coinspect

Grin

GRIN Source Code Audit

Prepared for Grin • October 2019

Grin v1014

1. Table of Contents

2. Executive Summary

In February 2019, Grin engaged Coinspect to audit the security of its MimbleWimble blockchain implementation.

During this engagement, Coinspect consultants used a hands-on approach to evaluate the platform security, which included:

- Source code review of Grin, including its core consensus rules, network protocols, and privacy features
- Rapid prototyping of potential attacks and proof of concept development

The objectives of the assessment included, but were not limited to, identifying the following types of security vulnerabilities: full system compromise, denial of service attacks, information disclosure, network protocol weaknesses, input validation, and misaligned incentives in consensus rules.

During the engagement, Coinspect identified the following issues:

Critical Risk	High Risk	Medium Risk	Low Risk
1	**5**	**7**	**1**

All these findings were remediated by the Grin team, and each fix was verified by Coinspect to be correct and complete during August and September 2019.

This report details the tasks performed and results obtained during this audit.

3. Source Code Audit

The following areas of the code were selected by Grin as the main focus for this engagement and were reviewed:

- The Grin core crate
- The Grin keychain crate
- The Grin chain crate
- The Grin wallet crate

All findings (except GRIN-001 and GRIN-002, which were discovered and reported before the engagement formal start date) have been identified and reproduced with local builds of Grin version 1.0.2, source code based on:

```
commit 7678aceddfd3e6af01632e2123f068457ee5164e
Author: Mark Renten <42224876+rentenmark@users.noreply.github.com>
Date:   Mon Mar 11 18:58:08 2019 -0400
Further
    Specify grin or nanogrins in API docs where applicable (#2642)
```

During the engagement timeframe, Coinspect was asked by the Grin team to switch its attention to the new wallet repository located in grin-wallet. There were no findings in this area of the codebase.

Overall, Coinspect found the project's source code to be clearly organized and readable, and most design and implementation decisions were oriented toward maintaining simplicity.

As a result of this audit, several vulnerabilities were identified and recommendations were made in order to fix them. Most of these findings can be grouped into the following categories, and special care should be taken to prevent these patterns from appearing again in the codebase:

1. Directory path traversal leading to remote code execution
2. Memory corruption vulnerabilities in unsafe code blocks located in third-party libraries
3. Denial of service caused by Rust panics, expects, and unhandled error conditions
4. Synchronization process denial of service caused by out-of-order P2P messages
5. Storage-based denial of service caused by failure to clean up temporary files
6. Node censorship through node ban feature abuse
7. Failure to ban ill-behaved nodes leading to CPU-based denial of service
8. Lack of validation of orphan blocks
9. Insecure file handling leading to local privilege escalation

The following list includes high-level weaknesses and suggested future work goals identified in order to improve the overall security of the codebase:

1. *Rust third-party libraries*: the quantity of Rust dependencies in use by the project exposes Grin to supply chain attacks, as well as a potential denial of service and/or remote code execution attacks via unsafe code blocks, as observed with the vulnerabilities reported related to the zip-rs and croaring-rs libraries.
2. *Transaction pool and new eviction policy*: even though Coinspect dedicated some time to understanding the transaction pool, it was not fully reviewed as it was outside the engagement scope. Coinspect recommends the pool and the new features be fully revised in order to rule out potential transaction spam attacks.
3. *Transaction and block processing times*: Coinspect suggests more tests are performed on a testnet in order to understand the impact of large transactions (with a big number of outputs, for example) on block processing and propagation delays, as long block validation times could be close to the 60-second block target time.
4. *Transaction creation workflow*: the current process for creating a new transaction requires the parties involved to establish and use a communication channel. The default mechanism currently offered by Grin relies on the sender and receiver directly connecting to each other via a non-encrypted channel, exposing their IP addresses and transactions slates and metadata to eavesdropping and man-in-the-middle attacks.

It is worth noting that the Grin codebase is under active development and new features are added and bugs fixed on a daily basis. Coinspect avoided reporting findings that were independently fixed and/or known by the Grin team before our notification. For example, the transaction pool implementation originally reviewed lacked the ability to evict transactions when full, allowing an attacker to fill it with cheap fee transactions, thus denying service to real users.

The full description for each finding is available in 6. Findings.

4. Remediations

On August 16, 2019, the Grin team provided Coinspect with a remediation report detailing one by one the fixes developed to address each finding reported. Every issue except #4, #10, and #11 were addressed by this report.

