbY7qq3r5xEBcZ3GsSmzB5JjcFnddA98da1uM
TxT
? Enter the required param <symbol>: ELF
? Enter the required param <amount>: 10000
? Enter the required param <memo>: 'first transfer'
The params you entered is:
{
  "to": "2mtJh6excBxbY7qq3r5xEBcZ3GsSmzB5JjcFnddA98da1uMTxT",
  "symbol": "ELF",
  "amount": 10000,
  "memo": "'first transfer'"
}
✓ Succeed!
aelf [Info]:
Result:
{
  "TransactionId": "5b39b9fd696cc9bf1e49fbc8c33d6dd46ea744919c1e35b890a784f105a397d1"
}
```

```
aelf-command send aelf.ContractNames.Token Transfer '{"symbol": "ELF", "to":
"2mtJh6excBxbY7qq3r5xEBcZ3GsSmzB5JjcFnddA98da1uMTxT", "amount": "10000"}'


aelf-command get-tx-result -e http://127.0.0.1:8001
aelf [Info]: {
  "TransactionId": "5b39b9fd696cc9bf1e49fbc8c33d6dd46ea744919c1e35b890a784f105a397d1",
  "Status": "MINED",
  "Logs": [
    {
      "Address": "JRmBduh4nXWi1aXgdUsj5gJrzeZb2LxmrAbf7W99faZSvoAaE",
      "Name": "TransactionFeeCharged",
      "Indexed": [],
      "NonIndexed": "CgNFTEYQ2Jy1DQ=="
    },
    {
      "Address": "JRmBduh4nXWi1aXgdUsj5gJrzeZb2LxmrAbf7W99faZSvoAaE",
      "Name": "Transferred",
      "Indexed": [
        "CilKIBceKvi1XdxdLxhFLw54VSs0bbreI/iUEZFnFqwcWolh",
        "EilKIOmcrBegmiYzVeZ80EmKSxV26soMf8Dyn1dhE8WcepbE",
        "GgNFTEY="
      ],
      "NonIndexed": "IJBOKhAnZmlyc3QgdHJhbnNmZXIn"
    }
  ],
  "Bloom":
"AAAAAAAAAAAAAAAAAAAAAAIAAQAIQAAAAAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAABAAAIAIAABAAAAAAAAEAAAAQAAAAAAAAQIAAAAAAAAAAAAAAAAIAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
A==",
  "BlockNumber": 341924,
  "BlockHash": "01cf9f1b48e5457337b4d77ca0a595a68a646af29377834b5a3b3e7c865e1e0f",
  "Transaction": {
    "From": "BBWq2c1Gi3NV34cgnBFtjNxyWATu3WNM3ebcb5ki8cHEB38hG",
    "To": "JRmBduh4nXWi1aXgdUsj5gJrzeZb2LxmrAbf7W99faZSvoAaE",
    "RefBlockNumber": 341922,
    "RefBlockPrefix": "lDcxHQ==",
    "MethodName": "Transfer",
    "Params": "{ \"to\": \"2mtJh6excBxbY7qq3r5xEBcZ3GsSmzB5JjcFnddA98da1uMTxT\", \"symbol\": \"ELF\", \"amount\":
\"10000\", \"memo\": \"'first transfer'\" }",
    "Signature":
"pJeSTZfg9VdOvfq/LlhRPWLwrdl8IMM/+p9HPvVuVXkzHvnQvRo8HSx4jQbnzSODEVCQaMZWQmMC5Z9A3QMWnAA="
  },
  "ReturnValue": "",
  "Error": null,
  "TransactionSize": 225
}
```


交易所在充值入账时不仅需要检验 Params 字段里的 to、symbol、amount，还需要校验调用的合约 Address 为 `JRmBduh4nXWi1aXgdUsj5gJrzeZb2LxmrAbf7W99faZSvoAaE` ，并检查 Status=="MINED"、ReturnValue==""和 Error==null，避免假充值攻击。

5 审计发现(Findings)

漏洞分布情况：

严重漏洞	0	
高危漏洞	0	

中危漏洞	1	■
低危漏洞	0	
弱点	1	■
增强建议	0	
合计	2	
■P2P 安全 ■RPC 安全 ■账户与交易模型安全 ■加密签名安全 ■其它		

5.1 P2P 未限制单 IP 连接数[中危]

默认最大连接数 25，默认不限制单 IP 最大连接数，女巫攻击成本较低，也可能导致拒绝服务攻击。

5.2 交易所“假充值”攻击风险[弱点]

交易所在充值入账时不仅需要检验 Params 字段里的 to、symbol、amount，还需要校验调用的合约 Address 为 `JRmBduh4nXWi1aXgdUsj5gJrzeZb2LxmrAbf7W99faZSvoAaE` ，并检查 Status=="MINED"、ReturnValue==""和 Error==null，避免假充值攻击。

6 修复情况(Fix Log)

6.1 修复 P2P 未限制单 IP 连接数问题

- 补丁 commit: 2697e64bae1f0bda8c9b9adeb1e6cb1d130ef058

- 补丁说明：将单 IP 最大连接数默认设置为 1

7 总结(Conclusion)

审计结果：通过

审计编号：BCA002011050001

审计日期：2020 年 11 月 05 日

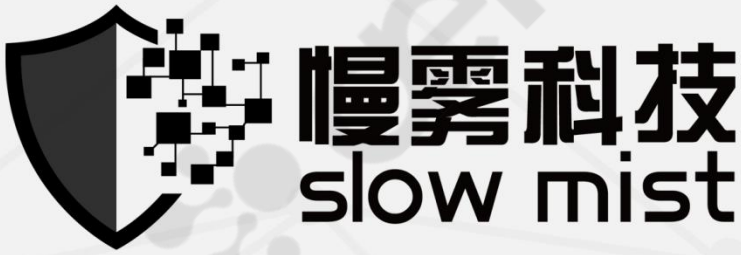
复审日期：2020 年 11 月 06 日

审计团队：慢雾安全团队

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

8 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