# Zcash Cryptography and Code Review

## Zcash

October 10, 2016

Prepared for

Nathan Wilcox
Daira Hopwood
Zooko Wilcox

Prepared by

Alex Balducci
Robert Seacord

# Executive Summary

## Synopsis

In August 2016, Zcash engaged NCC Group to perform a targeted review of the Zcash cryptocurrency implementation. The review was performed in two parts, conducted simultaneously. The first part was conducted by NCC Group's Cryptography Services Practice, and was focused on validating that the implementation of Zcash adhered to the specification of the Zcash Protocol Specification version 2016.0-alpha-3.1-58-g871[1] as well as ensuring that the cryptographic implementation was free from security errors. The second part was a C++ source code review for vulnerabilities that can occur in unmanaged languages, focusing on the Zcash and libsnark libraries. The review also included a cursory review of dependent libraries and recommendations for improving software assurance practices at Zcash.

This assessment was atypical for NCC Group because it emphasized remediation over analyzing exploitability, including issues reported by tools. This means that less time was spent determining how specific security flaws might be exploited and more time identifying as many possible security issues and associated remediation as time allowed.

The engagement ran from August 8 to September 1, 2016, with the majority of the work conducted in the first three weeks. Two consultants were staffed on each part of the engagement, with one consultant acting as Technical Lead.

## Scope

NCC Group's evaluation included:

- Statically analyzing the code using Coverity Prevent, GCC, and LDRA.
- Instrumenting the code with AddressSanitizer, ThreadSanitizer, and UndefinedBehaviorSanitizer and performing dynamic analysis by running the full test suite and RPC tests.
- Fuzzing Zcash and the libsnark library using the American Fuzzy Lop (AFL) fuzzer.[2]
- Reviewing the Zcash specification and matching the implementation against it.
- Analyzing the constraints used for proving and verifying zkSNARKs. Zero-Knowledge Succinct Non-interactive Argument of Knowledge (zkSNARK) is the cryptographic tool underlying Zcash.

## Key Findings

NCC Group did not find any critical severity issues that would undermine the integrity of the Zcash blockchain or undermine the security of confidential transactions. NCC Group identified a number of C++ coding errors that could result in stack-based buffer overflows, data races, memory use-after-free issues, memory leaks, and other potentially exploitable runtime error conditions. Additionally, most, if not all, third-party open source library dependencies were identified as being out of date. Key findings from the engagement included:

- A potentially remotely accessible stack-based buffer overflow that could allow an attacker the ability to execute arbitrary code on a running `zcashd` host.
- Zcash zkSNARK `proving` and `verification` keys can be tampered with in transit during initial download because of missing verification logic.
- Out-of-date versions of third-party libraries were identified, including Berkeley DB, OpenSSL, and Boost. Many of these libraries have significantly revised the versions used by Zcash to address multiple CVEs.

## Limitations

Zcash is a complicated cryptographic protocol that builds upon cutting-edge academic research. NCC Group reviewed the protocol by comparing the specification to the implementation, but did not give the protocol and underlying cryptographic constructions the scrutiny it might receive in a rigorous academic review.

Because of the quantity of static and dynamic analysis diagnostics, some findings were not fully analyzed during the assessment. The raw data, however, was provided to Zcash for further analysis. Some effort was redirected to propose detailed remediations to the development team in the form of pull requests to improve the likelihood that the repairs would be made before the initial release of the product.

[1] https://github.com/zcash/zips/blob/zips27.reorganisation.0/protocol/protocol.pdf
[2] http://lcamtuf.coredump.cx/afl/

# Dashboard

## Target Metadata

| | |
|---|---|
| **Name** | Zcash |
| **Type** | Bitcoin-forked cryptocurrency |
| **Platforms** | C++ |
| **Environment** | Branch zc.v0.11.2.z8 |

## Engagement Data

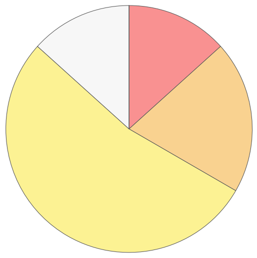
| | |
|---|---|
| **Type** | Cryptographic & Code Review |
| **Method** | Source Review and Static Analysis |
| **Dates** | 2016-08-08 to 2016-09-02 |
| **Consultants** | 4 |
| **Level of effort** | 45 person-days |

## Targets

| | |
|---|---|
| **Zcash v0.11.2.z8** | https://github.com/zcash/zcash/tree/v0.11.2.z8 |

## Vulnerability Breakdown

| | |
|---|---|
| Critical Risk issues | 0 |
| High Risk issues | 2 |
| Medium Risk issues | 3 |
| Low Risk issues | 8 |
| Informational issues | 2 |
| Total issues | 15 |



## Category Breakdown

| | | |
|---|---|---|
| Access Controls | 2 | 🟧 🟨 |
| Cryptography | 1 | 🟥 |
| Data Exposure | 2 | 🟧 🟨 |
| Data Validation | 4 | 🟨 🟨 🟨 ⬜ |
| Denial of Service | 1 | 🟨 |
| Patching | 1 | 🟥 |
| Timing | 2 | 🟧 🟨 |
| Other | 1 | ⬜ |
| Uncategorized | 1 | 🟨 |

## Component Breakdown

| | | |
|---|---|---|
| All | 2 | 🟥 🟨 |
| zcash | 2 | 🟥 🟨 |
| bitcoin | 9 | 🟧 🟧 🟧 🟨 🟨 🟨 🟨 🟨 ⬜ |
| libsnark | 2 | 🟨 🟨 |
| leveldb | 1 | 🟨 |

## Key

Critical 🟥 High 🟥 Medium 🟧 Low 🟨 Informational ⬜

## Cryptographic Review

### Specification Review

NCC Group's Cryptography Services reviewed Zcash's implementation of the *Zcash Protocol Specification* version 2016.0-alpha-3.1-58-g87[3] for inconsistencies and compared it to its precursor, described in *Zerocash: Decentralized Anonymous Payments from Bitcoin*.[4] The *Zcash Protocol Specification* is mostly complete; however, several important topics are implemented in the code but not yet fully documented in the specification[5]:

1. Coinbase and nonconfidential-to-confidential transactions are not adequately defined. When a transaction only has a `vpub_old`—a public input value and no confidential input notes—two *dummy* input notes are attached to the Zcash transaction. These dummy input notes have a value equal to zero and random values for the spending key (public and private portions), `rho` and `r`. While these values are random, the dummy note is still valid; that is, the nullifier and MAC are still well-formed.

2. Coinbase and nonconfidential-to-confidential transactions both skip the Merkle tree constraints. When a transaction does not have any previous confidential transactions associated with it, the Merkle tree checks (for example, does this note commitment have a valid path to the root of the tree?) must be skipped (because these are fake notes, they do not exist in the tree!). To simplify this process, an additional parameter `value_enforced` is included. When the transaction value is zero (that is, dummy input notes have been added), `value_enforced` is allowed to take the value of zero; when a transaction value is non-zero, `value_enforced` is set to one. The `value_enforced` argument is passed to the Merkle tree constraint system and will determine whether a valid path needs to be provided or not.

3. Coinbase transactions must be spent using a confidential transaction.

4. A founder's reward is taken from each block up until the first subsidy halving interval.

Issues 1, 2, and 4 listed above have been documented in the protocol specification since the original release of this report to Zcash.

### Implementation Review

NCC Group examined the implementation of the *Zcash Protocol Specification*. All details, cryptographic primitives, and constructions in the specification were in scope; however, an academic review of the scheme was not performed. NCC Group did not examine the specific encodings and serializations of addresses, keys, notes, and other items that were being changed.

#### Fairie Gold Attack

An initial focus of the review was on the Fairie Gold attack described in section 7.4 of the specification and an alternative attack put forward. In the original Zcash design, nullifiers of notes were derived from the spending key and a unique per note value `rho`. Because the value `rho` for each note is attacker controlled, the spender of a note can create multiple notes (spends) with the same nullifier; recipients of these crafted notes will not be able to spend them as the nullifier already exists in the nullifier set. To thwart this, Zcash forces the nullifiers to be unique by deriving them from several other parameters, including the nullifiers of the previous transaction, and enforces this in the zkSNARK circuit.

NCC Group also evaluated if an attacker can perform the same attack against coinbase transactions or nonconfidential-to-confidential transactions (where there are no previous nullifiers). For these two cases, because there are no confidential input notes, two fake, random, well-formed, zero-valued dummy notes are added to the transactions as described above. Without modification, the output notes will always have random nullifiers (because the input notes have random nullifiers) and the attack fails. The only way to repeat subsequent nullifiers (the goal of the attack) is to repeat the dummy input notes. However, repeating these dummy input notes will fail as nodes will flag this as a double-spend attempt (the nullifier is already present in the nullifier set). Because of this, a modified Fairie Gold attack

[3] https://github.com/zcash/zips/blob/zips27.reorganisation.0/protocol/protocol.pdf
[4] http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf
[5] It should be noted that the Zcash Protocol Specification highlights (literally) the differences between it and the Zerocash protocol.

is not possible against coinbase or nonconfidential-to-confidential transactions.

#### Walkthrough

NCC Group walked through each section of the specification and identified that the implementation matches each part. Furthermore, the constraint system used during proof creation and verification was reviewed. Each constraint was documented and compared to the JoinSplit circuit requirements in section 4.2.6 of the specification. Other constraints, such as booleanity and equivalence of packed and unpacked items were documented and determined to be correct. See Appendix F on page 61 for more details.

#### A Vulnerable Looking Function

During the review of the implementation, NCC Group identified an apparently dangerous encryption function that is actually completely safe. During note encryption, a symmetric key is generated by stretching a shared Diffie-Hellman secret, the spender's public key, the receiver's public key, `hSig`, and a nonce. Repeated calls to this encryption function will increase the nonce by one each time and keep all other parameters the same. The nonce is an `unsigned char` and thus the symmetric keys will be repeated after 256 encryption operations on implementations where `unsigned char` is represented by 8 bits. The symmetric encryption operation is ChaCha20, a stream cipher, resulting in a loss of confidentiality if one symmetric key encrypts two distinct notes. This is compounded by using a constant nonce for each operation. However, this function is not vulnerable to attack because of two design decisions that make such an attack impossible:

1. A compile time flag `ZC_NUM_JS_OUTPUTS` sets the limit of notes (and consequently, encryption operations) to two.

2. An `if` statement checks if the nonce has reached the maximum value `0xff` and if so, an error is thrown.

While this is not a vulnerability, *if* the limit on encrypted notes is ever increased, care should be taken when modifying this function. If the number of input / output notes on a transaction is increased, the size of the nonce *must* increase correspondingly.

#### libsnark Constraints

NCC Group reviewed the constraints used by the Zcash zkSNARK circuit and found that the necessary constraints exist to fulfill the obligations of the protocol. Additionally, a cursory review of constraints used by libsnark was performed. See Appendix F on page 61 for more details.

### Source Code Review

NCC Group analyzed the source code using a variety of static and dynamic analysis tools. Specifically, NCC Group:

1. Analyzed the code using a variety of additional GCC flags.

2. Instrumented the code using AddressSanitizer (ASan) and tested with full tests and RPC tests as described by Issue #777. The ASan memory error detector for C++ discovers:

   - Use after free (dangling pointer dereference)
   - Heap buffer overflow
   - Stack buffer overflow
   - Global buffer overflow
   - Use after return
   - Use after scope
   - Initialization order bugs
   - Memory leaks

3. Instrumented the code using ThreadSanitizer (TSan) and tested with full tests and RPC tests as described by Issue #1244. TSan is a data race detector for C++. Data races are one of the most common and hardest-to-debug types of bugs in concurrent systems. A data race occurs when two threads access the same variable concurrently and at least one of the accesses is write. Data races are undefined behavior in C++.

4. Instrumented the code using UndefinedBehaviorSanitizer (UBSan) and tested with full tests and RPC tests as described by Issue #1318. UBSan is a fast undefined behavior detector. UBSan modifies the program at compile time to catch various kinds of undefined behavior during program execution, for example:
    - Using misaligned or null pointer
    - Conversion to, from, or between floating-point types that would overflow the destination
5. Analyzed the 18,338,489 lines of code using Coverity Prevent, which identified 669 defects. These defects were triaged and 54 were dismissed. Defects in dependent but downstream versions of open source libraries were mostly unevaluated. Registered users of the service can sign in to Coverity Scan to view the results of these scans. To request access, contact Nathan Wilcox, Daira Hopwood, or another registered Zcash administrator.
6. Analyzed the code using the LDRA Testbed static analysis tool for violations of the *CERT C++ Coding Standard*[6] and *The CERT C Coding Standard*.[7]

NCC Group reviewed the output of these tools for true and false positives and in many cases provided patches to repair the defect or to silence false positive diagnostics. Additional recommendations are contained within this report.

NCC Group evaluated downstream dependencies used by Zcash, to determine if they should be repaired or upgraded including:

- libsnark
- Berkeley DB
- Boost
- OpenSSL

NCC Group provided diffs to update the build process to include recent version of these dependencies and tested the code with these updates.

[6] https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=637
[7] http://www.informit.com/store/cert-c-coding-standard-second-edition-98-rules-for-9780133805383

## Strategic Recommendations

**Integrate runtime analysis** Integrate AddressSanitizer, ThreadSanitizer, and UndefinedBehaviorSanitizer in the normal continuous integration and development process. Issue #777 describes how to modify the build process to test with AddressSanitizer. Issue #1244 describes how to modify the build process to test with ThreadSanitizer. Issue #1318 describes how to modify the build process to test with UndefinedBehaviorSanitizer.

**Use static analysis tools** Regularly run static analysis tools, such as the Coverity Prevent Scan,[8] and address any findings.

**Upgrade dependencies** Upgrade existing dependencies and create a process for identifying and upgrading dependent libraries to their latest versions. Rebaseline with Bitcoin, if feasible.

**Offer Training** Offer the "Secure Coding in C and C++" training available from NCC Group[9] or other providers to Zcash developers. Novice and intermediate developers can benefit from the entire four-day course while expert programmers may also benefit from training modules on Integers and Compiler Optimizations.

**Switch to C++11 threading** The C++11 threading capabilities update and replace those of Boost threading. Boost threading is less likely to be maintained now that this functionality has been incorporated into the standard. A similar transition is occurring in the Bitcoin code base (see Bitcoin Issue #6211, which will eventually necessitate this change. Consequently, rebaselining with Bitcoin should drive this change.

**Continue fuzzing libsnark using AFL** Using the drivers NCC Group has provided in Appendix E on page 58 as a base, continue fuzzing dependencies to identify malicious inputs and remediate any issues found.

**Complete the specification** As stated in Test Details on page 4, certain details of the specification are missing but have been implemented and some even have 'TODO' placeholders in the specification. All details of the protocol should be clearly documented within the specification.

**Perform a review of libsnark** NCC Group performed a cursory look at libsnark, mostly to gain an understanding of how it works, and not a full review. Consider performing a full review of libsnark functionality used by Zcash. In particular, confirm that all basic constraints work as assumed.

**Continue adhering to cryptographic best-practices** Both the *Zcash Protocol Specification* and implementation adhere to a high standard of quality. This results in an easy to audit codebase that is remarkably free of cryptographic errors. Continuing this practice will help Zcash hold up against attacks not known today.

**Consider implementing zkSNARK proofs in a memory safe language** The libsnark code is one of the more vulnerable components in the Zcash daemon. While great effort has been made to sanitize untrusted data passed into the third-party library, memory safety issues are still an area of concern. The Zcash team has been experimenting with implementing the zkSNARK proofs in Rust for verification and safety benefits. NCC Group supports replacing this code with a more secure implementation, if feasible.

[8] https://scan.coverity.com/
[9] https://www.nccgroup.trust/us/our-services/security-consulting/security-training/

## Table of Vulnerabilities

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. For an explanation of NCC Group's risk rating and vulnerability categorization, see Appendix A on page 39.

| Title | Status | ID | Risk |
|---|---|---|---|
| Use of Obsolete Open Source Libraries | Reported | 007 | High |
| Key Retrieval Script Subject to Man-in-the-Middle Attack | Reported | 017 | High |
| Lock Order Inversion | Reported | 006 | Medium |
| Heap Use After Free | Fixed | 011 | Medium |
| Stack-based Buffer Overflow in `LogPrint` | Reported | 014 | Medium |
| Direct Memory Leaks | Reported | 002 | Low |
| Conversion to `std::size_t` from `long int` | Reported | 003 | Low |
| Load of Misaligned Address for Type `const uint32_t` | Reported | 004 | Low |
| Multiple Data Races | Reported | 005 | Low |
| Uninitialized Objects | Reported | 008 | Low |
| Allocator/Deallocator Mismatch | Fixed | 010 | Low |
| Use of Reserved Identifiers | Reported | 012 | Low |
| Out-of-bounds Read in Boost `date` Class | Reported | 015 | Low |
| Dead Code | Reported | 009 | Informational |
| Generous Miners using Alternative Clients may Cause Unintentional Forks | Reported | 016 | Informational |

| | |
|---|---|
| **Vulnerability** | **Use of Obsolete Open Source Libraries** |
| Risk | High Impact: High, Exploitability: High |
| Identifier | NCC-Zcash2016-007 |
| Client Vulnerability ID | #1348, #1257, #759 |
| Status | Reported |
| Category | Patching |
| Component | All |
| Location | • depends/packages/bdb.mk • depends/packages/boost.mk • depends/packages/dbus.mk • depends/packages/openssl.mk • Other packages |
| Impact | Obsolete or outdated open source libraries can result in easily exploited vulnerabilities. These libraries are constantly evolving as defects and vulnerabilities are discovered in them. The history of these libraries, including any discovered vulnerabilities, is readily available, and exploits are frequently developed to target applications that use vulnerable versions of these libraries. |

**Description**

The depends/packages contains `*.mk` files for various packages including Berkeley DB, Boost, dbus, and OpenSSL. Many of these packages have not been updated for two years while none of them have been updated more recently than four months ago.