Coinspect reviewed each pull request source code and tested each vulnerability proof of concept on a new Grin build based on the following commit:

```
commit d220410571436a8659c8207a8cc47c15364019c0
Author: j01tz <47043188+j01tz@users.noreply.github.com>
Date:   Fri Aug 9 07:10:54 2019 -0700

   Improve error handling when computing PMMR roots (#2988)

   * Improve error handling for invalid PMMR roots

   * Update tests that rely on pmmr root

   * Fix pmmr store tests
```

Finally, in September 2019, Coinspect verified issues #10 and #11 had been properly addressed and merged into master based on the following commit:

```
commit 28d5ee8242be81d5629e4f4554df08261a926efe
Author: Antioch Peverell <apeverell@protonmail.com>
Date:   Thu Sep 12 21:04:09 2019 +0100

   Peer is_known robustness (#3040)

   * add some test coverage around peers map (peer_addr hashing impl)

   * make is_known a bit more robust

   * fix typos
```

Coinspect considers issue #4 properly fixed after the CRoaring library version update. Coinspect acknowledges that the Grin team understands the importance of keeping an eye on the library and is considering long-term alternatives, as stated in their remediation report and conversations in the project's Github. Coinspect believes this dependency represents a high-risk target that could potentially allow compromising Grin nodes.

To conclude, Coinspect considers every finding to be fully addressed in Grin version `grin 2.0.1-beta.1`. The pull request corresponding to each vulnerability remediation is listed below.

ID	Description	Pull Request
GRIN-001	Remote file system write access and code execution during TxHashSet processing	PR #2624
GRIN-002	PMMR panic after processing an invalid TxHashSet leaves node unable to sync	PR #2621
GRIN-003	zip-rs library panic and corrupted storage during TxHashSet processing results in node unable to sync	PR #2908
GRIN-004	CRoaring: memory corruption and DoS while processing bitmaps	PR #2763
GRIN-005	prune_list panic after processing an invalid TxHashSet leaves node unable to sync and restart	PR #2976
GRIN-006	Disk space DoS via TxHashSetRequest p2p messages	PR #2575
GRIN-007	Nodes can be indefinitely prevented from synchronizing the blockchain via unsolicited TxHashSetArchive p2p messages	PR #2984
GRIN-008	Insecure file handling local privilege escalation	PR #2753
GRIN-009	Nodes can be tricked into banning well-behaved peers (temporary file shared among peer threads)	PR #2753
GRIN-010	Node crashes when ulimit is reached with many incoming peer connections	PR #2985
GRIN-011	High CPU usage when handling many incoming peer connections results in stuck miner and unresponsive node	PR #2985
GRIN-012	Miner thread panic after long chain reorganization	PR #2988
GRIN-013	Arbitrary orphan blocks can be used to flush out legitimate ones from the OrphanBlockPool	PR #2981
GRIN-014	Known headers replay can be abused to clog victim node CPU with PoW computations	PR #2834

5. Summary Of Findings

ID	Description	Risk	Fixed
GRIN-001	Remote file system write access and code execution during TxHashSet processing	Critical	Yes
GRIN-002	PMMR panic after processing an invalid TxHashSet leaves node unable to sync	Medium	Yes
GRIN-003	zip-rs library panic and corrupted storage during TxHashSet processing results in node unable to sync	Medium	Yes
GRIN-004	CRoaring: memory corruption and DoS while processing bitmaps	High	Yes
GRIN-005	prune_list panic after processing an invalid TxHashSet leaves node unable to sync and restart	Medium	Yes
GRIN-006	Disk space DoS via TxHashSetRequest p2p messages	High	Yes
GRIN-007	Nodes can be indefinitely prevented from synchronizing the blockchain via unsolicited TxHashSetArchive p2p messages	High	Yes
GRIN-008	Insecure file handling local privilege escalation	Medium	Yes
GRIN-009	Nodes can be tricked into banning well-behaved peers (temporary file shared among peer threads)	High	Yes
GRIN-010	Node crashes when ulimit is reached with many incoming peer connections	High	Yes
GRIN-011	High CPU usage when handling many incoming peer connections results in stuck miner and unresponsive node	Medium	Yes
GRIN-012	Miner thread panic after long chain reorganization	Low	Yes
GRIN-013	Arbitrary orphan blocks can be used to flush out legitimate ones from the OrphanBlockPool	Medium	Yes
GRIN-014	Known headers replay can be abused to clog victim node CPU with PoW computations	Medium	Yes

6. Findings

GRIN-001 Remote file system write access and code execution during TxHashSet processing (CVE-2019-9195)

Total Risk	Impact	Location
Critical	High	util/src/zip.rs:85
Fixed	**Likelihood**	
Yes	High	

Description

Lack of input validation during the processing of TxHashSetArchive messages enables remote attackers to obtain file system write access and arbitrary code execution as a consequence.

