链安全审计报告

IPSE 公链安全审计报告

审计团队：零时科技安全团队

时间： 2021-05-10

IPSE 安全审计报告

1. 概述

零时科技安全团队于2021年4月6日至5月10日对 **IPSE** 项目进行了安全审计，这次审计主要关注于代码本身，对代码内容及引用库、依赖库的安全性进行了分析。跟进了挖矿流程中可能存在的安全性问题，也对常见的安卓漏洞进行了分析，在本次审计中，代码本身并未出现严重的安全问题，但在依赖库中发现了7个error级别的问题。App存在3个风险点，建议尽快对其中影响较大的error，使用对应提出的解决方法进行修复。

经过与 **IPSE** 项目方沟通反馈确认审计过程中发现的漏洞及风险均已修复或在可承受范围内，本次 **IPSE** 项目安全审计结果：**通过安全审计。**

审计报告MD5：C718CCA18ED54143EEBB2579C6F281DF

2. 项目背景

2.1 项目简介

> 项目名称：IPSE
>
> 项目官网：https://www.ipse.io/
>
> 代码仓库：https://github.com/IPSE-TEAM/ipse-core
>
> 审计版本：commit f3ac80074e6ed623c1d63ed3a036026eb6ab211c
>
> 主要编码语言：Rust

2.2 审计范围

IPSE公链代码审计目标：

> 公链代码仓库：https://github.com/IPSE-TEAM/ipse-core
>
> Python脚本文件：https://github.com/IPSE-TEAM/ipse2.0-mining
>
> 官方钱包APP：https://www.ipse.io/app/ipse.0.0.61.apk

2.3 安全审计项

零时科技安全团队对约定内的安全审计项目进行安全审计，本次安全审计的范围，不包含未来可能出现的新型攻击方式、升级活篡改后的代码、项目前端代码安全与项目平台服务器安全。

