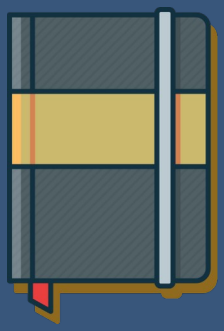
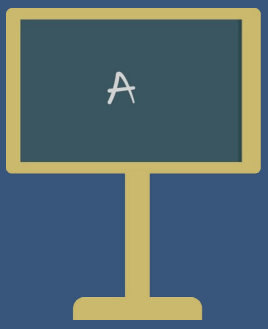
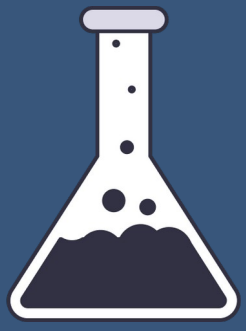
Perseus Audit

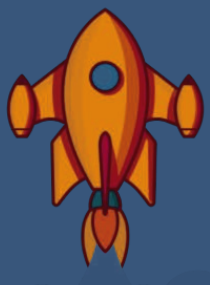
HNC Coin Audit October 2021

Contents









Introduction

Audit:

In October 2021 Perseus's audit report division performed an audit for the HNC Coin source code.

https://github.com/HellenicCoin-HNC

HNC Coin:

HNC Coin was established in 2015. First block was generated on Feb 8th 2015. The initial blockchain was a fork of Litecoin and after the last hard fork in July 2021 it became a fork of Dash Coin, focusing on maximizing safety and speed of transactions.

Introduction

Overview:

Information:
Name: HNC Coin

Supply:
Current: 93,486,639 HNC Coins
Max Supply: 100,000,000 HNC Coins

Explorers:
https://explorer.hnc-coin.com/

Websites:
https://hnc-coin.com

Links:
https://github.com/HellenicCoin-HNC

Audit Report Scope

Scope of services to be provided:

- Setup, compiling and test synchronization
- Preliminary Code Review
- Manual review line by line to identify any issues
- Live testing

Audit Report Scope

Categories:

High Severity:

High severity issues opens the code up for exploitation from malicious actors. We do not recommend using the code with high severity issues.

Medium Severity Issues:

Medium severity issues are errors found in the code that hampers the effectiveness of the code and may cause outcomes when interacting with the application. It is still recommended to fix these issues.

Low Severity Issues:

Low severity issues are warning of minor impact on the overall integrity of the code. These can be fixed with less urgency.

Audit Report

8	8	0
Identified	Confirmed	Critical

0	0	8
High	Medium	Low

Analysis:
https://github.com/HellenicCoin-HNC/source-code

Risk:
Low

Audit Report

Lines analyzed: 3824833
Physical Source Lines of Code (SLOC): 2793440

Minimum risk level: 1 (Low)

Low Severity Issues:

Setup, compiling and test synchronization:

Wrong file permission:

/autogen.sh
mode 100644
new mode 100755

/depends/config.guess
mode 100644
new mode 100755

/depends/config.sub
mode 100644
new mode 100755

/share/genbuild.sh
mode 100644
new mode 100755

Low Severity Issues:

(buffer) strlen: Does not handle strings that are not \0-terminated (it could cause a crash if unprotected).:

src/util.cpp:579:
```
if (pszHome == NULL || strlen(pszHome) == 0)
```

src/utilstrencodings.cpp:168:
```
vchRet.reserve(strlen(p)*3/4);
```

src/utilstrencodings.cpp:321:
```
vchRet.reserve((strlen(p))*5/8);
```

src/utilstrencodings.cpp:427:
```
if (str.size() != strlen(str.c_str()))
```

src/wallet/db.cpp:519:
```
strncmp(ssKey.data(), pszSkip, std::min(ssKey.size(), strlen(pszSkip))) == 0)
```

src/univalue/lib/univalue.cpp:25:
```
if (str.size() != strlen(str.c_str()))
```

src/stacktraces.cpp:94:
```
return strlen(buf);
```

src/rest.cpp:100:
```
if (strlen(rf_names[i].name) > 0) {
```

src/chainparams.cpp:29:
```
txNew.vin[0].scriptSig = CScript() << 486604799 << CScriptNum(4) <<
std::vector<unsigned char>((const unsigned char*)pszTimestamp, (const
unsigned char*)pszTimestamp + strlen(pszTimestamp));
```

src/base58.cpp:34:
```
int size = strlen(psz) * 733 /1000 + 1; // log(58) / log(256), rounded up.
```