TxHashSets are used during node chain synchronization. When a node lags behind the current chain tip a number of blocks above a certain threshold, a TxHashSet is requested from one of its peers. This TxHashSet consists of a ZIP encoded file containing data for three MMRs (output, rangeproof, and kernel).

This is the code responsible for unpacking the ZIP file, located in the zip.rs file, which depends on the zip-rs library:

```
/// Decompress a source file into the provided destination path.
pub fn decompress<R>(src_file: R, dest: &Path) -> ZipResult<()>
where
 R: io::Read + io::Seek,
{
 let mut archive = zip_rs::ZipArchive::new(src_file)?;

 for i in 0..archive.len() {
   let mut file = archive.by_index(i)?;
   let file_path = dest.join(file.name());

   if (&*file.name()).ends_with('/') {
     fs::create_dir_all(&file_path)?;
   } else {
     if let Some(p) = file_path.parent() {
       if !p.exists() {
         fs::create_dir_all(&p)?;
       }
     }

     //let mut outfile = fs::File::create(&file_path)?;
     let res = fs::File::create(&file_path);
```

The function above fails to validate the provided file names and extracts arbitrary files under the chain_data directory.

It is possible for an attacker to provide a malicious ZIP file with arbitrary content and file names using directory traversal sequences to write any part of the filesystem the node process has privileges to. **For example, an attacker could be able to overwrite the Grin node configuration, the system configuration files, or the cargo binary to achieve code execution.**

Recommendations

Use a whitelist to only allow the files included in the TxHashSet archive to be the exact set of expected files.

GRIN-002 PMMR panic after processing an invalid TxHashSet leaves node unable to sync

Total Risk	Impact	Location
Medium	Medium	core/pmmr/rewindable_pmmr.rs:109
Fixed	Likelihood	
Yes	High	

Description

Improper error handling during the processing of non-trusted PMMR data structures results in a panic in a peer thread, which leaves the node in an inconsistent state and unable to synchronize.

TxHashSets are used during node chain synchronization. When a node lags behind the current chain tip a number of blocks above a certain threshold, a TxHashSet is requested from one of its peers. This TxHashSet consists of a ZIP encoded file containing data for three MMRs (output, rangeproof, and kernel).

The next code is used to calculate the root of an MMR, which could have been provided by another node:

```
/// Computes the root of the MMR. Find all the peaks in the current
/// tree and "bags" them to get a single peak.
pub fn root(&self) -> Result<Hash, String> {
  if self.is_empty() {
    return Ok(ZERO_HASH);
  }
  let mut res = None;
  for peak in self.peaks().iter().rev() {
    res = match res {
      None => Some(*peak),
      Some(rhash) => Some((*peak, rhash).hash_with_index(self.unpruned_size())),
    }
  }
  res.expect("no root, invalid tree")
}
```

As expect is used, parsing errors result in a panic.