本次安全审计项目包括如下：

1. 代码合规审计

   代码相似度审计

代码补丁审计

路线图审计

充值方案审计

2. P2P 安全

节点连接数审计

节点性能审计

消息格式校验

消策略审计

通信加密审计

“异形攻击”审计

3. 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

4. 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

交易延展性审计

加解密模糊测试

5. 账户与交易模型安全

事务校验审计

事务重放审计

“假充值”审计

6. 静态代码检查

内置函数安全

标准库安全审计

第三方库安全审计

注入审计

序列化算法审计

内存泄露审计

算术运算审计

资源消耗审计

异常处理审计

日志安全审计

7. Python脚本安全审计

8. Android端APP安全测试

3. 架构分析

3.1 目录结构

```
├─.github
|  ├─ISSUE_TEMPLATE
|  └─workflows
├─.maintain
|  ├─chaostest
|  |  ├─bin
|  |      ├─commands
|  |      |  ├─clean
|  |      |  ├─singlenodeheight
|  |      |  └─spawn
|  |      ├─config
|  |      ├─hypervisor
|  |      |  ├─chainApi
|  |      |  ├─deployment
|  |      |  └─modules
|  |      └─utils
|  ├─docker
|  ├─github
|  ├─gitlab
|  ├─kubernetes
|  |  └─templates
|  ├─monitoring
|  |  ├─alerting-rules
|  |  └─grafana-dashboards
|  ├─node-template-release
|  |
|  └─sentry-node
|      ├─grafana
|      |  └─provisioning
|      |      ├─dashboards
|      |      └─datasources
|      └─prometheus
├─bin
|  ├─node
|  |  ├─bench
|  |  ├─browser-testing
|  |  ├─cli
|  |  |  ├─bin
|  |  |  ├─browser-demo
|  |  |  ├─doc
|  |  |  ├─res
|  |  |  ├─src
|  |  |  └─tests
|  |  ├─conjugate-poc
|  |  ├─executor
|  |  |  ├─benches
|  |  |  ├─src
|  |  |  └─tests
|  |  ├─inspect
|  |  ├─primitives
```

```
|   |   ├─rpc
|   |   ├─rpc-client
|   |   ├─runtime
|   |   |       ├─weights
|   |   |       └─文档
|   |   └─testing
|   ├─node-template
|   |   ├─node
|   |   ├─pallets
|   |   |   └─template
|   |   ├─runtime
|   |   └─scripts
|   └─utils
|       ├─chain-spec-builder
|       └─subkey
├─client
|   ├─api
|   ├─authority-discovery
|   |       └─worker
|   |           └─schema
|   ├─basic-authorship
|   ├─block-builder
|   ├─chain-spec
|   |   ├─derive
|   |   ├─res
|   ├─cli
|   |       ├─commands
|   |       └─params
|   ├─consensus
|   |   ├─aura
|   |   ├─babe
|   |   |   ├─rpc
|   |   ├─common
|   |   ├─epochs
|   |   ├─manual-seal
|   |   |       └─consensus
|   |   ├─pow
|   |   ├─slots
|   |   └─uncles
|   ├─db
|   |       └─cache
|   ├─executor
|   |   ├─common
|   |   ├─runtime-test
|   |   ├─src
|   |   |   └─integration_tests
|   |   ├─wasmi
|   |   └─wasmtime
|   |           └─instance_wrapper
|   ├─finality-grandpa
|   |   ├─rpc
|   |       └─communication
|   ├─informant
|   ├─keystore
|   ├─light
|   ├─network
|   |   ├─src
|   |   |   ├─gossip
```

```
|  |  |  ├─protocol
|  |  |  |  ├─generic_proto
|  |  |  |  |  ├─handler
|  |  |  |  |  └─upgrade
|  |  |  |  └─sync
|  |  |  ├─schema
|  |  |  └─service
|  |  └─test
|  ├─network-gossip
|  ├─offchain
|  |      └─api
|  ├─peerset
|  |  ├─src
|  |  └─tests
|  ├─proposer-metrics
|  ├─rpc
|  |      ├─author
|  |      ├─chain
|  |      ├─offchain
|  |      ├─state
|  |      └─system
|  ├─rpc-api
|  |      ├─author
|  |      ├─chain
|  |      ├─child_state
|  |      ├─offchain
|  |      ├─state
|  |      └─system
|  ├─rpc-servers
|  ├─service
|  |  ├─src
|  |  |  ├─chain_ops
|  |  |  ├─client
|  |  |  └─task_manager
|  |  └─test
|  ├─state-db
|  ├─telemetry
|  |      └─worker
|  ├─tracing
|  └─transaction-pool
|      ├─graph
|      |  ├─benches
├─docs
|  └─media
├─document