The recommendation section contains detailed recommendations for updating Berkeley DB and Boost but the remainder of these packages need to be evaluated as well. For example, the dbus package is currently at version 1.8.6 while the latest version from 2016-08-15 is 1.11.4.[10]

The OpenSSL package is currently at version 1.0.1 while the latest version is 1.0.2.[11] However, special care should be taken when upgrading OpenSSL because OpenSSL 1.1.0a, released on 22nd September 2016, contains a critical severity vulnerability where according to the OpenSSL Security Advisory[12]:

> a message larger than approx 16k is received then the underlying buffer to store the incoming message is reallocated and moved. A dangling pointer to the old location is left which results in an attempt to write to the previously freed location. This is likely to result in a crash; however, it could potentially lead to execution of arbitrary code.

As a result, OpenSSL 1.1.0 users should upgrade to 1.1.0b.

The CVE repository lists 24 vulnerabilities reported against the Berkeley DB in 2015[13] of which 23 have a CVSS Score of 6.9.

The Boost library has three reported CVEs[14] each with a CVSS Score of 5.0.

[10] https://dbus.freedesktop.org/releases/dbus/
[11] https://www.openssl.org/source/
[12] https://www.openssl.org/news/secadv/20160926.txt
[13] https://www.cvedetails.com/vulnerability-list/vendor_id-93/product_id-32070/Oracle-Berkeley-Db.html
[14] https://www.cvedetails.com/product/13033/Boost-Boost.html?vendor_id=7685

Zcash should also backport any security or other bug fixes from the Bitcoin 1.0 release codebase. See Issue #759.

**Recommendation**

NCC Group recommends following a continuous integration process whereby new, stable releases of open source libraries are continuously integrated into Zcash during development and locked down during final system test. The evolution of these libraries should be monitored and critical security patches applied. Examples of out-of-date open source libraries include Berkeley DB and Boost.

Upgrade from Berkeley DB v4.8 to the latest release, v6.2.23 at the time of writing (see Issue #1255). Among other changes (see Changes to upgrade bdb #1255), modify depends/packages/bdb.mk:

```
-$(package)_version=4.8.30
+$(package)_version=6.2.23
```

Upgrade from Boost version 1.57.0 to 1.61.0 (see Issue #944. Among other changes, modify depends/packages/boost.mk:

```
-$(package)_version=1_57_0
-$(package)_download_path=http://sourceforge.net/projects/boost/files/boost/1.57.0
+$(package)_version=1_61_0
+$(package)_download_path=https://sourceforge.net/projects/boost/files/boost/1.61.
➜   0
```

This upgrade requires a single change to the source code at src/wallet/db.cpp#L46:

```
     if (!fMockDb)
-        DbEnv(0).remove(strPath.c_str(), 0);
+        DbEnv(u_int32_t{0}).remove(strPath.c_str(), 0);
  }
```

The Boost upgrade also creates some problems with false positive deprecated header warnings (see Issue #1259) that should be resolved to improve the analyzability of the builds.

| | |
|---|---|
| **Vulnerability** | **Key Retrieval Script Subject to Man-in-the-Middle Attack** |
| **Risk** | High Impact: High, Exploitability: Medium |
| **Identifier** | NCC-Zcash2016-017 |
| **Client Vulnerability ID** | #1346 |
| **Status** | Reported |
| **Category** | Cryptography |
| **Component** | zcash |
| **Location** | zcutil/fetch-params.sh:17 |
| **Impact** | An adversary who is able to conduct a man-in-the-middle attack could tamper with verification and proving keys upon deploying the Zcash daemon. This exposes a variety of possible denial-of-service avenues, forgery scenarios, and memory safety issues that would otherwise be inaccessible when loading trusted keys. |

**Description**

The `fetch-params.sh` script retrieves the verification and proving keys from the `z.cash` domain and must be executed at least once for the daemon to function. While this connection is made over TLS, no certificate validation is performed because of a known `wget` discrepancy (see the code comments below). Rather, SHA256 checksums are computed and checked against known good values within the script body.

Unfortunately, these keys are copied and linked prior to validation and are not removed if the checksums do not match. The keys are loaded upon execution of the daemon regardless of prior validation result and could be tampered in transit, effectively bypassing the SHA256 checksums.

The following code segment highlights this issue. As shown below, the control flow of the script moves the keys prior to validation within the `fetch_params` function. Also note that there is no error handling logic to handle a failure condition during checksum validation defaulting to a warning message being displayed.

```
function fetch_params {
    local url="$1"
    local output="$2"
    local dlname="${output}.dl"

    if ! [ -f "$output" ]
    then
        echo "Retrieving: $url"
        # Note: --no-check-certificate should be ok, since we rely on
        # sha256 for integrity, and there's no confidentiality requirement.
        # Our website uses letsencrypt certificates which are not supported
        # by some wget installations, so we expect some cert failures.
        wget \
            --progress=dot:giga \
            --no-check-certificate \
            --output-document="$dlname" \
            --continue \
            "$url"

        # Only after successful download do we update the parameter load path:
        mv -v "$dlname" "$output"
    fi
}

...truncated for brevity...

mkdir -p "$REGTEST_DIR"

fetch_params "$REGTEST_PKEY_URL" "$REGTEST_DIR/$REGTEST_PKEY_NAME"
fetch_params "$REGTEST_VKEY_URL" "$REGTEST_DIR/$REGTEST_VKEY_NAME"

echo 'Updating testnet3 symlinks to regtest parameters.'
mkdir -p "$TESTNET3_DIR"
ln -sf "../regtest/$REGTEST_PKEY_NAME" "$TESTNET3_DIR/$REGTEST_PKEY_NAME"
ln -sf "../regtest/$REGTEST_VKEY_NAME" "$TESTNET3_DIR/$REGTEST_VKEY_NAME"

cd "$PARAMS_DIR"

# Now verify their hashes:
echo 'Verifying parameter file integrity via sha256sum...'
shasum -a 256 --check <<EOF
226913bbdc48b70834f8e044d194ddb61c8e15329f67cdc6014f4e5ac11a82ab  regtest/$REGTEST
➜   _PKEY_NAME
226913bbdc48b70834f8e044d194ddb61c8e15329f67cdc6014f4e5ac11a82ab  testnet3/$REGTES
➜   T_PKEY_NAME
4c151c562fce2cdee55ac0a0f8bd9454eb69e6a0db9a8443b58b770ec29b37f5  regtest/$REGTEST
➜   _VKEY_NAME
4c151c562fce2cdee55ac0a0f8bd9454eb69e6a0db9a8443b58b770ec29b37f5  testnet3/$REGTES
➜   T_VKEY_NAME
EOF
```

**Recommendation** Guarantee that corrupted keys are discarded upon checksum validation failure, and do not move keys until a successful verification has occurred. Alternatively, this routine could be implemented within the Zcash daemon to be performed during initialization.

Finally, to the extent that the underlying system supports it, the keys file and containing directories should have file permissions that prevent other users on the system from accessing the keys file.

| | |
|---|---|
| **Vulnerability** | **Lock Order Inversion** |
| **Risk** | **Medium** Impact: Medium, Exploitability: Medium |
| **Identifier** | NCC-Zcash2016-006 |
| **Client Vulnerability ID** | #1347 |
| **Status** | Reported |
| **Category** | Timing |
| **Component** | bitcoin |
| **Location** | • src/net.cpp:1762 • src/net.cpp:2049 • wallet/wallet.cpp:1406 |
| **Impact** | Zcash uses more than one lock at a time without an explicit lock order. Lock order inversion can cause deadlocks that are difficult to debug. For example, if lock A is acquired before lock B somewhere in the code, and lock B is acquired before lock C elsewhere in the code, then the lock order is A, B, C and this order should be followed throughout the code. Lock order inversion occurs when the locking order is not followed–for example, if lock B is acquired before lock A. If deadlock can be induced by an attacker, it can be used as a denial-of-service attack. |

**Description**

The cs_mapRelay lock is acquired at src/net.cpp:1762

```
void RelayTransaction(const CTransaction& tx, const CDataStream& ss)
{
    CInv inv(MSG_TX, tx.GetTxid());
    {
        LOCK(cs_mapRelay);
        // Expire old relay messages
```

The cs_mapRelay lock is acquired at src/net.cpp:2049

```
void CNode::BeginMessage(const char* pszCommand) EXCLUSIVE_LOCK_FUNCTION(cs_vSend)
{
    ENTER_CRITICAL_SECTION(cs_vSend);
    assert(ssSend.size() == 0);
    ssSend << CMessageHeader(Params().MessageStart(), pszCommand, 0);
    LogPrint("net", "sending: %s ", SanitizeString(pszCommand));
}
```

This lock is released in either CNode::AbortMessage() or CNode::EndMessage(), depending on how the message terminates.

The cs_wallet lock is acquired at wallet/wallet.cpp:1406

```
std::vector<uint256> CWallet::ResendWalletTransactionsBefore(int64_t nTime)
{
    std::vector<uint256> result;

    LOCK(cs_wallet);
```

**Reproduction Steps**

Run the full RPC Tests instrumented with ThreadSanitizer and search the output for "lock-order-inversion":

```
./qa/pull-tester/rpc-tests.sh
```

**Recommendation**

In the general case, all threads must acquire locks in the same order to avoid lock inversion.

The upstream Bitcoin master branch has a relevant pull request bitcoin/bitcoin#8606 This change ensures that the `cs_filter` lock is never held while taking the `cs_main` or `CNode::cs_vSend` locks.

```
   else if (inv.type == MSG_FILTERED_BLOCK)
     {
-     LOCK(pfrom->cs_filter);
-     if (pfrom->pfilter)
+      bool send = false;
+      CMerkleBlock merkleBlock;
          {
-          CMerkleBlock merkleBlock(block, *pfrom->pfilter);
+         LOCK(pfrom->cs_filter);
+           if (pfrom->pfilter) {
+             send = true;
+             merkleBlock = CMerkleBlock(block, *pfrom->pfilter);
+           }
+         }
+         if (send) {
              pfrom->PushMessage(NetMsgType::MERKLEBLOCK, merkleBlock);
```

Consequently, NCC Group recommends that this problem be addressed as part of the larger effort to rebaseline with the current bitcoin version.

| | |
|---|---|
| **Vulnerability** | **Heap Use After Free** |
| **Risk** | **Medium** Impact: Medium, Exploitability: Medium |
| **Identifier** | NCC-Zcash2016-011 |
| **Client Vulnerability ID** | #1241 |
| **Status** | Fixed |
| **Category** | Data Exposure |
| **Component** | bitcoin |
| **Location** | src/scheduler.cpp:58 |
| **Impact** | Reading deallocated memory can result in leaking sensitive information. |

Zcash enhances privacy for users by encrypting sender, amount, and recipient data within single-signature transactions published to its public blockchain ledger. Zcash privacy requires that the data itself is in need of protection.[15] Zcash does not encrypt data for multi-signature, protect against correlations made with public transactions (for example, when Zcash is traded to/from another cryptocurrency), or obfuscate IP addresses.

In general, information leaks can compromise security by leaking confidential or sensitive information including keys, passwords, and sensitive data. The information leak described in this vulnerability finding is in the Boost library and so it is difficult to determine everywhere this code is used, what types of information may be leaked, and the method by which they might be leaked to an attacker.

**Description** The following code at src/scheduler.cpp:58 results in a read of 8 bytes after the memory has been deallocated:

```
while (!shouldStop() && !taskQueue.empty() &&
  newTaskScheduled.wait_until<>(lock, taskQueue.begin()->first) !=
  boost::cv_status::timeout) {
    // Keep waiting until timeout
}
```

See Issue #1241 for a detailed stack trace.

This has defect was reported against the Boost C++ libraries as Ticket #11097. This bug was opened on March 10, 2015 with little progress before August 8, 2016, after which some additional prodding by Daira Hopwood of Zcash helped get the issue resolved.

Boost developers identified the following call to `super::data_.top().time` was passing the time by reference when it actually needed to be copied as the storage can be rearranged:

```
        super::not_empty_.wait_until(lk, super::data_.top().time);
```

The following patch was applied to repair the issue:

```
  template <class T, class Clock>
  bool sync_timed_queue<T, Clock>::wait_until_not_empty_time_reached_or_
➜   closed(unique_lock<mutex>& lk)
  {
    for (;;)
```

[15] https://z.cash/support/faq.html#does-zcash-offer-total-anonymity-for-transactions

```
    {
      if (super::closed(lk)) return true;
      while (! super::empty(lk)) {
        if (! time_not_reached(lk)) return false;
-         super::not_empty_.wait_until(lk, super::data_.top().time);
+         time_point tp = super::data_.top().time;
+         super::not_empty_.wait_until(lk, tp);
        if (super::closed(lk)) return true;
      }
      if (super::closed(lk)) return true;
      super::not_empty_.wait(lk);
    }
    //return false;
  }
```

See the following commit for additional details.

**Reproduction Steps** Run the full test suite instrumented with ThreadSanitizer and search the output for "heap-use-after-free":

```
./qa/zcash/full-test-suite.sh
```

**Recommendation** Upgrade to Boost 1.62.0 to integrate the repair.

| | |
|---|---|
| **Vulnerability** | **Stack-based Buffer Overflow in** `LogPrint` |
| **Risk** | **Medium** Impact: Medium, Exploitability: Medium |
| **Identifier** | NCC-Zcash2016-014 |
| **Client Vulnerability ID** | #1349 |
| **Status** | Reported |
| **Category** | Access Controls |
| **Component** | bitcoin |
| **Location** | • tinyformat.h:465<br>• src/util.h:90<br>• src/tinyformat.h:L901 |

**Impact** Stack-based buffer overflows can frequently be exploited to execute arbitrary code with the permissions of the vulnerable process. The `LogPrint` feature of Zcash is a potentially remotely-accessible attack surface.

**Description** The following macro invocation appears at locationutil.h:90:

```
TINYFORMAT_FOREACH_ARGNUM(MAKE_ERROR_AND_LOG_FUNC)
```

The macro `TINYFORMAT_FOREACH_ARGNUM` generates 16 user defined function bodies:

```
#define TINYFORMAT_FOREACH_ARGNUM(m) \
    m(1) m(2) m(3) m(4) m(5) m(6) m(7) m(8) m(9) m(10) m(11) m(12) m(13) m(14) m(1
➜   5) m(16)
```

A section of the `MAKE_ERROR_AND_LOG_FUNC` macro appears as follows:

```
#define MAKE_ERROR_AND_LOG_FUNC(n)                                         \
    /**   Print to debug.log if -debug=category switch is given OR category is NUL
➜   L. */ \
    template<TINYFORMAT_ARGTYPES(n)>                                           \
    static inline int LogPrint(const char* category, const char* format, TINYFORMA
➜   T_VARARGS(n))   \
    {                                                                          \
        if(!LogAcceptCategory(category)) return 0;                             \
        return LogPrintStr(tfm::format(format, TINYFORMAT_PASSARGS(n))); \
    }
```

The `format` method calls `FormatIterator` at location tinyformat.h:901:

```
void format(std::ostream& out, const char* fmt, const T1& v1, const Args&... args)
{
    detail::FormatIterator fmtIter(out, fmt);
    format(fmtIter, v1, args...);
}
```

A stack-based buffer overflow is present in the `tinyformat::detail::FormatIterator::FormatIterator(std::ostream& out, const char* fmt)` method of tinyformat.h:465:

```
// out is the output stream, fmt is the full format string
FormatIterator(std::ostream& out, const char* fmt)
    : m_out(out),
    m_fmt(fmt),
    m_extraFlags(Flag_None),
    m_wantWidth(false),
    m_wantPrecision(false),
    m_variableWidth(0),
    m_variablePrecision(0),
    m_origWidth(out.width()),
    m_origPrecision(out.precision()),
    m_origFlags(out.flags()),
    m_origFill(out.fill())
  { }
```

See Appendix C on page 53 for the complete diagnostic issued by AddressSanitizer.

**Reproduction Steps** Run the full RPC Tests instrumented with AddressSanitizer and search the output for "stack-buffer-overflow":

```
./qa/pull-tester/rpc-tests.sh
```

**Recommendation** These functions have been largely rewritten in the upstream versions of these files in the Bitcoin master tree. Integrate changes from src/util.cpp, src/util.h, src/tinyformat.h, src/utiltime.cpp, and src/utiltime.h.

## Vulnerability Direct Memory Leaks

**Risk** Low Impact: Low, Exploitability: Low

**Identifier** NCC-Zcash2016-002

**Client Vulnerability ID** #1217, #1218, #1225

**Status** Reported

**Category** Denial of Service

**Component** libsnark

**Location**
- zcash/JoinSplit.cpp:121
- zcash/JoinSplit.cpp:122

**Impact** Memory leaks can be used by an attacker to create a resource exhaustion attack. This is a primarily a concern if the library is used as a component of a long running service.