Then, during the fast sync window, it is possible for an attacker to provide a malicious ZIP file and crash the peer thread:

```
20190221 18:40:35.796 INFO grin_util::logger - log4rs is initialized, file level: Debug, stdout level:
Info, min. level: Debug
20190221 18:40:35.796 INFO grin - Using configuration file at
/home/u/GRIN/grin/config.jp.UserTesting/n2/grin-server.toml
20190221 18:40:35.797 INFO grin - This is Grin version 1.0.1 (git v1.0.1-29-gdc6542d), built for
```

```
x86_64-unknown-linux-gnu by rustc 1.32.0 (9fda7c223 2019-01-16).
20190221 18:40:35.797 WARN grin::cmd::server - Starting GRIN w/o UI...
20190221 18:40:35.798 INFO grin_servers::grin::server - Starting server, genesis block: 32c1193af373
20190221 18:40:36.468 INFO grin_servers::grin::server - Starting rest apis at: 127.0.0.1:4413
20190221 18:40:36.468 INFO grin_api::handlers - Starting HTTP API server at 127.0.0.1:4413.
20190221 18:40:36.469 INFO grin_servers::grin::server - Starting dandelion monitor: 127.0.0.1:4413
20190221 18:40:36.469 WARN grin_servers::grin::server - Grin server started.
20190221 18:41:08.612 INFO grin_servers::common::adapters - Received 29 block headers from
192.168.1.117:3414
20190221 18:41:11.137 ERROR grin_store::types - Corrupted storage, could not read an entry from data
file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190221 18:41:11.137 ERROR grin_store::types - Corrupted storage, could not read an entry from data
file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190221 18:41:11.138 ERROR grin_store::types - Corrupted storage, could not read an entry from data
file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190221 18:41:11.138 ERROR grin_store::types - Corrupted storage, could not read an entry from data
file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190221 18:41:12.775 ERROR grin_util::logger -
thread 'peer' panicked at 'no root, invalid tree': src/libcore/option.rs:1008stack backtrace:
   0: grin_util::logger::send_panic_to_log::{{closure}}::h223b6a9e01ac8c08 (0x7f574b1ea807)
             at util/src/logger.rs:237
   1: std::panicking::rust_panic_with_hook::h8cbdfe43764887be (0x7f574b6e64e9)
             at src/libstd/panicking.rs:495
   2: std::panicking::continue_panic_fmt::h3d3c5a833c00a5e1 (0x7f574b6e5f91)
             at src/libstd/panicking.rs:398
   3: rust_begin_unwind (0x7f574b6e5e75)
             at src/libstd/panicking.rs:325
   4: core::panicking::panic_fmt::h4d67173bc68f6d5a (0x7f574b702ffc)
             at src/libcore/panicking.rs:95
   5: core::option::expect_failed::h2f881c519f1d8001 (0x7f574b703072)
             at src/libcore/option.rs:1008
   6: <core::option::Option<T>>::expect::hc9e2530f58cd55a8 (0x7f574af579f6)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libcore/option.rs:322
   7: <grin_core::core::pmmr::rewindable_pmmr::RewindablePMMR<'a, T, B>>::root::h3c6e1b06e27fc3b6
(0x7f574b02cfe2)
             at /home/u/GRIN/grin/core/src/core/pmmr/rewindable_pmmr.rs:109
   8:
grin_chain::txhashset::rewindable_kernel_view::RewindableKernelView::validate_root::h9e72c34e2489386c
(0x7f574b0091cb)
             at chain/src/txhashset/rewindable_kernel_view.rs:71
   9: grin_chain::chain::Chain::validate_kernel_history::{{closure}}::h72a8a65d8e2f4365
(0x7f574afdcb59)
             at chain/src/chain.rs:718
  10: grin_chain::txhashset::txhashset::rewindable_kernel_view::hb668a07e19744eb3 (0x7f574b010c83)
             at chain/src/txhashset/txhashset.rs:401
  11: grin_chain::chain::Chain::validate_kernel_history::hf6cb0efa494dab56 (0x7f574affc8a8)
             at chain/src/chain.rs:715
  12: grin_chain::chain::Chain::txhashset_write::h5ddfc640845ea10c (0x7f574afffa42)
             at chain/src/chain.rs:878
  13: <grin_servers::common::adapters::NetToChainAdapter as
grin_p2p::types::ChainAdapter>::txhashset_write::h96375b5efa49becc (0x7f574a23d720)
             at servers/src/common/adapters.rs:352
  14: <grin_p2p::peers::Peers as grin_p2p::types::ChainAdapter>::txhashset_write::h8b00ccf90d98de01
(0x7f574ae95e11)
             at p2p/src/peers.rs:640
  15: <grin_p2p::peer::TrackingAdapter as
grin_p2p::types::ChainAdapter>::txhashset_write::h7d0b7762c4887eb7 (0x7f574aeb1c75)
             at p2p/src/peer.rs:622
  16: <grin_p2p::protocol::Protocol as grin_p2p::conn::MessageHandler>::consume::ha647bc0e1dc4336f
(0x7f574aeef9a4)
             at p2p/src/protocol.rs:337
  17: grin_p2p::conn::poll::{{closure}}::h9d2a4d299ac846f5 (0x7f574af28acf)
```

```
             at p2p/src/conn.rs:269
  18: std::sys_common::backtrace::__rust_begin_short_backtrace::ha78818f713c7badf (0x7f574af2cb52)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/sys_common/backtrace.rs:136
  19: std::thread::Builder::spawn_unchecked::{{closure}}::{{closure}}::h6ffda85dc1433e11
(0x7f574af2c815)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:477
  20: <std::panic::AssertUnwindSafe<F> as
core::ops::function::FnOnce<()>>::call_once::h88c630326803d115 (0x7f574af2ca55)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:319
  21: std::panicking::try::do_call::hb2136a08da364d92 (0x7f574af2cd09)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:310
  22: __rust_maybe_catch_panic (0x7f574b6fc2d9)
             at src/libpanic_unwind/lib.rs:102
  23: std::panicking::try::hb8c8611d6c8b089f (0x7f574af2cbef)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:289
  24: std::panic::catch_unwind::h4a53d0dd05504755 (0x7f574af2ca95)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:398
  25: std::thread::Builder::spawn_unchecked::{{closure}}::h7403e59a285d8b6f (0x7f574af2c5fd)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:476
  26: <F as alloc::boxed::FnBox<A>>::call_box::hb08ce337190283d8 (0x7f574af2c8e7)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:673
  27: <alloc::boxed::Box<(dyn alloc::boxed::FnBox<A, Output=R> + 'a)> as
core::ops::function::FnOnce<A>>::call_once::hece536cf07b94f8d (0x7f574b6efe9d)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:683
      std::sys_common::thread::start_thread::h9605a7df0f911844
             at src/libstd/sys_common/thread.rs:24
      std::sys::unix::thread::Thread::new::thread_start::hca8e72c41fa9d291
             at src/libstd/sys/unix/thread.rs:90
  28: start_thread (0x7f574873a183)
  29: clone (0x7f574825103c)
  30: <unknown> (0x0)


Thread 'peer' panicked with message:
"no root, invalid tree"
```