├─frame
|  ├─assets
|  ├─atomic-swap
|  ├─aura
|  ├─authority-discovery
|  ├─authorship
|  ├─babe
|  ├─balances
|  ├─benchmarking
|  ├─collective
|  ├─contracts
|  |  ├─common
|  |  ├─fixtures
```

```
|   |   |   └─benchmarks
|   |   ├─rpc
|   |   |   ├─runtime-api
|   |   |   └─wasm
|   |   |       └─env_def
|   ├─democracy
|   |   └─tests
|   ├─elections
|   ├─elections-phragmen
|   ├─evm
|   ├─example
|   ├─example-offchain-worker
|   ├─executive
|   ├─finality-tracker
|   ├─grandpa
|   ├─identity
|   ├─im-online
|   ├─indices
|   ├─membership
|   ├─metadata
|   ├─multisig
|   ├─nicks
|   ├─node-authorization
|   ├─offences
|   |   ├─benchmarking
|   ├─proxy
|   ├─randomness-collective-flip
|   ├─recovery
|   ├─scheduler
|   ├─scored-pool
|   ├─session
|   |   ├─benchmarking
|   |   └─historical
|   ├─society
|   ├─staking
|   |   ├─fuzzer
|   |   ├─reward-curve
|   ├─sudo
|   ├─support
|   |   ├─procedural
|   |   |   ├─src
|   |   |   |   ├─construct_runtime
|   |   |   |   └─storage
|   |   |   |       └─genesis_config
|   |   |   └─tools
|   |   ├─src
|   |   |   └─storage
|   |   |       └─generator
|   |   └─test
|   |       ├─src
|   |       └─tests
|   |           ├─construct_runtime_ui
|   |           ├─decl_module_ui
|   |           ├─decl_storage_ui
|   |           └─reserved_keyword
|   ├─system
|   |   ├─benches
|   |   ├─benchmarking
```

```
|  |  ├─rpc
|  |  |  └─runtime-api
|  |  |     └─extensions
|  ├─timestamp
|  ├─transaction-payment
|  |  ├─rpc
|  |  |  ├─runtime-api
|  ├─treasury
|  ├─utility
|  └─vesting
├─ipse-core
├─primitives
|  ├─allocator
|  ├─api
|  |  ├─proc-macro
|  |  |
|  |  ├─src
|  |  └─test
|  |     ├─benches
|  |     └─tests
|  |        └─ui
|  ├─application-crypto
|  |  ├─src
|  |  └─test
|  ├─arithmetic
|  |  ├─benches
|  |  ├─fuzzer
|  ├─authority-discovery
|  ├─authorship
|  ├─block-builder
|  ├─blockchain
|  ├─chain-spec
|  ├─consensus
|  |  ├─aura
|  |  |
|  |  ├─babe
|  |  |
|  |  ├─common
|  |  |     └─import_queue
|  |  ├─pow
|  |  ├─slots
|  |  └─vrf
|  ├─core
|  |  ├─benches
|  |  └─offchain
|  ├─database
|  ├─debug-derive
|  |  ├─src
|  |  └─tests
|  ├─externalities
|  ├─finality-grandpa
|  ├─finality-tracker
|  ├─inherents
|  ├─io
|  ├─keyring
|  ├─npos-elections
|  |  ├─benches
|  |  ├─compact
```

```
|  |  ├─fuzzer
|  ├─offchain
|  ├─panic-handler
|  ├─rpc
|  ├─runtime
|  |      ├─generic
|  |      └─offchain
|  ├─runtime-interface
|  |  ├─proc-macro
|  |  |      ├─pass_by
|  |  |      └─runtime_interface
|  |  ├─test
|  |  ├─test-wasm
|  |  ├─test-wasm-deprecated
|  |  └─tests
|  |      └─ui
|  ├─sandbox
|  ├─serializer
|  ├─session
|  ├─sr-api
|  |  └─proc-macro
|  ├─staking
|  ├─state-machine
|  |      ├─changes_trie
|  |      └─overlayed_changes
|  ├─std
|  ├─storage
|  ├─test-primitives
|  ├─timestamp
|  ├─tracing
|  ├─transaction-pool
|  ├─trie
|  |  ├─benches
|  |  ├─src
|  |  └─test-res
|  ├─utils
|  ├─version
|  └─wasm-interface
├─scripts
├─target
└─utils
    ├─browser
    ├─build-script-utils
    ├─fork-tree
    ├─frame
    |  ├─benchmarking-cli
    |  ├─frame-utilities-cli
    |  └─rpc
    |      ├─support
    |      └─system
    ├─prometheus
    ├─wasm-builder
    └─wasm-builder-runner
```