**Description** The proving and verification keys are generated on lines 119-122 of file JoinSplit.cpp:

```
void generate() {
  protoboard<FieldT> pb;

  joinsplit_gadget<FieldT, NumInputs, NumOutputs> g(pb);
  g.generate_r1cs_constraints();

  const r1cs_constraint_system<FieldT> constraint_system =
    pb.get_constraint_system();
  r1cs_ppzksnark_keypair<ppzksnark_ppT> keypair =
    r1cs_ppzksnark_generator<ppzksnark_ppT>(constraint_system);

  pk = keypair.pk;
  vk = keypair.vk;
}
```

The proving and verification keys are leaked when the `generate()` method exits. This results in a direct leak of 328,652,640 bytes in 1 object allocated in `operator new(unsigned long)` (see Issue #1217), a direct leak of 323860224 byte(s) in 1 object(s) allocated in `operator new(unsigned long)` (see Issue #1218), and a direct leak of 864 bytes in 1 object allocated from `operator new(unsigned long)` (see Issue #1225) for detailed diagnostics.

**Reproduction Steps** Run the full test suite instrumented with AddressSanitizer and search the output for "detected memory leaks":

```
./qa/zcash/full-test-suite.sh
```

**Recommendation** Memory associated with the proving and verification keys must be either statically allocated (provided no more than one copy is required) or deallocated when no longer required.

| | |
|---|---|
| **Vulnerability** | **Conversion to `std::size_t` from `long int`** |
| **Risk** | Low Impact: Low, Exploitability: Low |
| **Identifier** | NCC-Zcash2016-003 |
| **Client Vulnerability ID** | #1240 |
| **Status** | Reported |
| **Category** | Access Controls |
| **Component** | libsnark |
| **Location** | • alt_bn128_pairing.cpp:327:38 • alt_bn128_pairing.cpp:329:47 • alt_bn128_pairing.cpp:381:38 • alt_bn128_pairing.cpp:383:47 • alt_bn128_pairing.cpp:435:3 • alt_bn128_pairing.cpp:437:47 • field_utils.tcc:175:29 • field_utils.tcc:177:34 • field_utils.tcc:178:12 • field_utils.tcc:178:36 |

**Impact** The `long int` type is a signed, implementation-defined type that must minimally support values in the range of -2,147,483,647 to +2,147,483,647. The `size_t` type is an unsigned type that can represent the size of any object that can be allocated by an implementation. The `size_t` type is typically a 32-bit unsigned value (0 to 4,294,967,295) on implementations that support 32-bit addressing and a 64-bit unsigned value (0 to 18,446,744,073,709,551,615) on implementations that support 64-bit addressing. If the `size_t` bound of the loop is greater than `LONG_MAX`, `long int` iterators are likely to wrap before reaching the bound, which will result in negative indices and out-of-bounds memory accesses.

**Description** The following diagnostics are generated by gcc when compiling with the `-Wsign-conversion` flag:

```
src/algebra/curves/alt_bn128/alt_bn128_pairing.cpp:327:38: warning: conversion to
→  long int from size_t {aka long unsigned int} may change the sign of the result
→  [-Wsign-conversion]
     for (long i = loop_count.max_bits(); i >= 0; --i)
                                       ^
src/algebra/curves/alt_bn128/alt_bn128_pairing.cpp:329:47: warning: conversion to
→  std::size_t {aka long unsigned int} from long int may change the sign of the r
→  esult [-Wsign-conversion]
         const bool bit = loop_count.test_bit(i);
                                               ^
src/algebra/curves/alt_bn128/alt_bn128_pairing.cpp: In function libsnark::alt_bn12
→  8_Fq12 libsnark::alt_bn128_ate_miller_loop(const libsnark::alt_bn128_ate_G1_pr
→  ecomp&, const libsnark::alt_bn128_ate_G2_precomp&):
src/algebra/curves/alt_bn128/alt_bn128_pairing.cpp:381:38: warning: conversion to
→  long int from size_t {aka long unsigned int} may change the sign of the result
→  [-Wsign-conversion]
      for (long i = loop_count.max_bits(); i >= 0; --i)
                                       ^
src/algebra/curves/alt_bn128/alt_bn128_pairing.cpp:383:47: warning: conversion to
→  std::size_t {aka long unsigned int} from long int may change the sign of the r
→  esult [-Wsign-conversion]
         const bool bit = loop_count.test_bit(i);
                                               ^
src/algebra/curves/alt_bn128/alt_bn128_pairing.cpp: In function libsnark::alt_bn12
→  8_Fq12 libsnark::alt_bn128_ate_double_miller_loop(const libsnark::alt_bn128_at
→  e_G1_precomp&, const libsnark::alt_bn128_ate_G2_precomp&, const libsnark::alt_
→  bn128_ate_G1_precomp&, const libsnark::alt_bn128_ate_G2_precomp&):
```

```
src/algebra/curves/alt_bn128/alt_bn128_pairing.cpp:435:38: warning: conversion to
➜  long int from size_t {aka long unsigned int} may change the sign of the result
➜  [-Wsign-conversion]
     for (long i = loop_count.max_bits(); i >= 0; --i)
                                      ^
src/algebra/curves/alt_bn128/alt_bn128_pairing.cpp:437:47: warning: conversion to
➜  std::size_t {aka long unsigned int} from long int may change the sign of the r
➜  esult [-Wsign-conversion]
         const bool bit = loop_count.test_bit(i);
                                               ^
src/algebra/fields/field_utils.tcc:175:29: warning: conversion to  long int from
➜  std::vector<libsnark::Fp2_model<4l, ((const libsnark::bigint<4l>&)(& libsnar
➜  k::alt_bn128_modulus_q))> >::size_type {aka long unsigned int} may change the
➜  sign of the result [-Wsign-conversion]
     for (long i = vec.size()-1; i >= 0; --i)
                             ^
src/algebra/fields/field_utils.tcc:177:34: warning: conversion to  std::vector<lib
➜  snark::Fp2_model<4l, ((const libsnark::bigint<4l>&)(& libsnark::alt_bn128_modu
➜  lus_q))> >::size_type {aka long unsigned int} from  long int may change the si
➜  gn of the result [-Wsign-conversion]
         const FieldT old_el = vec[i];
                                  ^
src/algebra/fields/field_utils.tcc:178:12: warning: conversion to  std::vector<lib
➜  snark::Fp2_model<4l, ((const libsnark::bigint<4l>&)(& libsnark::alt_bn128_modu
➜  lus_q))> >::size_type {aka long unsigned int} from  long int may change the si
➜  gn of the result [-Wsign-conversion]
         vec[i] = acc_inverse * prod[i];
            ^
src/algebra/fields/field_utils.tcc:178:36: warning: conversion to  std::vector<lib
➜  snark::Fp2_model<4l, ((const libsnark::bigint<4l>&)(& libsnark::alt_bn128_modu
➜  lus_q))> >::size_type {aka long unsigned int} from  long int may change the si
➜  gn of the result [-Wsign-conversion]
         vec[i] = acc_inverse * prod[i];
                                    ^
```

**Reproduction Steps** Compile with GCC using the `-Wsign-conversion` flag.

**Recommendation** Declare the loop counter as `size_t` when the loop bound can be greater than `LONG_MAX`, for example:

```
-    for (long i = loop_count.max_bits(); i >= 0; --i)
+    for (size_t i = loop_count.max_bits(); i >= size_t{0}; --i)
```

| | |
|---|---|
| **Vulnerability** | **Load of Misaligned Address for Type** `const uint32_t` |
| **Risk** | Low  Impact: Low, Exploitability: Low |
| **Identifier** | NCC-Zcash2016-004 |
| **Client Vulnerability ID** | #1246 |
| **Status** | Reported |
| **Category** | Data Validation |
| **Component** | bitcoin |
| **Location** | • uint256.h:22<br>• uint256.cpp:129 |
| **Impact** | Object types have alignment requirements that place restrictions on the addresses at which an object of that type may be allocated. A pointer to an object type may be converted to a pointer to a different object type. If the resulting pointer is not correctly aligned for the referenced type, the behavior is undefined. Undefined behavior allows the compiler to ignore the situation completely, with unpredictable results. Frequently, misaligned access results in crashes and performance issues. The type mismatch, that is the underlying cause of the alignment error, can also result in invalid optimizations resulting from incorrect aliasing assumptions. |

**Description** Line 22 of file uint256.h includes the following definition:

```
    uint8_t data[WIDTH];
```

Note that `data` is defined as an array of `uint8_t` .

The `uint256::GetHash` function is defined at uint256.cpp:129:

```
uint64_t uint256::GetHash(const uint256& salt) const
{
    uint32_t a, b, c;
    const uint32_t *pn = (const uint32_t*)data;
    const uint32_t *salt_pn = (const uint32_t*)salt.data;
    a = b = c = 0xdeadbeef + WIDTH;

    a += pn[0] ^ salt_pn[0];
    b += pn[1] ^ salt_pn[1];
    c += pn[2] ^ salt_pn[2];
    HashMix(a, b, c);
    a += pn[3] ^ salt_pn[3];
    b += pn[4] ^ salt_pn[4];
    c += pn[5] ^ salt_pn[5];
    HashMix(a, b, c);
    a += pn[6] ^ salt_pn[6];
    b += pn[7] ^ salt_pn[7];
    HashFinal(a, b, c);

    return ((((uint64_t)b) << 32) | c);
}
```

The cast of `data` to `const uint32_t*` in the second line of the function creates a misaligned pointer and is thus undefined behavior.

**Reproduction Steps** Run the full RPC Tests instrumented with UndefinedBehaviorSanitizer and search the output for "load of misaligned address":

```
./qa/pull-tester/rpc-tests.sh
```

**Recommendation** Align `data` to have the same alignment as `uint32_t`:

```
{
  protected:
    enum { WIDTH=BITS/8 };
-   uint8_t data[WIDTH];
+   alignas(uint32_t) uint8_t data[WIDTH];
  public:
    base_blob()
    {
```

| | |
|---|---|
| **Vulnerability** | **Multiple Data Races** |
| **Risk** | Low  Impact: Medium, Exploitability: Low |
| **Identifier** | NCC-Zcash2016-005 |
| **Client Vulnerability ID** | #1247, #1280, #1281, #1286 |
| **Status** | Reported |
| **Category** | Timing |
| **Component** | bitcoin |

**Location**

- src/init.cpp:116
- src/main.cpp:1430
- src/main.cpp:4268
- src/net.cpp:75
- src/net.cpp:106
- src/net.cpp:501
- src/net.h:147
- src/net.h:169
- src/net.h:241
- src/net.h:250
- src/net.h:253
- src/net.h:260
- src/net.h:316
- src/rpcnet.cpp:138
- src/wallet/db.h:23
- src/net.h:261

**Impact** The execution of a program contains a data race if it contains two conflicting actions in different threads, at least one of which is not atomic, and neither happens before the other. Any such data race results in undefined behavior. The compiler can ignore undefined behavior completely, with unpredictable results. Data races can frequently result in reading and writing corrupted values.

**Description** Zcash makes use of both pthreads and OpenMP for parallel execution. Instrumenting the code using ThreadAnalyzer (see Issue #1244 for details of the test configuration) resulted in the generation of ~10.5MB of diagnostic information. Implementation of the recommended repairs in for this finding reduced the size of this file to 62KB. Further repairs are required to eliminate the remaining data races.

**Reproduction Steps** Run the full RPC Tests instrumented with ThreadSanitizer and search the output for "data race":

```
./qa/pull-tester/rpc-tests.sh
```

**Recommendation** Most of the data races involved reads and writes of 1, 4, or 8 bytes. These problems are best remedied by making the associated Boolean, integer, and pointer objects atomic. Atomic types are types that encapsulate a value whose access is guaranteed to not cause data races and can be used to synchronize memory accesses among different threads.

In some cases, atomic types cannot be easily used because they are not supported by the JSON Spirit library or the serialization code used by Bitcoin. In other documented cases, this problem can be solved by introducing Critical Sections. However, this is a heavy weight solution to this problem which might be better addressed by the use of atomics if the JSON Spirit and Serialization libraries are updated or replaced. Replacing the JSON-spirit API[16] with the UniValue API may solve this problem, but has not been tested. This change is also necessary to address a separate stack overflow issue (see Issue #1122).

[16] http://www.codeproject.com/Articles/20027/JSON-Spirit-A-C-JSON-Parser-Generator-Implemented

In src/init.cpp:116 make fRequestShutdown atomic:

```
-volatile bool fRequestShutdown = false;
+volatile std::atomic<bool>fRequestShutdown(false);
```

In src/main.cpp:1430 make both fLargeWorkForkFound and fLargeWorkInvalidChainFound atomic:

```
-bool fLargeWorkForkFound = false;
-bool fLargeWorkInvalidChainFound = false;
+std::atomic<bool>fLargeWorkForkFound(false);
+std::atomic<bool> fLargeWorkInvalidChainFound(false);
```

In src/main.cpp:4268 acquire and release the lock for the cs_pfrom CCriticalSection.

```
bool static ProcessMessage(CNode* pfrom, string strCommand, CDataStream& vRecv,
➜   int64_t nTimeReceived)
 {
+    LOCK(cs_pfrom);
     const CChainParams& chainparams = Params();
     RandAddSeedPerfmon();
```

In src/net.cpp:75 create a CCriticalSection object cs_pfrom.

```
+CCriticalSection cs_pfrom;
```

In src/net.cpp:106 create a CCriticalSection object cs_CNodeStatus.

```
+CCriticalSection cs_CNodeStatus;
```

In src/net.cpp:501 acquire and release the lock for the cs_CNodeStatus CCriticalSection:

```
void CNode::copyStats(CNodeStats &stats)
 {
+    LOCK(cs_CNodeStatus);
     stats.nodeid = this->GetId();
```

In src/net.h:147 create an external reference for cs_pfrom:

```
+extern CCriticalSection cs_pfrom;
```

In src/net.h:169 create an external reference for cs_CNodeStatus:

```
+extern CCriticalSection cs_CNodeStatus;
```

In src/net.h:241 make nSendSize, nSendOffset, and nSendBytes atomic types:

```
-    size_t nSendSize; // total size of all vSendMsg entries
-    size_t nSendOffset; // offset inside the first vSendMsg already sent
-    uint64_t nSendBytes;
+    std::atomic<size_t> nSendSize; // total size of all vSendMsg entries
+    std::atomic<size_t> nSendOffset; // offset inside the sent vSendMsg
+    std::atomic<uint64_t> nSendBytes;
```

In src/net.h:250 make nRecvBytes and nRecvVersion atomic types:

```
-   uint64_t nRecvBytes;
-   int nRecvVersion;
+   std::atomic<uint64_t> nRecvBytes;
+   std::atomic<int> nRecvVersion;
```

In src/net.h:253 make nLastSend, nLastRecv, nTimeConnected, and nTimeOffset atomic types:

```
-   int64_t nLastSend;
-   int64_t nLastRecv;
-   int64_t nTimeConnected;
-   int64_t nTimeOffset;
+   std::atomic<int64_t> nLastSend;
+   std::atomic<int64_t> nLastRecv;
+   std::atomic<int64_t> nTimeConnected;
+   std::atomic<int64_t> nTimeOffset;
```

In src/net.h:260 make fWhitelisted, fOneShot, fClient, fInbound, fNetworkNode, fNetworkNode, fSuccessfullyConnected, and fDisconnect atomic types:

```
-   bool fWhitelisted; // This peer can bypass DoS banning.
-   bool fOneShot;
-   bool fClient;
-   bool fInbound;
-   bool fNetworkNode;
-   bool fSuccessfullyConnected;
-   bool fDisconnect;
+   std::atomic<bool> fWhitelisted; // This peer can bypass DoS banning.
+   std::atomic<bool> fOneShot;
+   std::atomic<bool> fClient;
+   std::atomic<bool> fInbound;
+   std::atomic<bool> fNetworkNode;
+   std::atomic<bool> fSuccessfullyConnected;
+   std::atomic<bool> fDisconnect;
```

In src/net.h:316 make nPingNonceSent, nPingUsecStart, nPingUsecTime, and fPingQueued atomic types:

```
-   uint64_t nPingNonceSent;
+   std::atomic<uint64_t> nPingNonceSent;
    // Time (in usec) the last ping was sent, or 0 if no ping was ever sent.
-   int64_t nPingUsecStart;
+   std::atomic<int64_t> nPingUsecStart;
    // Last measured round-trip time.
-   int64_t nPingUsecTime;
+   std::atomic<int64_t> nPingUsecTime;
    // Whether a ping is requested.
-   bool fPingQueued;
+   std::atomic<bool> fPingQueued;
```

In src/rpcnet.cpp:138 cast stats.nTimeConnected and stats.nTimeOffset to int64_t:

```
-       obj.push_back(Pair("conntime", stats.nTimeConnected));
-       obj.push_back(Pair("timeoffset", stats.nTimeOffset));
+       obj.push_back(Pair("conntime", (int64_t)stats.nTimeConnected));
+       obj.push_back(Pair("timeoffset", (int64_t)stats.nTimeOffset));
```

In src/wallet/db.h:23 make nWalletDBUpdated an atomic type:

```
-extern unsigned int nWalletDBUpdated;
+extern std::atomic<unsigned int> nWalletDBUpdated;
```

The nVersion field of class CNode defined at line in net.h:261 needs to be atomic:

```
    std::atomic<int> nVersion;
```

Unfortunately, this creates a compilation problem with the serialization:

```
In file included from amount.h:9:0,
                 from main.h:13,
                 from main.cpp:6:
serialize.h: In instantiation of void Unserialize(Stream&, T&, long int, int) [wit
➜   h Stream = CDataStream; T = std::atomic<int>]:
streams.h:291:22:   required from CDataStream& CDataStream::operator>>(T&) [with T
➜   = std::atomic<int>]
main.cpp:4293:25:   required from here
serialize.h:572:7: error: struct std::atomic<int> has no member Unserialize
     a.Unserialize(is, (int)nType, nVersion);
```

Ideally this problem would be solved by fixing or replacing the serialization code with code that supports atomics. Alternatively, a critical section can be used although this is an overly heavy weight mechanism to deal with this specific issue.