Low Severity Issues:

(port) snprintf: On some very old systems, snprintf is incorrectly implemented and permits buffer overflows; there are also incompatible standard definitions of it. Check it during installation, or use something else.

src/test/dbwrapper_tests.cpp:280:
snprintf(buf, sizeof(buf), "%d", x);

src/test/dbwrapper_tests.cpp:296:
snprintf(buf, sizeof(buf), "%d", seek_start);

src/test/dbwrapper_tests.cpp:301:
snprintf(buf, sizeof(buf), "%d", x);

Low Severity Issues:

(buffer) strncpy: Easily used incorrectly; doesn't always \0-terminate or check for invalid pointers.

src/protocol.cpp:197:
strncpy(pchCommand, pszCommand, COMMAND_SIZE);

Low Severity Issues:

(access) umask: Ensure that umask is given most restrictive possible setting (e.g., 066 or 077).

src/init.cpp:1064:
umask(077);

Low Severity Issues:

(misc) fopen: Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents?.

src/addrdb.cpp:40:
FILE *file = fopen(pathTmp.string().c_str(), "wb");

src/addrdb.cpp:65:
FILE *file = fopen(pathBanlist.string().c_str(), "rb");

src/addrdb.cpp:137:
FILE *file = fopen(pathTmp.string().c_str(), "wb");

src/addrdb.cpp:162:
FILE *file = fopen(pathAddr.string().c_str(), "rb");

src/util.cpp:659:
FILE* configFile = fopen(GetConfigFile(confPath).string().c_str(), "a");

src/util.cpp:695:
FILE* file = fopen(path.string().c_str(), "w");

src/util.cpp:845:
file = fopen(pathLog.string().c_str(), "w");

Low Severity Issues:

(buffer) char: Statically-sized arrays can be overflowed. Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.

src/util.cpp:861:
char pszPath[MAX_PATH] = "";

src/util.cpp:899:
char name[16];

src/utilstrencodings.cpp:35:
const signed char p_util_hexdigit[256] =

src/utilstrencodings.h:99:
static const char hexmap[16] = { '0', '1', '2', '3', '4', '5', '6', '7',

src/validation.cpp:4209:
unsigned char buf[CMessageHeader::MESSAGE_START_SIZE];

src/wallet/crypter.cpp:27:
unsigned char buf[CSHA512::OUTPUT_SIZE];

src/protocol.h:40:
typedef unsigned char MessageStartChars[MESSAGE_START_SIZE];

src/pubkey.cpp:30:
unsigned char tmpsig[64] = {0};

Low Severity Issues:

(buffer) memcpy: Does not check for buffer overflows when copying to destination. Make sure destination can always hold the source data.

src/prevector.h:169, 186:
memcpy(dst, src, size() * sizeof(T));

src/pubkey.cpp:140:
memcpy(tmpsig + 32 - rlen, input + rpos, rlen);

src/pubkey.cpp:152:
memcpy(tmpsig + 64 - slen, input + spos, slen);

src/stacktraces.cpp:168:
memcpy(&context, pContext, sizeof(CONTEXT));

src/streams.h:59:
memcpy(vchData.data() + nPos, reinterpret_cast<const unsigned char*>(pch), nOverwrite);

src/streams.h:477:
unsigned char data[4096];

Conclusion

We performed the procedures as laid out in the scope of the audit and there were 8 findings, 8 low. The medium risk issues do not pose a security risk as they are best practice issues that is why the overall risk level is low.

Disclaimer

Perseus audit is not a security warranty, investment advice, or an endorsement of the HNC Coin platform. This audit does not provide a security or correctness guarantee of the audited code. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing code is a multistep process. One audit cannot be considered enough. We recommend that the the HNC Coin team put in place a bug bounty program to encourage further analysis of the code by other third parties.