4. 审计详情

4.1 公链代码审计

4.1.1 Dangling reference in `access::Map` with Constant

> Crate: arc-swap
>
> Version: 0.4.7
>
> Date: 2020-12-10
>
> ID: RUSTSEC-2020-0091

Description:

Using the `arc_swap::access::Map` with the `Constant` test helper (or with user-provided implementation of the `Access` trait) could sometimes lead to the map returning dangling references.

Replaced by implementation without `unsafe`, at the cost of added `Clone` bound on the closure and small penalty on performance.

Solution:

Upgrade to >=1.1.0 OR >=0.4.8

4.1.2 `futures_task::waker` may cause a use-after-free if used on a type that isn't 'static

> Crate: futures-task
>
> Version: 0.3.5
>
> Date: 2020-09-04
>
> ID: RUSTSEC-2020-0060

Description:

Affected versions of the crate did not properly implement a `'static` lifetime bound on the `waker` function. This resulted in a use-after-free if `Waker::wake()` is called after original data had been dropped.

The flaw was corrected by adding `'static` lifetime bound to the data `waker` takes.

Solution:

Upgrade to >=0.3.6

4.1.3 `MutexGuard::map` can cause a data race in safe code

> Crate: futures-util
>
> Version: 0.3.5
>
> Title: MutexGuard::map can cause a data race in safe code

> Date: 2020-10-22
>
> ID: RUSTSEC-2020-0059

Description:

Affected versions of the crate had a Send/Sync implementation for MappedMutexGuard that only considered variance on T, while MappedMutexGuard dereferenced to U.

This could of led to data races in safe Rust code when a closure used in MutexGuard::map() returns U that is unrelated to T.

The issue was fixed by fixing `Send` and `Sync` implementations, and by adding a `PhantomData<&'a mut U>` marker to the `MappedMutexGuard` type to tell the compiler that the guard is over U too.

Solution:

Upgrade to >=0.3.7

4.1.4 arr! macro erases lifetimes

> Crate: generic-array
>
> Version: 0.12.3
>
> Title: arr! macro erases lifetimes
>
> Date: 2020-04-09
>
> ID: RUSTSEC-2020-0146

Description:

Affected versions of this crate allowed unsoundly extending lifetimes using `arr!` macro. This may result in a variety of memory corruption scenarios, most likely use-after-free.

Solution:

Upgrade to >=0.8.4, <0.9.0 OR >=0.9.1, <0.10.0 OR >=0.10.1, <0.11.0 OR >=0.11.2, <0.12.0 OR >=0.12.4, <0.13.0 OR >=0.13.3

4.1.5 Multiple Transfer-Encoding headers misinterprets request payload

> Crate: hyper
>
> Version: 0.12.35、0.13.7
>
> Title: Multiple Transfer-Encoding headers misinterprets request payload
>
> Date: 2021-02-05
>
> ID: RUSTSEC-2021-0020

Description:

hyper's HTTP server code had a flaw that incorrectly understands some requests with multiple transfer-encoding headers to have a chunked payload, when it should have been rejected as illegal. This combined with an upstream HTTP proxy that understands the request payload boundary differently can result in "request smuggling" or "desync attacks".

Solution:

Upgrade to >=0.14.3 OR >=0.13.10, <0.14.0 OR >=0.12.36, <0.13.0

4.1.6 Contents of uninitialized memory exposed in DeflateOutput's AsyncRead

> Crate: libp2p-deflate
>
> Version: 0.22.0
>
> Title: Contents of uninitialized memory exposed in DeflateOutput's AsyncRead implementation
>
> Date: 2020-01-24
>
> ID: RUSTSEC-2020-0123

Description:

Affected versions of this crate passes an uninitialized buffer to a user-provided trait function `AsyncRead::poll_read()`.

Arbitrary `AsyncRead::poll_read()` implementations can read from the uninitialized buffer (memory exposure) and also can return incorrect number of bytes written to the buffer. Reading from uninitialized memory produces undefined values that can quickly invoke undefined behavior.

The flaw was fixed in commit 5ba266a by ensuring the newly allocated part of the buffer is zero-initialized before passing it to a user-provided `AsyncRead::poll_read()`.

Solution:

Upgrade to >=0.27.1

4.1.7 Unexpected panic in multihash `from_slice` parsing code

> Crate: multihash
>
> Version: 0.11.2
>
> Title: Unexpected panic in multihash `from_slice` parsing code
>
> Date: 2020-11-08
>
> ID: RUSTSEC-2020-0068

Description:

In versions prior 0.11.3 it's possible to make `from_slice` panic by feeding it certain malformed input. It's never documented that `from_slice` (and `from_bytes` which wraps it) can panic, and its' return type (`Result<Self, DecodeError>`) suggests otherwise.

In practice, `from_slice`/`from_bytes` is frequently used in networking code (for example in rust-libp2p) and is being called with unsanitized data from untrusted sources. This can allow attackers to cause DoS by causing an unexpected `panic` in the network client's code.