See Appendix B on page 41 for a sample ThreadSanitizer diagnostic associated with this error.

| | |
|---|---|
| **Vulnerability** | **Uninitialized Objects** |
| **Risk** | Low Impact: Low, Exploitability: Low |
| **Identifier** | NCC-Zcash2016-008 |
| **Client Vulnerability ID** | CID 1352727, CID 1352712, CID 1352709, CID 1352698, CID 1352676, CID 1352664, CID 1352635, CID 1352623, CID 1064010, CID 1064009, etc. |
| **Status** | Reported |
| **Category** | Data Validation |
| **Component** | bitcoin \| zcash \| leveldb |

**Location**

- /src/rpcrawtransaction.cpp:429
- /src/test/coins_tests.cpp:487
- /src/test/coins_tests.cpp:512
- /src/policy/fees.h:108
- /src/chainparams.cpp:83
- /src/wallet/test/wallet_tests.cpp:45
- /src/wallet/wallet.cpp:1970
- /src/miner.cpp:355
- /src/leveldb/table/block.cc:38
- /src/leveldb/port/atomic_pointer.h:112
- /src/leveldb/include/leveldb/cache.h:34
- /src/leveldb/db/version_edit.h:25
- /src/leveldb/db/dbformat.h:75
- /src/leveldb/db/db_impl.cc:48
- /src/leveldb/db/db_impl.cc:81
- /src/zcash/IncrementalMerkleTree.hpp:139
- etc.

**Impact**

Reading an uninitialized object is undefined behavior. Any operation performed on indeterminate values will have an indeterminate value as a result. Library functions will exhibit undefined behavior when used on indeterminate values. Compilers may optimize out expressions that include indeterminate values or treat uninitialized objects as wobbly values that are allowed to change value on each read. This can lead to unpredictable code execution and results that vary based on the specific compiler implementation used.

For example, consider the following code:

```
void f(void) {
  unsigned char x[1]; /* intentionally uninitialized */
  x[0] ^= x[0];
  printf("%d\n", x[0]);
  printf("%d\n", x[0]);
  return;
}
```

In this example, the `unsigned char array x` is intentionally uninitialized, but cannot contain a trap representation because it has a character type. Consequently, the value is both indeterminate and an unspecified value. The bitwise exclusive OR operation, which would produce a zero on an initialized value, will produce an indeterminate result that may or may not be zero. An optimizing compiler has the license to remove this code because it has undefined behavior. The two `printf` calls exhibit undefined behavior, and consequently, might do anything including printing two different values for `x[0]`.

**Description**

Local, automatic variables assume unexpected values if they are read before they are initialized. The C++ Standard,[17] [dcl.init], paragraph 12, states:

> If no initializer is specified for an object, the object is default-initialized. When

[17] ISO/IEC 14882-2014. Programming Languages –- C++, Fourth Edition, 2014.

storage for an object with automatic or dynamic storage duration is obtained, the object has an indeterminate value, and if no initialization is performed for the object, that object retains an indeterminate value until that value is replaced. If an indeterminate value is produced by an evaluation, the behavior is undefined except in the following cases:

1. If an indeterminate value of `unsigned char` type is produced by the evaluation of:
   - the second or third operand of a conditional expression,
   - the right operand of a comma expression,
   - the operand of a cast or conversion to an unsigned narrow character type, or
   - a discarded-value expression, then the result of the operation is an indeterminate value.
2. If an indeterminate value of `unsigned char` type is produced by the evaluation of the right operand of a simple assignment operator whose first operand is an lvalue of unsigned narrow character type, an indeterminate value replaces the value of the object referred to by the left operand.
3. If an indeterminate value of `unsigned char` type is produced by the evaluation of the initialization expression when initializing an object of unsigned narrow character type, that object is initialized to an indeterminate value.

As a result, objects of type `T` with automatic or dynamic storage duration must be explicitly initialized before having their value read as part of an expression unless `T` is a class type or an array thereof or is an unsigned narrow character type. If `T` is an unsigned narrow character type, it may be used to initialize an object of unsigned narrow character type, which results in both objects having an indeterminate value. This technique can be used to implement copy operations such as `std::memcpy()` without triggering undefined behavior.

Additionally, memory dynamically allocated with a new expression is default-initialized when the new-initialized is omitted. Memory allocated by the standard library function `std::calloc()` is zero-initialized. Memory allocated by the standard library function `std::realloc()` assumes the values of the original pointer but may not initialize the full range of memory. Memory allocated by any other means (`std::malloc()`, allocator objects, `operator new()`, and so on) is assumed to be default-initialized.

Objects of static or thread storage duration are zero-initialized before any other initialization takes place and need not be explicitly initialized before having their value read.

**Reproduction Steps** Run the Coverity Prevent scan and filter for issue types: "uninitialized scalar variable", "uninitialized scalar field", and "uninitialized pointer field".

**Recommendation** Thoroughly investigating this is an unbounded problem, while repairing it is fairly straightforward. Provide initial values for all uninitialized objects.

| | |
|---|---|
| **Vulnerability** | **Allocator/Deallocator Mismatch** |
| **Risk** | Low Impact: Low, Exploitability: Low |
| **Identifier** | NCC-Zcash2016-010 |
| **Client Vulnerability ID** | #1214 |
| **Status** | Fixed |
| **Category** | Uncategorized |
| **Component** | bitcoin |
| **Location** | equihash.cpp:203 |
| **Impact** | Passing a pointer value to a deallocation function that was not previously obtained by the matching allocation function results in undefined behavior, which can lead to exploitable vulnerabilities. |

**Description**

The following code at equihash.cpp:203 incorrectly matches operator `new []` with operator `delete`:

```
template<size_t WIDTH>
std::shared_ptr<eh_trunc> TruncatedStepRow<WIDTH>::GetTruncatedIndices(size_t len,
→  size_t lenIndices) const
{
    std::shared_ptr<eh_trunc> p (new eh_trunc[lenIndices]);
    std::copy(hash+len, hash+len+lenIndices, p.get());
    return p;
}
```

See Issue #1214 for a detailed stack trace of the error.

This is a violation of the CERT C++ Secure Coding Rule MEM51-CPP. Properly deallocate dynamically allocated resources.

By default, `shared_ptr` calls `delete` on the managed object when no more references remain to it. However, allocations using the `new []` operator must be matched by a call to `delete []` to free the array.

To correctly use `shared_ptr` with an array it is necessary to supply a custom deleter.

```
template< typename T >
  struct array_deleter {
    void operator ()( T const * p) {
      delete[] p;
    }
};
```

This is accomplished by declaring the `shared_ptr` as follows:

```
std::shared_ptr sp(new int[10], array_deleter() );
```

The `shared_ptr` will now correctly call `delete[]` when destroying the managed object.

In C++11, the `std::default_delete` partial specialization can be used for array types instead of `array_deleter`:

```
std::shared_ptr sp(new int[10], std::default_delete() );
```

A lambda expression can also be used instead of functors:

```
std::shared_ptr sp(new int[10], { delete[] p; } );
```

Unless the managed object must be shared, a `unique_ptr` is better suited for this task, because it has a partial specialization for array types.

```
std::unique_ptr up( new int[10] );  // this will correctly call delete []
```

**Reproduction Steps** Run the full test suite instrumented with AddressSanitizer and search the output for "alloc-dealloc-mismatch":

```
./qa/zcash/full-test-suite.sh
```

**Recommendation** Modify the statement at equihash.cpp:203 to retain the shared pointer and use the `std::default_delete` partial specialization as follows:

```
-    std::shared_ptr<eh_trunc> p (new eh_trunc[lenIndices]);
+    std::shared_ptr<eh_trunc> p (new eh_trunc[lenIndices], std::default_delete<eh
➜   _trunc[]>());
```

Use of `unique_ptr` in this solution breaks some interfaces, so the issue was mitigated using the `std::default_delete` partial specialization solution.

| | |
|---|---|
| **Vulnerability** | **Use of Reserved Identifiers** |
| **Risk** | Low Impact: Low, Exploitability: Low |
| **Identifier** | NCC-Zcash2016-012 |
| **Client Vulnerability ID** | #1272 |
| **Status** | Reported |
| **Category** | Data Validation |
| **Component** | All |
| **Location** | • src/zcash/util.h<br>• src/secp256k1/src/ecmult.h<br>• zcash/Note.hpp<br>• Numerous other locations (see Appendix D on page 55 for a complete list) |
| **Impact** | The use of reserved identifiers in inclusion guards or elsewhere in a program is undefined behavior and can result in unexpected program behavior. |

**Description**

A common practice is to use a macro in a preprocessor conditional that guards against multiple inclusions of a header file. While this is a recommended practice, many programs use reserved names as the header guards. Such a name may clash with reserved names defined by the implementation of the C++ standard template library in its headers or with reserved names implicitly predefined by the compiler even when no C++ standard library header is included.

This is a violation of the CERT C++ Secure Coding Rule DCL51-CPP. Do not declare or define a reserved identifier.

Here are examples of using reserved identifiers in inclusion guards:

In file src/zcash/util.h:

```
#define __ZCASH_UTIL_H
```

In file src/secp256k1/src/ecmult.h:

```
#define _SECP256K1_ECMULT_
```

In file zcash/Note.hpp:

```
#define _ZCNOTE_H_
```

See Appendix D on page 55 for a complete list of reserved identifiers used in Zcash.

The C++ Standard, [reserved.names] [ISO/IEC 14882-2014], specifies the following rules regarding reserved names:

- A translation unit that includes a standard library header shall not `#define` or `#undef` names declared in any standard library header.
- A translation unit shall not `#define` or `#undef` names lexically identical to keywords, to the identifiers listed in Table 3, or to the attribute-tokens described in 7.6.
- Each name that contains a double underscore __ or begins with an underscore followed by an uppercase letter is reserved to the implementation for any use.

- Each name that begins with an underscore is reserved to the implementation for use as a name in the global namespace.
- Each name declared as an object with external linkage in a header is reserved to the implementation to designate that library object with external linkage, both in namespace `std` and in the global namespace.
- Each global function signature declared with external linkage in a header is reserved to the implementation to designate that function signature with external linkage.
- Each name from the Standard C library declared with external linkage is reserved to the implementation for use as a name with extern "C" linkage, both in namespace `std` and in the global namespace.
- Each function signature from the Standard C library declared with external linkage is reserved to the implementation for use as a function signature with both extern "C" and extern "C++" linkage, or as a name of namespace scope in the global namespace. For each type T from the Standard C library, the types `::T` and `std::T` are reserved to the implementation and, when defined, `::T` shall be identical to `std::T`.
- Literal suffix identifiers that do not start with an underscore are reserved for future standardization.

The identifiers and attribute names referred to in the preceding excerpt are `override`, `final`, `alignas`, `carries_dependency`, `deprecated`, and `noreturn`.

No other identifiers are reserved. Declaring or defining an identifier in a context in which it is reserved results in undefined behavior.

**Reproduction Steps** Analyze using LDRA or use a grep command such as:

```
grep -r "#define _"
```

**Recommendation** Ensure that all uses of reserved identifiers in inclusion guards and elsewhere are eliminated from the code.

One solution for inclusion guards is to simply rename the identifiers as to not violate the C++ rules for reserved identifiers.

Another solution is to replace the use of inclusion guards with the once pragma:

```
#pragma once
```

The once pragma is already used extensively in this code base so it should be supported by all currently supported compilers.

There are a variety of arguments for and against each approach, but a single solution (that does not involve reserved identifiers) should be selected and used consistently.

**Vulnerability** **Out-of-bounds Read in Boost `date` Class**

**Risk** Low Impact: Low, Exploitability: Low

**Identifier** NCC-Zcash2016-015

**Client Vulnerability ID** #1459

**Status** Reported

**Category** Data Exposure

**Component** bitcoin

**Location**
- utiltime.cpp:37
- date.hpp:71
- gregorian_calendar.ipp

**Impact** An out-of-bounds read in a date constructor in the Boost libraries is undefined behavior and may result in data leaks.

**Description** AddressSanitizer reports a two byte out-of-bounds read occurs at location date.hpp:71:

```
    date(year_type y, month_type m, day_type d)
      : days_(calendar::day_number(ymd_type(y, m, d)))
    {}
```

`days_` is an `int` type:

```
    explicit date(date_int_type days) : days_(days) {}
    explicit date(date_rep_type days) : days_(days.as_number()) {}
    date_int_type days_;
```

The `ymd_type_` type is defined in gregorian_calendar.ipp and defines a date split into components of `unsigned short` type:

```
  template<typename ymd_type_, typename date_int_type_>
  BOOST_DATE_TIME_INLINE
  unsigned short
  gregorian_calendar_base<ymd_type_,date_int_type_>::day_of_week(const ymd_type& y
→   md) {
    unsigned short a = static_cast<unsigned short>((14-ymd.month)/12);
    unsigned short y = static_cast<unsigned short>(ymd.year - a);
    unsigned short m = static_cast<unsigned short>(ymd.month + 12*a - 2);
    unsigned short d = static_cast<unsigned short>((ymd.day + y + (y/4) - (y/100)
→   + (y/400) + (31*m)/12) % 7);
    //std::cout << year << "-" << month << "-" << day << " is day: " << d << "\n";
    return d;
  }
```

The `day_number()` member function returns a value of type `uint32_t`:

```
boost::gregorian::date d(2014, 10, 18);
uint32_t number = d.day_number();
```

It is unclear where the two byte out-of-bounds read occurs, but it may be a mismatch between the four byte `uint32_t` type and the two byte `unsigned short`.

**Reproduction Steps** Run the full RPC Tests instrumented with AddressSanitizer and search the output for "stack-buffer-overflow":

```
./qa/pull-tester/rpc-tests.sh
```

**Recommendation**

The invoking code in Zcash is at location utiltime.cpp:37:

```
int64_t now = (boost::posix_time::microsec_clock::universal_time() -
       boost::posix_time::ptime(boost::gregorian::date(1970,1,1))).total_microseco
➜   nds();
```

Replacing this code with the following code that uses the C++ `std::chrono` class eliminates the problem:

```
int64_t now = std::chrono::duration_cast<std::chrono::milliseconds>(std::chrono::s
➜   ystem_clock::now().time_since_epoch()).count();
```

| | |
|---|---|
| **Vulnerability** | **Dead Code** |
| **Risk** | **Informational** Impact: Low, Exploitability: None |
| **Identifier** | NCC-Zcash2016-009 |
| **Client Vulnerability ID** | CID 1352634 |
| **Status** | Reported |
| **Category** | Data Validation |
| **Component** | bitcoin |
| **Location** | rpcserver.cpp:1032 |

**Impact** Code that has no effect or is never executed (that is, dead or unreachable code) is typically the result of a coding error and can cause unexpected behavior. Such code is usually optimized out of a program during compilation. However, to improve readability and ensure that logic errors are resolved, dead code should be identified, understood, and eliminated. Similarly, statements or expressions that have no effect should be identified and removed from code.

**Description** The `g_rpcSignals.PostCommand(*pcmd)` at location rpcserver.cpp:1032 is unreachable as the preceding `try` block will either return or throw an exception.

```
1013 json_spirit::Value CRPCTable::execute(const std::string &strMethod, const jso
➜   n_spirit::Array &params) const
1014 {
1015     // Find method
1016     const CRPCCommand *pcmd = tableRPC[strMethod];
1017     if (!pcmd)
1018         throw JSONRPCError(RPC_METHOD_NOT_FOUND, "Method not found");
1019
1020     g_rpcSignals.PreCommand(*pcmd);
1021
1022     try
1023     {
1024         // Execute
1025         return pcmd->actor(params, false);
1026     }
1027     catch (const std::exception& e)
1028     {
1029         throw JSONRPCError(RPC_MISC_ERROR, e.what());
1030     }
1031
1032     g_rpcSignals.PostCommand(*pcmd);
1033 }
```

Consequently, the post commands are never processed. This is a violation of the CERT C Secure Coding Standard recommendation MSC12-C. Detect and remove code that has no effect or is never executed.

**Reproduction Steps** Run the Coverity Prevent scan and filter for issues of type "Structurally dead code".

**Recommendation** Rewrite the code to return after the post commands are processed:

```
json_spirit::Value CRPCTable::execute(const std::string &strMethod, const json_spi
➜   rit::Array &params) const
{
    json_spirit::Value value;
    // Find method
    const CRPCCommand *pcmd = tableRPC[strMethod];
    if (!pcmd)
        throw JSONRPCError(RPC_METHOD_NOT_FOUND, "Method not found");

    g_rpcSignals.PreCommand(*pcmd);

    try
    {
        // Execute
        value = pcmd->actor(params, false);
    }
    catch (const std::exception& e)
    {
        throw JSONRPCError(RPC_MISC_ERROR, e.what());
    }

    g_rpcSignals.PostCommand(*pcmd);
    return value;
}
```

| | |
|---|---|
| **Vulnerability** | **Generous Miners using Alternative Clients may Cause Unintentional Forks** |
| **Risk** | **Informational** Impact: None, Exploitability: None |
| **Identifier** | NCC-Zcash2016-016 |
| **Client Vulnerability ID** | #1460 |
| **Status** | Reported |
| **Category** | Other |
| **Impact** | Alternative client implementations that allow larger founder's rewards may result in unintentional forks. |

**Description**

The Zcash protocol specifies that a founder's reward will be paid out to a specific address up until a certain point in time.

Currently, there are two ways to pay into this founder's address:

1. As the founder's address is a public, normal address, any user can simply create a transaction with the founder's address listed as the output.
2. Each block that is mined must send 20% of the reward to the founder's address.

When checking the validity of a block, the 20% of a reward is checked at location main.cpp:3075:

```
BOOST_FOREACH(const CTxOut& output, block.vtx[0].vout) {
  if (output.scriptPubKey == ParseHex(FOUNDERS_REWARD_SCRIPT)) {
    if (output.nValue == (GetBlockSubsidy(nHeight, consensusParams) / 5)) {
      found = true;
      break;
    }
  }
}
```

The straight equality check may lead to forks if a miner is particularly generous. If an alternative client is put forth that allows more than 20% of the reward being sent to the founder address, forks *may* arise because of the difference in consensus rules.

**Recommendation**

Instead of a strict equality check, perform a greater-than-or-equal-to check against 20% of the block subsidy.

# Appendix A: Vulnerability Field Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

## Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing vulnerabilities. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

### Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a vulnerability poses to the target system or systems. It takes into account the impact of the vulnerability, the difficulty of exploitation, and any other relevant factors.

| | |
|---|---|
| **Critical** | Implies an immediate, easily accessible threat of total compromise. |
| **High** | Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach. |
| **Medium** | A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application. |
| **Low** | Implies a relatively minor threat to the application. |
| **Informational** | No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable vulnerability. |

### Impact

Impact reflects the effects that successful exploitation upon the target system or systems have. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

| | |
|---|---|
| **High** | Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level. |
| **Medium** | Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information. |
| **Low** | Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security. |

### Exploitability

Exploitability reflects the ease with which attackers may exploit a vulnerability. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc., and other impediments to exploitation.

| | |
|---|---|
| **High** | Attackers can unilaterally exploit the vulnerability without special permissions or significant roadblocks. |
| **Medium** | Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the vulnerability. |
| **Low** | Exploitation requires implausible social engineering, a difficult race condition, guessing difficult-to-guess data, or is otherwise unlikely. |

## Category

NCC Group categorizes vulnerabilities based on the security area to which those vulnerabilities belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

| | |
|---|---|
| **Access Controls** | Related to authorization of users, and assessment of rights. |
| **Auditing and Logging** | Related to auditing of actions, or logging of problems. |
| **Authentication** | Related to the identification of users. |
| **Configuration** | Related to security configurations of servers, devices, or software. |
| **Cryptography** | Related to mathematical protections for data. |
| **Data Exposure** | Related to unintended exposure of sensitive information. |
| **Data Validation** | Related to improper reliance on the structure or values of data. |
| **Denial of Service** | Related to causing system failure. |
| **Error Reporting** | Related to the reporting of error conditions in a secure fashion. |
| **Patching** | Related to keeping software up to date. |
| **Session Management** | Related to the identification of authenticated users. |
| **Timing** | Related to race conditions, locking, or order of operations. |

# Appendix B: ThreadSanitizer Diagnostic

This appendix contains a sample ThreadSanitizer diagnostic associated with for the vulnerability finding described by finding NCC-Zcash2016-005 on page 24. Diagnostics listed in this section can also be found on GitHub.[18]

```
WARNING: ThreadSanitizer: data race (pid=28297)
  Read of size 4 at 0x7d640006ba1c by thread T4 (mutexes: write M83, write M137, write M155):
    #0 CNode::copyStats(CNodeStats&) /home/test/zcash/src/net.cpp:508 (zcashd+0x0000005c1fcf)
    #1 CopyNodeStats /home/test/zcash/src/rpcnet.cpp:73 (zcashd+0x0000006efd0e)
    #2 getpeerinfo(std::vector<json_spirit::Value_impl<json_spirit::Config_vector<std::__cxx11::basic_str
→   ing<char, std::char_traits<char>, std::allocator<char> > > >, std::allocator<json_spirit::Value_impl<
→   json_spirit::Config_vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<ch
→   ar> > > > > > const&, bool) /home/test/zcash/src/rpcnet.cpp:121 (zcashd+0x0000006efd0e)
    #3 CRPCTable::execute(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >
→   const&, std::vector<json_spirit::Value_impl<json_spirit::Config_vector<std::__cxx11::basic_string<cha
→   r, std::char_traits<char>, std::allocator<char> > > >, std::allocator<json_spirit::Value_impl<json_sp
→   irit::Config_vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >
→   > > > const&) const /home/test/zcash/src/rpcserver.cpp:1026 (zcashd+0x00000060ac69)
    #4 HTTPReq_JSONRPC /home/test/zcash/src/rpcserver.cpp:950 (zcashd+0x00000060c636)
    #5 ServiceConnection(AcceptedConnection*) /home/test/zcash/src/rpcserver.cpp:998 (zcashd+0x00000060c6
→   36)
    #6 RPCAcceptHandler<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp>
→   > /home/test/zcash/src/rpcserver.cpp:563 (zcashd+0x00000060d7bb)
    #7 void boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boos
→   t::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference_
→   wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<Acce
→   ptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> >::operator()<void (*)(boost::shared_ptr<
→   boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::
→   asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnection>, boost::s
→   ystem::error_code const&), boost::_bi::rrlist1<boost::system::error_code const&> >(boost::_bi::type<v
→   oid>, void (*&)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asi
→   o::socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared
→   _ptr<AcceptedConnection>, boost::system::error_code const&), boost::_bi::rrlist1<boost::system::error
→   _code const&>&, int) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:53
→   1 (zcashd+0x00000061fde7)
    #8 void boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost:
→   :asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::con
→   text&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::error_code const&), boost::_bi::li
→   st5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boos
→   t::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::s
→   sl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boo
→   st::asio::ip::tcp> > >, boost::arg<1> > >::operator()<boost::system::error_code const&>(boost::system
→   ::error_code const&) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:12
→   34 (zcashd+0x00000061fde7)
    #9 boost::asio::detail::binder1<boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::asio::bas
→   ic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> >
→   >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::error_code
→   const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boo
→   st::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference
→   _wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<Acc
→   eptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> > >, boost::system::error_code>::operato
→   r()() /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/bind_handler.hpp:47
→   (zcashd+0x00000061fde7)
```