Solution:

Upgrade to >=0.11.3

4.1.8 Soundness issues in `raw-cpuid`

> Crate: raw-cpuid
>
> Version: 7.0.3
>
> Title: Soundness issues in `raw-cpuid`
>
> Date: 2021-01-20
>
> ID: RUSTSEC-2021-0013

Description:

Undefined behavior in `as_string()` methods

`VendorInfo::as_string()`, `SoCVendorBrand::as_string()`, and `ExtendedFunctionInfo::processor_brand_string()` construct byte slices using `std::slice::from_raw_parts()`, with data coming from `#[repr(Rust)]` structs. This is always undefined behavior.

See https://github.com/gz/rust-cpuid/issues/40.

This flaw has been fixed in v9.0.0, by making the relevant structs `#[repr(C)]`.

`native_cpuid::cpuid_count()` is unsound

`native_cpuid::cpuid_count()` exposes the unsafe `__cpuid_count()` intrinsic from `core::arch::x86` or `core::arch::x86_64` as a safe function, and uses it internally, without checking the safety requirement:

> The CPU the program is currently running on supports the function being called.

CPUID is available in most, but not all, x86/x86_64 environments. The crate compiles only on these architectures, so others are unaffected.

This issue is mitigated by the fact that affected programs are expected to crash deterministically every time.

See https://github.com/gz/rust-cpuid/issues/41.

The flaw has been fixed in v9.0.0, by intentionally breaking compilation when targetting SGX or 32-bit x86 without SSE. This covers all affected CPUs.

Solution:

Upgrade to >=9.0.0

4.1.9 Buffer overflow in `SmallVec::insert_many`

> Crate: smallvec
>
> Version: 0.6.13、1.4.1
>
> Title: Buffer overflow in SmallVec::insert_many
>
> Date: 2021-01-08

> ID: RUSTSEC-2021-0003

Description:

A bug in the `SmallVec::insert_many` method caused it to allocate a buffer that was smaller than needed. It then wrote past the end of the buffer, causing a buffer overflow and memory corruption on the heap.

This bug was only triggered if the iterator passed to `insert_many` yielded more items than the lower bound returned from its `size_hint` method.

The flaw was corrected in smallvec 0.6.14 and 1.6.1, by ensuring that additional space is always reserved for each item inserted. The fix also simplified the implementation of `insert_many` to use less unsafe code, so it is easier to verify its correctness.

Solution:

Upgrade to >=0.6.14, <1.0.0 OR >=1.6.1

4.2 App安全审计

4.2.1 BroadcastReceiver组件暴露风险(风险)

漏洞描述：

以下广播可被外部调用导致敏感信息泄露,建议设置android:exported="false"，若需要外部调用，需自定义signature或者signatureOrSystem级别的权限
`com.dexterous.flutterlocalnotifications.ScheduledNotificationBootReceiver`

`\androidx\appcompat\app\g$l.smali`

`\io\flutter\plugins\urllauncher\WebViewActivity.smali`

`\io\sentry\android\core\SystemEventsBreadcrumbsIntegration.smali`

修复建议：

无需暴露的组件请设置`exported="false"`；若需要外部调用，建议添加自定义 signature 或 signatureOrSystem 级别的私有权限保护；需要暴露的组件请严格检查输入参数，避免应用出现拒绝服务。
进程内动态广播注册建议使用 LocalBroadcastManager；或者使用 `registerReceiver(BroadcastReceiver, IntentFilter, broadcastPermission, Handler)` 替换 `registerReceiver(BroadcastReceiver, IntentFilter)`.