[18] Issue #1247, Issue #1280, Issue #1281, and Issue #1286.

```
  #10 void boost::asio::asio_handler_invoke<boost::asio::detail::binder1<boost::_bi::bind_t<void, void
→ (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_ac
→ ceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<Accepte
→ dConnection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_pt
→ r<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost
→ ::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>,
→ boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> >
→ >, boost::system::error_code> >(boost::asio::detail::binder1<boost::_bi::bind_t<void, void (*)(boost:
→ :shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_ser
→ vice<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnectio
→ n>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::a
→ sio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip
→ ::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_b
→ i::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> > >, boost
→ ::system::error_code>&, ...) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/han
→ dler_invoke_hook.hpp:69 (zcashd+0x00000061fde7)
  #11 void boost_asio_handler_invoke_helpers::invoke<boost::asio::detail::binder1<boost::_bi::bind_t<vo
→ id, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::
→ socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_pt
→ r<AcceptedConnection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::
→ shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_serv
→ ice<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::val
→ ue<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost
→ ::arg<1> > >, boost::system::error_code>, boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::
→ asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::i
→ p::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::
→ error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_a
→ cceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost
→ ::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::sha
→ red_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> > > >(boost::asio::detail::bi
→ nder1<boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::a
→ sio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::conte
→ xt&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::error_code const&), boost::_bi::list
→ 5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost:
→ :asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl
→ ::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost
→ ::asio::ip::tcp> > >, boost::arg<1> > >, boost::system::error_code>&, boost::_bi::bind_t<void, void (
→ *)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acc
→ eptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<Accepted
→ Connection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr
→ <boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost:
→ :asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>,
→ boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> >
→ >&) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/handler_invoke_helper
→ s.hpp:37 (zcashd+0x00000061fde7)
  #12 boost::asio::detail::reactive_socket_accept_op<boost::asio::basic_socket<boost::asio::ip::tcp, bo
→ ost::asio::stream_socket_service<boost::asio::ip::tcp> >, boost::asio::ip::tcp, boost::_bi::bind_t<vo
→ id, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::
→ socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_pt
→ r<AcceptedConnection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::
→ shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_serv
→ ice<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::val
→ ue<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost
→ ::arg<1> > > >::do_complete(boost::asio::detail::task_io_service*, boost::asio::detail::task_io_servi
→ ce_operation*, boost::system::error_code const&, unsigned long) /home/test/zcash/depends/x86_64-unkno
→ wn-linux-gnu/include/boost/asio/detail/reactive_socket_accept_op.hpp:123 (zcashd+0x00000061fde7)
  #13 boost::asio::detail::task_io_service_operation::complete(boost::asio::detail::task_io_service&, b
→ oost::system::error_code const&, unsigned long) /home/test/zcash/depends/x86_64-unknown-linux-gnu/inc
→ lude/boost/asio/detail/task_io_service_operation.hpp:38 (zcashd+0x00000061ac70)
```

```
    #14 boost::asio::detail::epoll_reactor::descriptor_state::do_complete(boost::asio::detail::task_io_se
→   rvice*, boost::asio::detail::task_io_service_operation*, boost::system::error_code const&, unsigned l
→   ong) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/impl/epoll_reactor.i
→   pp:651 (zcashd+0x00000061ac70)
    #15 boost::asio::detail::task_io_service_operation::complete(boost::asio::detail::task_io_service&, b
→   oost::system::error_code const&, unsigned long) /home/test/zcash/depends/x86_64-unknown-linux-gnu/inc
→   lude/boost/asio/detail/task_io_service_operation.hpp:38 (zcashd+0x00000061e4f3)
    #16 boost::asio::detail::task_io_service::do_run_one(boost::asio::detail::scoped_lock<boost::asio::de
→   tail::posix_mutex>&, boost::asio::detail::task_io_service_thread_info&, boost::system::error_code con
→   st&) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/impl/task_io_service
→   .ipp:372 (zcashd+0x00000061e4f3)
    #17 boost::asio::detail::task_io_service::run(boost::system::error_code&) /home/test/zcash/depends/x8
→   6_64-unknown-linux-gnu/include/boost/asio/detail/impl/task_io_service.ipp:149 (zcashd+0x00000061e4f3)
    #18 boost::asio::io_service::run() /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/as
→   io/impl/io_service.ipp:59 (zcashd+0x00000062475d)
    #19 boost::_mfi::mf0<unsigned long, boost::asio::io_service>::operator()(boost::asio::io_service*) co
→   nst /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/mem_fn_template.hpp:49 (zcas
→   hd+0x000000614b2c)
    #20 unsigned long boost::_bi::list1<boost::_bi::value<boost::asio::io_service*> >::operator()<unsigne
→   d long, boost::_mfi::mf0<unsigned long, boost::asio::io_service>, boost::_bi::list0>(boost::_bi::type
→   <unsigned long>, boost::_mfi::mf0<unsigned long, boost::asio::io_service>&, boost::_bi::list0&, long)
→   /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:249 (zcashd+0x000000614
→   b2c)
    #21 boost::_bi::bind_t<unsigned long, boost::_mfi::mf0<unsigned long, boost::asio::io_service>, boost
→   ::_bi::list1<boost::_bi::value<boost::asio::io_service*> > >::operator()() /home/test/zcash/depends/x
→   86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:1222 (zcashd+0x000000614b2c)
    #22 boost::detail::thread_data<boost::_bi::bind_t<unsigned long, boost::_mfi::mf0<unsigned long, boos
→   t::asio::io_service>, boost::_bi::list1<boost::_bi::value<boost::asio::io_service*> > > >::run() /hom
→   e/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/thread/detail/thread.hpp:116 (zcashd+0x00
→   0000614b2c)
    #23 thread_proxy <null> (zcashd+0x00000094e31a)

  Previous write of size 4 at 0x7d640006ba1c by thread T12 (mutexes: write M801, write M852, write M114):
    #0 void Unserialize<CDataStream>(CDataStream&, int&, int, int) /home/test/zcash/src/serialize.h:221 (
→   zcashd+0x000000543c0c)
    #1 CDataStream& CDataStream::operator>><int>(int&) /home/test/zcash/src/streams.h:291 (zcashd+0x00000
→   0543c0c)
    #2 ProcessMessage /home/test/zcash/src/main.cpp:4294 (zcashd+0x000000543c0c)
    #3 ProcessMessages(CNode*) /home/test/zcash/src/main.cpp:5170 (zcashd+0x00000054eb5f)
    #4 boost::detail::function::function_invoker1<bool (*)(CNode*), bool, CNode*>::invoke(boost::detail::
→   function::function_buffer&, CNode*) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/f
→   unction/function_template.hpp:101 (zcashd+0x0000005513e2)
    #5 boost::function1<bool, CNode*>::operator()(CNode*) const /home/test/zcash/depends/x86_64-unknown-l
→   inux-gnu/include/boost/function/function_template.hpp:771 (zcashd+0x0000005d3cce)
    #6 bool boost::signals2::detail::call_with_tuple_args<bool>::m_invoke<boost::function<bool (CNode*)>,
→   0u, CNode*&>(boost::function<bool (CNode*)>&, boost::signals2::detail::unsigned_meta_array<0u>, std::
→   tuple<CNode*&> const&, boost::disable_if<boost::is_void<boost::function<bool (CNode*)>::result_type>,
→   void>::type*) const /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/signals2/detail/v
→   ariadic_slot_invoker.hpp:98 (zcashd+0x0000005d3cce)
    #7 bool boost::signals2::detail::call_with_tuple_args<bool>::operator()<boost::function<bool (CNode*)
→   >, CNode*&, 1ul>(boost::function<bool (CNode*)>&, std::tuple<CNode*&> const&, mpl_::size_t<1ul>) cons
→   t /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/signals2/detail/variadic_slot_invok
→   er.hpp:90 (zcashd+0x0000005d3cce)
    #8 bool boost::signals2::detail::variadic_slot_invoker<bool, CNode*>::operator()<boost::shared_ptr<bo
→   ost::signals2::detail::connection_body<std::pair<boost::signals2::detail::slot_meta_group, boost::opt
→   ional<int> >, boost::signals2::slot<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2:
→   :mutex> > >(boost::shared_ptr<boost::signals2::detail::connection_body<std::pair<boost::signals2::det
→   ail::slot_meta_group, boost::optional<int> >, boost::signals2::slot<bool (CNode*), boost::function<bo
→   ol (CNode*)> >, boost::signals2::mutex> > const&) const /home/test/zcash/depends/x86_64-unknown-linux
→   -gnu/include/boost/signals2/detail/variadic_slot_invoker.hpp:134 (zcashd+0x0000005d3cce)
```

```
  #9 boost::signals2::detail::slot_call_iterator_t<boost::signals2::detail::variadic_slot_invoker<bool
➔ , CNode*>, std::_List_iterator<boost::shared_ptr<boost::signals2::detail::connection_body<std::pair<b
➔ oost::signals2::detail::slot_meta_group, boost::optional<int> >, boost::signals2::slot<bool (CNode*),
➔ boost::function<bool (CNode*)> >, boost::signals2::mutex> > >, boost::signals2::detail::connection_bo
➔ dy<std::pair<boost::signals2::detail::slot_meta_group, boost::optional<int> >, boost::signals2::slot
➔ <bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2::mutex> >::dereference() const /hom
➔ e/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/signals2/detail/slot_call_iterator.hpp:10
➔ 9 (zcashd+0x0000005d3cce)
  #10 boost::signals2::detail::slot_call_iterator_t<boost::signals2::detail::variadic_slot_invoker<boo
➔ l, CNode*>, std::_List_iterator<boost::shared_ptr<boost::signals2::detail::connection_body<std::pair<
➔ boost::signals2::detail::slot_meta_group, boost::optional<int> >, boost::signals2::slot<bool (CNode*)
➔ , boost::function<bool (CNode*)> >, boost::signals2::mutex> > >, boost::signals2::detail::connection_
➔ body<std::pair<boost::signals2::detail::slot_meta_group, boost::optional<int> >, boost::signals2::slo
➔ t<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2::mutex> >::reference boost::iterat
➔ ors::iterator_core_access::dereference<boost::signals2::detail::slot_call_iterator_t<boost::signals2:
➔ :detail::variadic_slot_invoker<bool, CNode*>, std::_List_iterator<boost::shared_ptr<boost::signals2::
➔ detail::connection_body<std::pair<boost::signals2::detail::slot_meta_group, boost::optional<int> >, b
➔ oost::signals2::slot<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2::mutex> > >, bo
➔ ost::signals2::detail::connection_body<std::pair<boost::signals2::detail::slot_meta_group, boost::opt
➔ ional<int> >, boost::signals2::slot<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2:
➔ :mutex> > >(boost::signals2::detail::slot_call_iterator_t<boost::signals2::detail::variadic_slot_invo
➔ ker<bool, CNode*>, std::_List_iterator<boost::shared_ptr<boost::signals2::detail::connection_body<std
➔ ::pair<boost::signals2::detail::slot_meta_group, boost::optional<int> >, boost::signals2::slot<bool (
➔ CNode*), boost::function<bool (CNode*)> >, boost::signals2::mutex> > >, boost::signals2::detail::conn
➔ ection_body<std::pair<boost::signals2::detail::slot_meta_group, boost::optional<int> >, boost::signal
➔ s2::slot<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2::mutex> > const&) /home/tes
➔ t/zcash/depends/x86_64-unknown-linux-gnu/include/boost/iterator/iterator_facade.hpp:549 (zcashd+0x000
➔ 0005d3cce)
  #11 boost::iterators::detail::iterator_facade_base<boost::signals2::detail::slot_call_iterator_t<boos
➔ t::signals2::detail::variadic_slot_invoker<bool, CNode*>, std::_List_iterator<boost::shared_ptr<boost
➔ ::signals2::detail::connection_body<std::pair<boost::signals2::detail::slot_meta_group, boost::option
➔ al<int> >, boost::signals2::slot<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2::mu
➔ tex> > >, boost::signals2::detail::connection_body<std::pair<boost::signals2::detail::slot_meta_group
➔ , boost::optional<int> >, boost::signals2::slot<bool (CNode*), boost::function<bool (CNode*)> >, boos
➔ t::signals2::mutex> >, bool, boost::iterators::single_pass_traversal_tag, bool&, long, false, false>:
➔ :operator*() const /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/iterator/iterator_
➔ facade.hpp:655 (zcashd+0x0000005d3cce)
  #12 bool CombinerAll::operator()<boost::signals2::detail::slot_call_iterator_t<boost::signals2::detai
➔ l::variadic_slot_invoker<bool, CNode*>, std::_List_iterator<boost::shared_ptr<boost::signals2::detail
➔ ::connection_body<std::pair<boost::signals2::detail::slot_meta_group, boost::optional<int> >, boost::
➔ signals2::slot<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2::mutex> > >, boost::s
➔ ignals2::detail::connection_body<std::pair<boost::signals2::detail::slot_meta_group, boost::optional<
➔ int> >, boost::signals2::slot<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2::mutex
➔ > > >(boost::signals2::detail::slot_call_iterator_t<boost::signals2::detail::variadic_slot_invoker<bo
➔ ol, CNode*>, std::_List_iterator<boost::shared_ptr<boost::signals2::detail::connection_body<std::pair
➔ <boost::signals2::detail::slot_meta_group, boost::optional<int> >, boost::signals2::slot<bool (CNode*
➔ ), boost::function<bool (CNode*)> >, boost::signals2::mutex> > >, boost::signals2::detail::connection
➔ _body<std::pair<boost::signals2::detail::slot_meta_group, boost::optional<int> >, boost::signals2::sl
➔ ot<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2::mutex> >, boost::signals2::detai
➔ l::slot_call_iterator_t<boost::signals2::detail::variadic_slot_invoker<bool, CNode*>, std::_List_iter
➔ ator<boost::shared_ptr<boost::signals2::detail::connection_body<std::pair<boost::signals2::detail::sl
➔ ot_meta_group, boost::optional<int> >, boost::signals2::slot<bool (CNode*), boost::function<bool (CNo
➔ de*)> >, boost::signals2::mutex> > >, boost::signals2::detail::connection_body<std::pair<boost::signa
➔ ls2::detail::slot_meta_group, boost::optional<int> >, boost::signals2::slot<bool (CNode*), boost::fun
➔ ction<bool (CNode*)> >, boost::signals2::mutex> >) const /home/test/zcash/src/net.h:92 (zcashd+0x0000
➔ 005d3cce)
```

```
    #13 bool boost::signals2::detail::combiner_invoker<bool>::operator()<CombinerAll, boost::signals2::de
→   tail::slot_call_iterator_t<boost::signals2::detail::variadic_slot_invoker<bool, CNode*>, std::_List_i
→   terator<boost::shared_ptr<boost::signals2::detail::connection_body<std::pair<boost::signals2::detail:
→   :slot_meta_group, boost::optional<int> >, boost::signals2::slot<bool (CNode*), boost::function<bool (
→   CNode*)> >, boost::signals2::mutex> > >, boost::signals2::detail::connection_body<std::pair<boost::si
→   gnals2::detail::slot_meta_group, boost::optional<int> >, boost::signals2::slot<bool (CNode*), boost::
→   function<bool (CNode*)> >, boost::signals2::mutex> > >(CombinerAll&, boost::signals2::detail::slot_ca
→   ll_iterator_t<boost::signals2::detail::variadic_slot_invoker<bool, CNode*>, std::_List_iterator<boost
→   ::shared_ptr<boost::signals2::detail::connection_body<std::pair<boost::signals2::detail::slot_meta_gr
→   oup, boost::optional<int> >, boost::signals2::slot<bool (CNode*), boost::function<bool (CNode*)> >, b
→   oost::signals2::mutex> > >, boost::signals2::detail::connection_body<std::pair<boost::signals2::detai
→   l::slot_meta_group, boost::optional<int> >, boost::signals2::slot<bool (CNode*), boost::function<bool
→   (CNode*)> >, boost::signals2::mutex> >, boost::signals2::detail::slot_call_iterator_t<boost::signals2
→   ::detail::variadic_slot_invoker<bool, CNode*>, std::_List_iterator<boost::shared_ptr<boost::signals2:
→   :detail::connection_body<std::pair<boost::signals2::detail::slot_meta_group, boost::optional<int> >,
→   boost::signals2::slot<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2::mutex> > >, b
→   oost::signals2::detail::connection_body<std::pair<boost::signals2::detail::slot_meta_group, boost::op
→   tional<int> >, boost::signals2::slot<bool (CNode*), boost::function<bool (CNode*)> >, boost::signals2
→   ::mutex> >) const /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/signals2/detail/res
→   ult_type_wrapper.hpp:53 (zcashd+0x0000005d3cce)
    #14 boost::signals2::detail::signal_impl<bool (CNode*), CombinerAll, int, std::less<int>, boost::func
→   tion<bool (CNode*)>, boost::function<bool (boost::signals2::connection const&, CNode*)>, boost::signa
→   ls2::mutex>::operator()(CNode*) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/signa
→   ls2/detail/signal_template.hpp:247 (zcashd+0x0000005d3cce)
    #15 boost::signals2::signal<bool (CNode*), CombinerAll, int, std::less<int>, boost::function<bool (CN
→   ode*)>, boost::function<bool (boost::signals2::connection const&, CNode*)>, boost::signals2::mutex>::
→   operator()(CNode*) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/signals2/detail/si
→   gnal_template.hpp:723 (zcashd+0x0000005d3cce)
    #16 ThreadMessageHandler() /home/test/zcash/src/net.cpp:1448 (zcashd+0x0000005d3cce)
    #17 void TraceThread<void (*)()>(char const*, void (*)()) /home/test/zcash/src/util.h:217 (zcashd+0x0
→   000005e344f)
    #18 void boost::_bi::list2<boost::_bi::value<char const*>, boost::_bi::value<void (*)()> >::operator(
→   )<void (*)(char const*, void (*)()), boost::_bi::list0>(boost::_bi::type<void>, void (*&)(char const*
→   , void (*)()), boost::_bi::list0&, int) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boo
→   st/bind/bind.hpp:319 (zcashd+0x0000005d55d2)
    #19 boost::_bi::bind_t<void, void (*)(char const*, void (*)()), boost::_bi::list2<boost::_bi::value<c
→   har const*>, boost::_bi::value<void (*)()> > >::operator()() /home/test/zcash/depends/x86_64-unknown-
→   linux-gnu/include/boost/bind/bind.hpp:1222 (zcashd+0x0000005d55d2)
    #20 boost::detail::thread_data<boost::_bi::bind_t<void, void (*)(char const*, void (*)()), boost::_bi
→   ::list2<boost::_bi::value<char const*>, boost::_bi::value<void (*)()> > > >::run() /home/test/zcash/d
→   epends/x86_64-unknown-linux-gnu/include/boost/thread/detail/thread.hpp:116 (zcashd+0x0000005d55d2)
    #21 thread_proxy <null> (zcashd+0x00000094e31a)

  Location is heap block of size 1136 at 0x7d640006b800 allocated by thread T4:
    #0 operator new(unsigned long) <null> (zcashd+0x000000464983)
    #1 ConnectNode(CAddress, char const*) /home/test/zcash/src/net.cpp:392 (zcashd+0x0000005cca24)
    #2 OpenNetworkConnection(CAddress const&, CSemaphoreGrant*, char const*, bool) /home/test/zcash/src/n
→   et.cpp:1399 (zcashd+0x0000005ccf91)
    #3 addnode(std::vector<json_spirit::Value_impl<json_spirit::Config_vector<std::__cxx11::basic_string
→   <char, std::char_traits<char>, std::allocator<char> > > >, std::allocator<json_spirit::Value_impl<jso
→   n_spirit::Config_vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>
→   > > > > > const&, bool) /home/test/zcash/src/rpcnet.cpp:192 (zcashd+0x0000006f6357)
    #4 CRPCTable::execute(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >
→   const&, std::vector<json_spirit::Value_impl<json_spirit::Config_vector<std::__cxx11::basic_string<cha
→   r, std::char_traits<char>, std::allocator<char> > > >, std::allocator<json_spirit::Value_impl<json_sp
→   irit::Config_vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >
→   > > > const&) const /home/test/zcash/src/rpcserver.cpp:1026 (zcashd+0x00000060ac69)
    #5 HTTPReq_JSONRPC /home/test/zcash/src/rpcserver.cpp:950 (zcashd+0x00000060c636)
    #6 ServiceConnection(AcceptedConnection*) /home/test/zcash/src/rpcserver.cpp:998 (zcashd+0x00000060c6
→   36)
    #7 RPCAcceptHandler<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp>
→   > /home/test/zcash/src/rpcserver.cpp:563 (zcashd+0x00000060d7bb)
```

```
  #8 void boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boos
→ t::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference_
→ wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<Acce
→ ptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> >::operator()<void (*)(boost::shared_ptr<
→ boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::
→ asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnection>, boost::s
→ ystem::error_code const&), boost::_bi::rrlist1<boost::system::error_code const&> >(boost::_bi::type<v
→ oid>, void (*&)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asi
→ o::socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared
→ _ptr<AcceptedConnection>, boost::system::error_code const&), boost::_bi::rrlist1<boost::system::error
→ _code const&>&, int) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:53
→ 1 (zcashd+0x00000061fde7)
  #9 void boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost:
→ :asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::con
→ text&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::error_code const&), boost::_bi::li
→ st5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boos
→ t::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::s
→ sl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boo
→ st::asio::ip::tcp> > >, boost::arg<1> > >::operator()<boost::system::error_code const&>(boost::system
→ ::error_code const&) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:12
→ 34 (zcashd+0x00000061fde7)
  #10 boost::asio::detail::binder1<boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::asio::bas
→ ic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> >
→ >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::error_code
→ const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boo
→ st::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference
→ _wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<Acc
→ eptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> > >, boost::system::error_code>::operato
→ r()() /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/bind_handler.hpp:47
→ (zcashd+0x00000061fde7)
  #11 void boost::asio::asio_handler_invoke<boost::asio::detail::binder1<boost::_bi::bind_t<void, void
→ (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_ac
→ ceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<Accepte
→ dConnection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_pt
→ r<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost
→ ::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>,
→ boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> >
→ >, boost::system::error_code> >(boost::asio::detail::binder1<boost::_bi::bind_t<void, void (*)(boost:
→ :shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_ser
→ vice<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnectio
→ n>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::a
→ sio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip
→ ::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_b
→ i::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> > >, boost
→ ::system::error_code>&, ...) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/han
→ dler_invoke_hook.hpp:69 (zcashd+0x00000061fde7)
```

```
 #12 void boost_asio_handler_invoke_helpers::invoke<boost::asio::detail::binder1<boost::_bi::bind_t<vo
➜ id, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::
➜ socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_pt
➜ r<AcceptedConnection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::
➜ shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_serv
➜ ice<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::val
➜ ue<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost
➜ ::arg<1> > >, boost::system::error_code>, boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::
➜ asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::i
➜ p::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::
➜ error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_a
➜ cceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost
➜ ::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::sha
➜ red_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> > > >(boost::asio::detail::bi
➜ nder1<boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::a
➜ sio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::conte
➜ xt&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::error_code const&), boost::_bi::list
➜ 5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost:
➜ :asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl
➜ ::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost
➜ ::asio::ip::tcp> > >, boost::arg<1> > >, boost::system::error_code>&, boost::_bi::bind_t<void, void (
➜ *)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acc
➜ eptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<Accepted
➜ Connection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr
➜ <boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost:
➜ :asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>,
➜ boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> >
➜ >&) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/handler_invoke_helper
➜ s.hpp:37 (zcashd+0x00000061fde7)
 #13 boost::asio::detail::reactive_socket_accept_op<boost::asio::basic_socket<boost::asio::ip::tcp, bo
➜ ost::asio::stream_socket_service<boost::asio::ip::tcp> >, boost::asio::ip::tcp, boost::_bi::bind_t<vo
➜ id, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::
➜ socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_pt
➜ r<AcceptedConnection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::
➜ shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_serv
➜ ice<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::val
➜ ue<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost
➜ ::arg<1> > > >::do_complete(boost::asio::detail::task_io_service*, boost::asio::detail::task_io_servi
➜ ce_operation*, boost::system::error_code const&, unsigned long) /home/test/zcash/depends/x86_64-unkno
➜ wn-linux-gnu/include/boost/asio/detail/reactive_socket_accept_op.hpp:123 (zcashd+0x00000061fde7)
 #14 boost::asio::detail::task_io_service_operation::complete(boost::asio::detail::task_io_service&, b
➜ oost::system::error_code const&, unsigned long) /home/test/zcash/depends/x86_64-unknown-linux-gnu/inc
➜ lude/boost/asio/detail/task_io_service_operation.hpp:38 (zcashd+0x00000061ac70)
 #15 boost::asio::detail::epoll_reactor::descriptor_state::do_complete(boost::asio::detail::task_io_se
➜ rvice*, boost::asio::detail::task_io_service_operation*, boost::system::error_code const&, unsigned l
➜ ong) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/impl/epoll_reactor.i
➜ pp:651 (zcashd+0x00000061ac70)
 #16 boost::asio::detail::task_io_service_operation::complete(boost::asio::detail::task_io_service&, b
➜ oost::system::error_code const&, unsigned long) /home/test/zcash/depends/x86_64-unknown-linux-gnu/inc
➜ lude/boost/asio/detail/task_io_service_operation.hpp:38 (zcashd+0x00000061e4f3)
 #17 boost::asio::detail::task_io_service::do_run_one(boost::asio::detail::scoped_lock<boost::asio::de
➜ tail::posix_mutex>&, boost::asio::detail::task_io_service_thread_info&, boost::system::error_code con
➜ st&) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/impl/task_io_service
➜ .ipp:372 (zcashd+0x00000061e4f3)
 #18 boost::asio::detail::task_io_service::run(boost::system::error_code&) /home/test/zcash/depends/x8
➜ 6_64-unknown-linux-gnu/include/boost/asio/detail/impl/task_io_service.ipp:149 (zcashd+0x00000061e4f3)
 #19 boost::asio::io_service::run() /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/as
➜ io/impl/io_service.ipp:59 (zcashd+0x00000062475d)
 #20 boost::_mfi::mf0<unsigned long, boost::asio::io_service>::operator()(boost::asio::io_service*) co
➜ nst /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/mem_fn_template.hpp:49 (zcas
➜ hd+0x000000614b2c)
```

```
    #21 unsigned long boost::_bi::list1<boost::_bi::value<boost::asio::io_service*> >::operator()<unsigne
➜   d long, boost::_mfi::mf0<unsigned long, boost::asio::io_service>, boost::_bi::list0>(boost::_bi::type
➜   <unsigned long>, boost::_mfi::mf0<unsigned long, boost::asio::io_service>&, boost::_bi::list0&, long)
➜   /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:249 (zcashd+0x000000614
➜   b2c)
    #22 boost::_bi::bind_t<unsigned long, boost::_mfi::mf0<unsigned long, boost::asio::io_service>, boost
➜   ::_bi::list1<boost::_bi::value<boost::asio::io_service*> > >::operator()() /home/test/zcash/depends/x
➜   86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:1222 (zcashd+0x000000614b2c)
    #23 boost::detail::thread_data<boost::_bi::bind_t<unsigned long, boost::_mfi::mf0<unsigned long, boos
➜   t::asio::io_service>, boost::_bi::list1<boost::_bi::value<boost::asio::io_service*> > > >::run() /hom
➜   e/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/thread/detail/thread.hpp:116 (zcashd+0x00
➜   0000614b2c)
    #24 thread_proxy <null> (zcashd+0x00000094e31a)

  Mutex M83 (0x00000200ca20) created at:
    #0 pthread_mutex_init <null> (zcashd+0x0000004236ae)
    #1 boost::recursive_mutex::recursive_mutex() /home/test/zcash/depends/x86_64-unknown-linux-gnu/includ
➜   e/boost/thread/pthread/recursive_mutex.hpp:79 (zcashd+0x0000004dd1fa)
    #2 AnnotatedMixin<boost::recursive_mutex>::AnnotatedMixin() /home/test/zcash/src/sync.h:55 (zcashd+0x
➜   00000040f95e)
    #3 __static_initialization_and_destruction_0 /home/test/zcash/src/main.cpp:49 (zcashd+0x00000040f95e)
    #4 _GLOBAL__sub_I_cs_main /home/test/zcash/src/main.cpp:5491 (zcashd+0x00000040f95e)
    #5 __libc_csu_init <null> (zcashd+0x000000d3b4ac)

  Mutex M137 (0x00000200dce0) created at:
    #0 pthread_mutex_init <null> (zcashd+0x0000004236ae)
    #1 boost::recursive_mutex::recursive_mutex() /home/test/zcash/depends/x86_64-unknown-linux-gnu/includ
➜   e/boost/thread/pthread/recursive_mutex.hpp:79 (zcashd+0x0000004dd1fa)
    #2 AnnotatedMixin<boost::recursive_mutex>::AnnotatedMixin() /home/test/zcash/src/sync.h:55 (zcashd+0x
➜   000000410b49)
    #3 __static_initialization_and_destruction_0 /home/test/zcash/src/net.cpp:87 (zcashd+0x000000410b49)
    #4 _GLOBAL__sub_I_fDiscover /home/test/zcash/src/net.cpp:2104 (zcashd+0x000000410b49)
    #5 __libc_csu_init <null> (zcashd+0x000000d3b4ac)

  Mutex M155 (0x00000200d9e0) created at:
    #0 pthread_mutex_init <null> (zcashd+0x0000004236ae)
    #1 boost::recursive_mutex::recursive_mutex() /home/test/zcash/depends/x86_64-unknown-linux-gnu/includ
➜   e/boost/thread/pthread/recursive_mutex.hpp:79 (zcashd+0x0000004dd1fa)
    #2 AnnotatedMixin<boost::recursive_mutex>::AnnotatedMixin() /home/test/zcash/src/sync.h:55 (zcashd+0x
➜   0000004111d1)
    #3 __static_initialization_and_destruction_0 /home/test/zcash/src/net.cpp:105 (zcashd+0x0000004111d1)
    #4 _GLOBAL__sub_I_fDiscover /home/test/zcash/src/net.cpp:2104 (zcashd+0x0000004111d1)
    #5 __libc_csu_init <null> (zcashd+0x000000d3b4ac)

  Mutex M801 (0x7f7fcdbcacf0) created at:
    #0 pthread_mutex_init <null> (zcashd+0x0000004236ae)
    #1 boost::mutex::mutex() /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/thread/pthre
➜   ad/mutex.hpp:101 (zcashd+0x0000005dde1b)
    #2 ThreadMessageHandler() /home/test/zcash/src/net.cpp:1416 (zcashd+0x0000005d2c7d)
    #3 void TraceThread<void (*)()>(char const*, void (*)()) /home/test/zcash/src/util.h:217 (zcashd+0x00
➜   00005e344f)
    #4 void boost::_bi::list2<boost::_bi::value<char const*>, boost::_bi::value<void (*)()> >::operator(
➜   )<void (*)(char const*, void (*)()), boost::_bi::list0>(boost::_bi::type<void>, void (*&)(char const*
➜   , void (*)()), boost::_bi::list0&, int) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boo
➜   st/bind/bind.hpp:319 (zcashd+0x0000005d55d2)
    #5 boost::_bi::bind_t<void, void (*)(char const*, void (*)()), boost::_bi::list2<boost::_bi::value<ch
➜   ar const*>, boost::_bi::value<void (*)()> > >::operator()() /home/test/zcash/depends/x86_64-unknown-l
➜   inux-gnu/include/boost/bind/bind.hpp:1222 (zcashd+0x0000005d55d2)
    #6 boost::detail::thread_data<boost::_bi::bind_t<void, void (*)(char const*, void (*)()), boost::_bi:
➜   :list2<boost::_bi::value<char const*>, boost::_bi::value<void (*)()> > > >::run() /home/test/zcash/de
➜   pends/x86_64-unknown-linux-gnu/include/boost/thread/detail/thread.hpp:116 (zcashd+0x0000005d55d2)
    #7 thread_proxy <null> (zcashd+0x00000094e31a)
```

```
Mutex M852 (0x7d640006b968) created at:
  #0 pthread_mutex_init <null> (zcashd+0x0000004236ae)
  #1 boost::recursive_mutex::recursive_mutex() /home/test/zcash/depends/x86_64-unknown-linux-gnu/includ
→ e/boost/thread/pthread/recursive_mutex.hpp:79 (zcashd+0x0000004dd1fa)
  #2 AnnotatedMixin<boost::recursive_mutex>::AnnotatedMixin() /home/test/zcash/src/sync.h:55 (zcashd+0x
→ 0000005cb822)
  #3 CNode::CNode(unsigned int, CAddress, std::__cxx11::basic_string<char, std::char_traits<char>, std:
→ :allocator<char> >, bool) /home/test/zcash/src/net.cpp:1954 (zcashd+0x0000005cb822)
  #4 ConnectNode(CAddress, char const*) /home/test/zcash/src/net.cpp:392 (zcashd+0x0000005cca68)
  #5 OpenNetworkConnection(CAddress const&, CSemaphoreGrant*, char const*, bool) /home/test/zcash/src/n
→ et.cpp:1399 (zcashd+0x0000005ccf91)
  #6 addnode(std::vector<json_spirit::Value_impl<json_spirit::Config_vector<std::__cxx11::basic_string
→ <char, std::char_traits<char>, std::allocator<char> > > >, std::allocator<json_spirit::Value_impl<jso
→ n_spirit::Config_vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>
→ > > > > > const&, bool) /home/test/zcash/src/rpcnet.cpp:192 (zcashd+0x0000006f6357)
  #7 CRPCTable::execute(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >
→ const&, std::vector<json_spirit::Value_impl<json_spirit::Config_vector<std::__cxx11::basic_string<cha
→ r, std::char_traits<char>, std::allocator<char> > > >, std::allocator<json_spirit::Value_impl<json_sp
→ irit::Config_vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >
→ > > > const&) const /home/test/zcash/src/rpcserver.cpp:1026 (zcashd+0x00000060ac69)
  #8 HTTPReq_JSONRPC /home/test/zcash/src/rpcserver.cpp:950 (zcashd+0x00000060c636)
  #9 ServiceConnection(AcceptedConnection*) /home/test/zcash/src/rpcserver.cpp:998 (zcashd+0x00000060c6
→ 36)
  #10 RPCAcceptHandler<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp>
→ > /home/test/zcash/src/rpcserver.cpp:563 (zcashd+0x00000060d7bb)
  #11 void boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boo
→ st::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference
→ _wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<Acc
→ eptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> >::operator()<void (*)(boost::shared_ptr
→ <boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost:
→ :asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnection>, boost::
→ system::error_code const&), boost::_bi::rrlist1<boost::system::error_code const&> >(boost::_bi::type<
→ void>, void (*&)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::as
→ io::socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::share
→ d_ptr<AcceptedConnection>, boost::system::error_code const&), boost::_bi::rrlist1<boost::system::erro
→ r_code const&>&, int) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:5
→ 31 (zcashd+0x00000061fde7)
  #12 void boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost
→ ::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::co
→ ntext&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::error_code const&), boost::_bi::l
→ ist5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boo
→ st::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::
→ ssl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<bo
→ ost::asio::ip::tcp> > >, boost::arg<1> > >::operator()<boost::system::error_code const&>(boost::syste
→ m::error_code const&) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:1
→ 234 (zcashd+0x00000061fde7)
  #13 boost::asio::detail::binder1<boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::asio::bas
→ ic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> >
→ >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::error_code
→ const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boo
→ st::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference
→ _wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<Acc
→ eptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> > >, boost::system::error_code>::operato
→ r()() /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/bind_handler.hpp:47
→ (zcashd+0x00000061fde7)
```

```
    #14 void boost::asio::asio_handler_invoke<boost::asio::detail::binder1<boost::_bi::bind_t<void, void
→   (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_ac
→   ceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<Accepte
→   dConnection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_pt
→   r<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost
→   ::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>,
→   boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> >
→   >, boost::system::error_code> >(boost::asio::detail::binder1<boost::_bi::bind_t<void, void (*)(boost:
→   :shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_ser
→   vice<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnectio
→   n>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::a
→   sio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip
→   ::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_b
→   i::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> > >, boost
→   ::system::error_code>&, ...) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/han
→   dler_invoke_hook.hpp:69 (zcashd+0x00000061fde7)
    #15 void boost_asio_handler_invoke_helpers::invoke<boost::asio::detail::binder1<boost::_bi::bind_t<vo
→   id, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::
→   socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_pt
→   r<AcceptedConnection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::
→   shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_serv
→   ice<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::val
→   ue<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost
→   ::arg<1> > >, boost::system::error_code>, boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::
→   asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::i
→   p::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::
→   error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_a
→   cceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost
→   ::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>, boost::_bi::value<boost::sha
→   red_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> > > >(boost::asio::detail::bi
→   nder1<boost::_bi::bind_t<void, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::a
→   sio::ip::tcp, boost::asio::socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::conte
→   xt&, bool, boost::shared_ptr<AcceptedConnection>, boost::system::error_code const&), boost::_bi::list
→   5<boost::_bi::value<boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost:
→   :asio::socket_acceptor_service<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl
→   ::context>, boost::_bi::value<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost
→   ::asio::ip::tcp> > >, boost::arg<1> > >, boost::system::error_code>&, boost::_bi::bind_t<void, void (
→   *)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acc
→   eptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_ptr<Accepted
→   Connection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::shared_ptr
→   <boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_service<boost:
→   :asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::value<bool>,
→   boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost::arg<1> >
→   >&) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/handler_invoke_helper
→   s.hpp:37 (zcashd+0x00000061fde7)
    #16 boost::asio::detail::reactive_socket_accept_op<boost::asio::basic_socket<boost::asio::ip::tcp, bo
→   ost::asio::stream_socket_service<boost::asio::ip::tcp> >, boost::asio::ip::tcp, boost::_bi::bind_t<vo
→   id, void (*)(boost::shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::
→   socket_acceptor_service<boost::asio::ip::tcp> > >, boost::asio::ssl::context&, bool, boost::shared_pt
→   r<AcceptedConnection>, boost::system::error_code const&), boost::_bi::list5<boost::_bi::value<boost::
→   shared_ptr<boost::asio::basic_socket_acceptor<boost::asio::ip::tcp, boost::asio::socket_acceptor_serv
→   ice<boost::asio::ip::tcp> > > >, boost::reference_wrapper<boost::asio::ssl::context>, boost::_bi::val
→   ue<bool>, boost::_bi::value<boost::shared_ptr<AcceptedConnectionImpl<boost::asio::ip::tcp> > >, boost
→   ::arg<1> > > >::do_complete(boost::asio::detail::task_io_service*, boost::asio::detail::task_io_servi
→   ce_operation*, boost::system::error_code const&, unsigned long) /home/test/zcash/depends/x86_64-unkno
→   wn-linux-gnu/include/boost/asio/detail/reactive_socket_accept_op.hpp:123 (zcashd+0x00000061fde7)
    #17 boost::asio::detail::task_io_service_operation::complete(boost::asio::detail::task_io_service&, b
→   oost::system::error_code const&, unsigned long) /home/test/zcash/depends/x86_64-unknown-linux-gnu/inc
→   lude/boost/asio/detail/task_io_service_operation.hpp:38 (zcashd+0x00000061ac70)
```

```
    #18 boost::asio::detail::epoll_reactor::descriptor_state::do_complete(boost::asio::detail::task_io_se
➔   rvice*, boost::asio::detail::task_io_service_operation*, boost::system::error_code const&, unsigned l
➔   ong) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/impl/epoll_reactor.i
➔   pp:651 (zcashd+0x00000061ac70)
    #19 boost::asio::detail::task_io_service_operation::complete(boost::asio::detail::task_io_service&, b
➔   oost::system::error_code const&, unsigned long) /home/test/zcash/depends/x86_64-unknown-linux-gnu/inc
➔   lude/boost/asio/detail/task_io_service_operation.hpp:38 (zcashd+0x00000061e4f3)
    #20 boost::asio::detail::task_io_service::do_run_one(boost::asio::detail::scoped_lock<boost::asio::de
➔   tail::posix_mutex>&, boost::asio::detail::task_io_service_thread_info&, boost::system::error_code con
➔   st&) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/asio/detail/impl/task_io_service
➔   .ipp:372 (zcashd+0x00000061e4f3)
    #21 boost::asio::detail::task_io_service::run(boost::system::error_code&) /home/test/zcash/depends/x8
➔   6_64-unknown-linux-gnu/include/boost/asio/detail/impl/task_io_service.ipp:149 (zcashd+0x00000061e4f3)
    #22 boost::asio::io_service::run() /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/as
➔   io/impl/io_service.ipp:59 (zcashd+0x00000062475d)
    #23 boost::_mfi::mf0<unsigned long, boost::asio::io_service>::operator()(boost::asio::io_service*) co
➔   nst /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/mem_fn_template.hpp:49 (zcas
➔   hd+0x000000614b2c)
    #24 unsigned long boost::_bi::list1<boost::_bi::value<boost::asio::io_service*> >::operator()<unsigne
➔   d long, boost::_mfi::mf0<unsigned long, boost::asio::io_service>, boost::_bi::list0>(boost::_bi::type
➔   <unsigned long>, boost::_mfi::mf0<unsigned long, boost::asio::io_service>&, boost::_bi::list0&, long)
➔   /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:249 (zcashd+0x000000614
➔   b2c)
    #25 boost::_bi::bind_t<unsigned long, boost::_mfi::mf0<unsigned long, boost::asio::io_service>, boost
➔   ::_bi::list1<boost::_bi::value<boost::asio::io_service*> > >::operator()() /home/test/zcash/depends/x
➔   86_64-unknown-linux-gnu/include/boost/bind/bind.hpp:1222 (zcashd+0x000000614b2c)
    #26 boost::detail::thread_data<boost::_bi::bind_t<unsigned long, boost::_mfi::mf0<unsigned long, boos
➔   t::asio::io_service>, boost::_bi::list1<boost::_bi::value<boost::asio::io_service*> > > >::run() /hom
➔   e/test/zcash/depends/x86_64-unknown-linux-gnu/include/boost/thread/detail/thread.hpp:116 (zcashd+0x00
➔   0000614b2c)
    #27 thread_proxy <null> (zcashd+0x00000094e31a)

  Mutex M114 (0x00000200c6e0) created at:
    #0 pthread_mutex_init <null> (zcashd+0x0000004236ae)
    #1 boost::recursive_mutex::recursive_mutex() /home/test/zcash/depends/x86_64-unknown-linux-gnu/includ
➔   e/boost/thread/pthread/recursive_mutex.hpp:79 (zcashd+0x0000004dd1fa)
    #2 AnnotatedMixin<boost::recursive_mutex>::AnnotatedMixin() /home/test/zcash/src/sync.h:55 (zcashd+0x
➔   00000041017d)
    #3 __static_initialization_and_destruction_0 /home/test/zcash/src/main.cpp:4265 (zcashd+0x00000041017
➔   d)
    #4 _GLOBAL__sub_I_cs_main /home/test/zcash/src/main.cpp:5491 (zcashd+0x00000041017d)
    #5 __libc_csu_init <null> (zcashd+0x000000d3b4ac)

  Thread T4 (tid=28309, running) created by main thread at:
    #0 pthread_create <null> (zcashd+0x000000422e80)
    #1 boost::thread::start_thread_noexcept() <null> (zcashd+0x00000094cdc8)
    #2 StartRPCThreads() /home/test/zcash/src/rpcserver.cpp:731 (zcashd+0x0000006109d9)
    #3 AppInit2(boost::thread_group&, CScheduler&) /home/test/zcash/src/init.cpp:988 (zcashd+0x0000004cad
➔   d4)
    #4 AppInit(int, char**) /home/test/zcash/src/bitcoind.cpp:147 (zcashd+0x0000004a01f1)
    #5 main /home/test/zcash/src/bitcoind.cpp:176 (zcashd+0x00000040eea8)

  Thread T12 'bitcoin-msghand' (tid=28317, running) created by main thread at:
    #0 pthread_create <null> (zcashd+0x000000422e80)
    #1 boost::thread::start_thread_noexcept() <null> (zcashd+0x00000094cdc8)
    #2 boost::thread* boost::thread_group::create_thread<boost::_bi::bind_t<void, void (*)(char const*, v
➔   oid (*)()), boost::_bi::list2<boost::_bi::value<char const*>, boost::_bi::value<void (*)()> > >(boo
➔   st::_bi::bind_t<void, void (*)(char const*, void (*)()), boost::_bi::list2<boost::_bi::value<char con
➔   st*>, boost::_bi::value<void (*)()> > >) /home/test/zcash/depends/x86_64-unknown-linux-gnu/include/bo
➔   ost/thread/detail/thread_group.hpp:79 (zcashd+0x0000005dee97)
    #3 StartNode(boost::thread_group&, CScheduler&) /home/test/zcash/src/net.cpp:1683 (zcashd+0x0000005d2
➔   705)
```

```
    #4 AppInit2(boost::thread_group&, CScheduler&) /home/test/zcash/src/init.cpp:1469 (zcashd+0x0000004cf
➜   05b)
    #5 AppInit(int, char**) /home/test/zcash/src/bitcoind.cpp:147 (zcashd+0x0000004a01f1)
    #6 main /home/test/zcash/src/bitcoind.cpp:176 (zcashd+0x00000040eea8)

SUMMARY: ThreadSanitizer: data race /home/test/zcash/src/net.cpp:508 in CNode::copyStats(CNodeStats&)
```