4.2.2 stack-protector编译选项检测(风险)

漏洞描述：

启用该选项后编译器会产生额外的代码来检测缓冲区溢出，例如栈溢出攻击。这是通过在有缺陷的函数中添加一个保护变量来实现的。这包括会调用到alloca的函数，以及具有超过8个字节缓冲区的函数。当执行到这样的函数时，保护变量会得到初始化，而函数退出时会检测保护变量。如果检测失败，会输出一个错误信息并退出程序。

修复建议：

使用 NDK 编译 so 时，在 Android.mk 文件中添加： LOCAL_CFLAGS := -fstack-protector-all
lib/arm64-v8a/libapp.so

lib/arm64-v8a/libflutter.so

lib/arm64-v8a/libijkffmpeg.so

lib/armeabi-v7a/libapp.so

lib/armeabi-v7a/libijkffmpeg.so

lib/x86_64/libapp.so

lib/x86_64/libflutter.so

lib/x86_64/libijkffmpeg.so

4.2.3 PIE编译选项检测(风险)

漏洞描述：

Android 的动态链接器支持位置独立的可执行文件 (PIE)。 从 Android 5.0（API 级别 21）开始，可执行文件需要 PIE。要使用 PIE 构建可执行文件，请设置 `-fPIE` 标志。 此标志增大了通过随机化代码位置来利用内存损坏缺陷的难度。 默认情况下，如果项目针对 `android-16` 或更高版本，`ndk-build` 会自动将此值设置为 `true`。您可以手动将其设置为 `true` 或 `false`

修复建议：

使用 NDK 编译 so 时，在 Android.mk 文件中添加： LOCAL_CFLAGS := -fPIE
lib/arm64-v8a/libapp.so

lib/armeabi-v7a/libapp.so

lib/x86_64/libapp.so

4.3 Python脚本代码审计

4.3.1 未使用的变量(缺陷)

缺陷描述：

未使用的变量

位置：ipse3.2-mining/supervision.py

变量：result

```
        # 没有日志记录或是没有日志文件 说明没有启动软件
        except Exception as e:
            print("没有启动挖矿软件！", e)
            kill_process(SupervisionFileName, FileName)
            print("关闭挖矿软件!")
            result = os.system(r'{0} > {1}.log 2>&1 &'.format(FileName,
FileName))
            print("启动挖矿软件!")
            count = 0

        time.sleep(10)
```

修复建议：

删除不必要的变量

5. 安全审计工具

工具名称	功能
零时内部工具包	零时(鹰眼系统)自研发工具包
codeql	为全球安全研究人员提供支持的库和查询

6. 漏洞风险评估标准

高等危害

高等危害是指漏洞发生在核心系统业务逻辑（区块、交易、资金、共识验证处理等涉及核心资产与数据的逻辑），对整个区块链体系造成大量经济损失、大面积混乱、或获取节点宿主机权限等严重且多数不可逆的危害。

包括但不限于：

- 任意节点远程命令执行
- 区块链网络分叉
- 篡改历史区块数据
- 伪造、重放任意交易或区块并大量获益
- 获取任意节点托管的私钥
- 任意铸币、盗币
- 给任意账户造成资金损失
- 篡改鉴权、收费、转账等核心系统逻辑
- 破坏链上保密设计

中等危害

中等危害是指漏洞对部分节点或账户造成较严重危害，可以使部分区块链系统停滞，造成较大混乱或经济损失的问题。

包括但不限于：

- 任意节点程序崩溃或无响应
- 任意节点宿主机崩溃或无响应
- 使任意节点无法验收合法交易
- 使任意节点无法与其他节点维持任何有效连接
- 断开任意节点与其他节点的连接
- 伪造、重放任意交易或区块但无法大量获益
- 伪造签名、获得使用他人私钥给任意数据签名的能力
- 获取某些账户的私钥
- 获得少量非预期资金收益
- 给某些账户造成资金损失
- 越权修改账户地址或权限设置

低等危害

低等危害是指漏洞对部分节点或账户造成一定程度的混乱或经济损失的问题，不会对区块链系统、节点或账户造成实质性损害，但依然需要改进，具有潜在风险的问题。

包括但不限于：

- 重放特定交易或区块
- 使任意节点启动失败
- 使任意节点无法与其他节点建立有效连接
- 显著降低其他攻击的利用难度
- 使服务端RPC接口失效
- 不会直接造成经济损失的敏感信息泄漏
- 一定程度降低其他攻击的利用难度

零时科技
NONEAGE

咨询电话：86-17391945345 18511993344
邮　　箱：support@noneage.com
官　　网：www.noneage.com
微　　博：weibo.com/noneage