## Appendix C: AddressSanitizer Stack-based Buffer Overflow

This appendix contains the complete diagnostic issued by AddressSanitizer for the vulnerability finding described by finding NCC-Zcash2016-014 on page 17.

```
  #0 0x556e92f92356 in tinyformat::detail::FormatIterator::FormatIterator(std::ostream&, char const*) tin
➜  yformat.h:465
    #0 0x55c01b453356 in tinyformat::detail::FormatIterator::FormatIterator(std::ostream&, char const*) t
➜  inyformat.h:465
    #0 0x560804de9356 in tinyformat::detail::FormatIterator::FormatIterator(std::ostream&, char const*) t
➜  inyformat.h:465
    #0 0x559cc96d9356 in tinyformat::detail::FormatIterator::FormatIterator(std::ostream&, char const*) t
➜  inyformat.h:465
    #1 0x556e92fab6eb in void tinyformat::format<char const*>(std::ostream&, char const*, char const* con
➜  st&) /home/test/zcash/src/tinyformat.h:901
    #2 0x556e92fab6eb in std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >
➜  tinyformat::format<char const*>(char const*, char const* const&) /home/test/zcash/src/tinyformat.h:90
➜  9
    #3 0x556e9310a801 in LogPrint<char const*> /home/test/zcash/src/util.h:91
    #4 0x556e9310aa6d in void TraceThread<void (*)()>(char const*, void (*)()) /home/test/zcash/src/util.
➜  h:222
    #5 0x556e9357d78a in thread_proxy (/home/test/zcash/src/zcashd+0x84c78a)
    #6 0x7f8433e9d6f9 in start_thread (/lib/x86_64-linux-gnu/libpthread.so.0+0x76f9)
    #7 0x7f84339bcb5c in clone (/lib/x86_64-linux-gnu/libc.so.6+0x106b5c)

Address 0x7f842b49da40 is located in stack of thread T11 (bitcoin-opencon) at offset 32 in frame
    #0 0x556e92fab63f in std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >
➜  tinyformat::format<char const*>(char const*, char const* const&) /home/test/zcash/src/tinyformat.h:90
➜  6

  This frame has 2 object(s):
    [32, 88) fmtIter <== Memory access at offset 32 is inside this variable
    [128, 504) oss

Thread T11 (bitcoin-opencon) created by T0 here:
    #0 0x556e92e7fdf8 in __interceptor_pthread_create (/home/test/zcash/src/zcashd+0x14edf8)
    #1 0x556e9357c238 in boost::thread::start_thread_noexcept() (/home/test/zcash/src/zcashd+0x84b238)

SUMMARY: AddressSanitizer: stack-buffer-overflow tinyformat.h:465 in tinyformat::detail::FormatIterator::
➜  FormatIterator(std::ostream&, char const*)
Shadow bytes around the buggy address:
  0x0ff10568baf0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff10568bb00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff10568bb10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff10568bb20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff10568bb30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f1 f1
=>0x0ff10568bb40: f1 f1 00 00 f1 f1 f1 f1[f2]f2 00 00 00 00 00 f4
  0x0ff10568bb50: f2 f2 f2 f2 00 00 00 00 f4 f4 f2 f2 f2 f2 00 00
  0x0ff10568bb60: f4 f4 f2 f2 f2 f2 00 00 f4 f4 f2 f2 f2 f2 00 00
  0x0ff10568bb70: 00 00 00 00 f2 f2 00 00 00 00 00 00 f2 f2 00 00
  0x0ff10568bb80: 00 00 00 f4 f2 f2 00 00 00 00 00 f4 f4 f4 f2 f2
  0x0ff10568bb90: f2 f2 f1 f1 f1 f1 00 00 00 00 f3 f3 f3 f3 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Heap right redzone:      fb
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack partial redzone:   f4
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
```

# Appendix D: Use of Reserved Identifiers

This appendix contains a complete list of locations for the vulnerability finding described by finding NCC-Zcash2016-012 on page 32.

In file src/zcash/util.h:

```
#define __ZCASH_UTIL_H
```

In file src/secp256k1/src/ecmult.h:

```
#define _SECP256K1_ECMULT_
```

In file secp256k1/src/num_impl.h:

```
#define _SECP256K1_NUM_IMPL_H_
```

In file secp256k1/src/field_5x52_asm_impl.h:

```
#define _SECP256K1_FIELD_INNER5X52_IMPL_H_
```

In file secp256k1/src/ecmult_impl.h:

```
#define _SECP256K1_ECMULT_IMPL_H_
```

In file secp256k1/src/scalar_4x64_impl.h:

```
#define _SECP256K1_SCALAR_REPR_IMPL_H_
```

In file secp256k1/src/hash_impl.h:

```
#define _SECP256K1_HASH_IMPL_H_
```

In file secp256k1/src/field_5x52_int128_impl.h:

```
#define _SECP256K1_FIELD_INNER5X52_IMPL_H_
```

In file secp256k1/src/eckey_impl.h:

```
#define _SECP256K1_ECKEY_IMPL_H_
```

In file secp256k1/src/field.h:

```
#define _SECP256K1_FIELD_
```

In file secp256k1/src/scalar_impl.h:

```
#define _SECP256K1_SCALAR_IMPL_H_
```

In file secp256k1/src/scalar_8x32_impl.h:

```
#define _SECP256K1_SCALAR_REPR_IMPL_H_
```

In file secp256k1/src/ecdsa.h:

```
#define _SECP256K1_ECDSA_
```

In file secp256k1/src/group_impl.h:

```
#define _SECP256K1_GROUP_IMPL_H_
```

In file secp256k1/src/bench.h:

```
#define _SECP256K1_BENCH_H_
```

In file secp256k1/src/group.h:

```
#define _SECP256K1_GROUP_
```

In file secp256k1/src/util.h:

```
#define _SECP256K1_UTIL_H_
```

In file secp256k1/src/num_gmp.h:

```
#define _SECP256K1_NUM_REPR_
```

In file secp256k1/src/ecmult_gen.h:

```
#define _SECP256K1_ECMULT_GEN_
```

In file secp256k1/src/num.h:

```
#define _SECP256K1_NUM_
```

In file secp256k1/src/scalar.h:

```
#define _SECP256K1_SCALAR_
```

In file secp256k1/src/testrand_impl.h:

```
#define _SECP256K1_TESTRAND_IMPL_H_
```

In file secp256k1/src/ecdsa_impl.h:

```
#define _SECP256K1_ECDSA_IMPL_H_
```

In file secp256k1/src/testrand.h:

```
#define _SECP256K1_TESTRAND_H_
```

In file secp256k1/src/field_5x52_impl.h:

```
#define _SECP256K1_FIELD_REPR_IMPL_H_
```

In file secp256k1/src/field_10x26.h:

```
#define _SECP256K1_FIELD_REPR_
```

In file secp256k1/src/field_5x52.h:

```
#define _SECP256K1_FIELD_REPR_
```

In file secp256k1/src/scalar_4x64.h:

```
#define _SECP256K1_SCALAR_REPR_
```

In file secp256k1/src/scalar_8x32.h:

```
#define _SECP256K1_SCALAR_REPR_
```

In file secp256k1/src/eckey.h:

```
#define _SECP256K1_ECKEY_
```

In file secp256k1/src/field_10x26_impl.h:

```
#define _SECP256K1_FIELD_REPR_IMPL_H_
```

In file secp256k1/src/ecmult_gen_impl.h:

```
#define _SECP256K1_ECMULT_GEN_IMPL_H_
```

In file secp256k1/src/num_gmp_impl.h:

```
#define _SECP256K1_NUM_REPR_IMPL_H_
```

In file secp256k1/src/hash.h:

```
#define _SECP256K1_HASH_
```

In file secp256k1/src/field_impl.h:

```
#define _SECP256K1_FIELD_IMPL_H_
```

In file secp256k1/src/java/org_bitcoin_NativeSecp256k1.h:

```
#define _Included_org_bitcoin_NativeSecp256k1
```

In file zcash/prf.h:

```
#define _PRF_H_
```

In file zcash/JoinSplit.hpp:

```
#define _ZCJOINSPLIT_H_
```

In file zcash/util.h:

```
#define __ZCASH_UTIL_H
```

In file zcash/Address.hpp:

```
#define _ZCADDRESS_H_
```

In file zcash/Zcash.h:

```
#define _ZCCONSTANTS_H_
```

In file zcash/Note.hpp:

```
#define _ZCNOTE_H_
```

One solution is to just remove the "__" from the beginning of the name. Another solution is to use:

```
#pragma once
```

## Appendix E: Zcash and Dependencies - Fuzzing Efforts

NCC Group's Cryptography Services began fuzzing libsnark and the Zcash bitcoin fork in early July using AFL and AddressSanitizer. Two distinct efforts were made focusing on the libsnark fork in isolation and zcashd peer-to-peer network entry points. The former used the zkSNARK prover and verification routines without prior sanitization, this is different to the latter effort that executed deserialization logic similar to zcashd. Symbolic execution was also used in an attempt to generate better corpora to achieve greater coverage. NCC Group also performed cursory fuzzing of the JSON_Spirit implementation and Note encryption/decryption logic.

Isolated testing of libsnark produced a number of memory safety issues that have been raised outside of this report. Tests that mimicked Zcash networked data-flows were not affected by these vulnerabilities due to deserialization error handling and input validation. Despite this, NCC Group reminds the Zcash team that fuzzing should be integrated into their continuous test regime (or proofs be implemented in a memory safe language such as Rust).

The following programs are examples of template harnesses used in this exercise. Extract one exercises zkSNARK generator and witness and prover routines, the latter sample targets the `CheckTransaction` P2P entry point, both to be tailored accordingly.

```
#include <iostream>
#include <memory>
#include <cassert>
#include <cstdio>
#include <stdexcept>
#include <boost/foreach.hpp>
#include <boost/format.hpp>
#include <boost/optional.hpp>
#include <fstream>

#include "common/default_types/r1cs_ppzksnark_pp.hpp"
#include "common/profiling.hpp"
#include "common/utils.hpp"
#include "relations/constraint_satisfaction_problems/r1cs/examples/r1cs_examples.hpp"
#include "zk_proof_systems/ppzksnark/r1cs_ppzksnark/examples/run_r1cs_ppzksnark.hpp"

using namespace std;
using namespace libsnark;

int main(int argc, char** argv)
{
  default_r1cs_ppzksnark_pp::init_public_params();
  r1cs_example<Fr<default_r1cs_ppzksnark_pp> > system;

  ifstream paramFile("parameters.bin");
  paramFile>>system.constraint_system;
  paramFile.close();

  try {

    r1cs_ppzksnark_keypair<default_r1cs_ppzksnark_pp> keypair = r1cs_ppzksnark_generator<default_r1cs_ppz
➜   ksnark_pp>(system.constraint_system);
    r1cs_ppzksnark_proving_key<default_r1cs_ppzksnark_pp> pk = keypair.pk;

    ifstream witnessFile("witness.bin");
    r1cs_ppzksnark_primary_input<default_r1cs_ppzksnark_pp> witness;
    witnessFile>>witness;
    witnessFile.close();

    ifstream auxFile("aux.bin");
    r1cs_ppzksnark_auxiliary_input<default_r1cs_ppzksnark_pp> aux;
    auxFile>>aux;
    auxFile.close();

    r1cs_ppzksnark_prover<default_r1cs_ppzksnark_pp>(pk, witness, aux);

  } catch (exception &e) {

    cout << e.what() << endl;
    return -1;
  }

  return 0;
}
```

```
#include "primitives/transaction.h"
#include "clientversion.h"
#include "serialize.h"
#include "streams.h"
#include "uint256.h"
#include "util.h"
#include "utilstrencodings.h"
#include "../depends/x86_64-unknown-linux-gnu/include/boost/log/support/exception.hpp"
#include <iostream>
#include <stdexcept>
#include <fstream>
#include <string>

#include "main.h"
#include "consensus/validation.h"
#include "json/json_spirit_reader.h"
#include "json/json_spirit_reader_template.h"
#include "test/test_bitcoin.h"
#include "zcash/NoteEncryption.hpp"
#include "zcash/Zcash.h"
#include "zcash/JoinSplit.hpp"
#include "zcash/Proof.hpp"
#include "init.h"
#include "primitives/transaction.h"

using namespace json_spirit;
using namespace std;

//JSDescription js = ZCJoinSplit::Unopened();

int GetValidTransaction(JSDescription js) {
    CMutableTransaction mtx;
    mtx.vin.resize(2);
    mtx.vin[0].prevout.hash = uint256S("0000000000000000000000000000000000000000000000000000000000000001"
➜   );
    mtx.vin[0].prevout.n = 0;
    mtx.vin[1].prevout.hash = uint256S("0000000000000000000000000000000000000000000000000000000000000002"
➜   );
    mtx.vin[1].prevout.n = 0;
    mtx.vout.resize(2);
    mtx.vout[0].nValue = 0;
    mtx.vout[1].nValue = 0;
    mtx.vjoinsplit.resize(1);

    mtx.vjoinsplit[0] = js;

    uint256 joinSplitPubKey;
    unsigned char joinSplitPrivKey[crypto_sign_SECRETKEYBYTES];
    crypto_sign_keypair(joinSplitPubKey.begin(), joinSplitPrivKey);
    mtx.joinSplitPubKey = joinSplitPubKey;

    static const uint256 one(uint256S("0000000000000000000000000000000000000000000000000000000000000001"
➜   ));
    CScript scriptCode;
    CTransaction signTx(mtx);
    uint256 dataToBeSigned = SignatureHash(scriptCode, signTx, NOT_AN_INPUT, SIGHASH_ALL);

    crypto_sign_detached(&mtx.joinSplitSig[0], NULL,
                         dataToBeSigned.begin(), 32,
                         joinSplitPrivKey
                        );

    CTransaction ctx(mtx);
    CValidationState state;
    CheckTransaction(ctx, state);
    return 0;
}

struct Init : BasicTestingSetup
{
    Init() {
        fPrintToDebugLog = true;
    };
    ~Init() {};
};

int main(int argc, char** argv)
{

    Init();
    string str;
    ifstream infile(argv[1]);
    infile >> str;
    vector<unsigned char> tc(str.begin(), str.end());
    try {
        CDataStream cds(tc, SER_NETWORK, CLIENT_VERSION);
        JSDescription js;
        cds >> js;
        cout << GetValidTransaction(js) << endl;
    } catch (exception &ea) {

        cout << e.what() << endl;
        return -1;
    }
    return 0;
}
```

# Appendix F: Proof Constraints

The Zcash circuit relies on a set of constraints on the inputs to prove the following properties (see section 4.2.6[19] for more information):

- Merkle Path Validity
- Balance
- Nullifier Integrity
- Spend Authority
- Non-malleability
- Uniqueness of Rho
- Commitment Integrity

In addition to these straightforward constraints, several others exist. NCC Group's Cryptography Services identified the implemented constraints, starting with the `joinsplit_gadget` (the top-level entry point for proving and generating witnesses for the zkSNARK) and working through to libsnark. NCC Group performed only a cursory look at the constraints implemented by libsnark gadgets used by Zcash. Below are the constraints that were identified.

**Boolean constraints:**

- `zk_merkle_root`, the root of the Merkle Tree
- `zk_h_sig`, h_sig
- `zk_input_nullifiers`, the nullifiers for the notes to be spent
- `zk_output_commitments`, the commitments for each output note
- `zk_vpub_old`, the public value entering the confidential pool
- `zk_vpub_new`, the public value leaving the confidential pool
- `zk_phi`, random nonce used to generate nullifiers
- `a_sk`, the spending key
- `value_enforce`, the flag used during Merkle Tree verification and proving
- `zk_input_macs`, the authentication tags binding `h_sig` to the spending keys
- Values of `rho`
- New note nullifiers
- The value of all input notes
- Any output from the libsnark SHA256 gadget

**PRF constraints:**

- `a_pk = PRF(a_sk)` [Providing the Spend Authority property]
- `nullifier = PRF(a_sk, rho)` [Providing the Nullifier Integrity property]
- `zk_input_macs = PRF(zk_hi, zk_h_sig, i)` [Providing the Non-malleability property]
- `rho = PRF(phi, h_sig, note position) # output note` [Uniqueness of Rho property]

**Commitment constraints for input and output notes:**

- `commitment = sha256(a_pk, value, rho, r)`
- `commitment = sha256(a_pk, value, rho, r)` [Commitment Integrity property]

**Equality constraints:**

- `note value * (1 - value_enforce) = 0`
- `sum inputs == sum outputs` [Balance property]

**Merkle Tree constraints:**

- The Merkle Tree is path is constructed correctly [Merkle Path Validity property]

[19] https://github.com/zcash/zips/blob/zips27.reorganisation.0/protocol/protocol.pdf