Even though the node continues operating normally, as a consequence of this panic, **the synchronization process is left in an inconsistent state that prevents the node from synchronizing with the rest of its peers**.

Recommendations

In order to prevent panics, validate the archive contains the expected contents before processing them, and handle malformed PMMRs gracefully.

GRIN-003 zip-rs library panic and corrupted storage during TxHashSet processing results in node unable to sync

Total Risk	Impact	Location
Medium	Medium	zip-0.4.2/src/read.rs:85 store/src/types.rs:66
Fixed	**Likelihood**	
Yes	High	

Description

Lack of input validation in the zip-rs library used by Grin during the processing of TxHashSetArchive messages enables remote attackers to crash the peer thread. As a result, the storage is left in an inconsistent state, which prevents the node from properly synchronizing with any peer.

TxHashSets are used during node chain synchronization. When a node lags behind the current chain tip a number of blocks above a certain threshold, a TxHashSet is requested from one of its peers. This TxHashSet consists of a ZIP encoded file containing data for three MMRs (output, rangeproof, and kernel).

This is the code responsible for unpacking the ZIP file, located in the read.rs file, part of the zip-rs library:

```
let search_upper_bound = cde_start_pos
        .checked_sub(60) // minimum size of Zip64CentralDirectoryEnd + Zip64CentralDirectoryEndLocator
        .ok_or(ZipError::InvalidArchive("File cannot contain ZIP64 central directory end"))?;
let (footer, archive_offset) = spec::Zip64CentralDirectoryEnd::find_and_parse(
        Reader,
        Locator64.end_of_central_directory_offset,
        search_upper_bound)?;

if footer.disk_number != footer.disk_with_central_directory {
        return unsupported_zip_error("Support for multi-disk files is not implemented")
}

// arithmetic overflow triggered by user controlled values
let directory_start = footer.central_directory_offset + archive_offset;
Ok((archive_offset, directory_start, footer.number_of_files as usize))
```

The code above is responsible for parsing ZIP 64 format files. It fails to validate the user-controlled values central_directory_offset and archive_offset.

During the fast sync window, it is possible for an attacker to provide a malicious ZIP file and crash the peer thread:

```
which is not on local chain: d456d242ca3c at 1
20190311 15:27:31.493 DEBUG grin_servers::grin::sync::body_sync - body_sync: cannot sync full blocks
earlier than horizon. will request txhashset
20190311 15:27:31.502 DEBUG grin_servers::grin::sync::state_sync - state_sync: before txhashset
request, header head: 1992 / 40e2f3bc79f7, txhashset_head: 1971 / 3ea8667af90e
20190311 15:27:31.503 DEBUG grin_p2p::peer - Asking 192.168.1.117:3414 for txhashset archive at 1971
3ea8667af90e.
20190311 15:27:31.503 DEBUG grin_servers::common::types - sync_state: sync_status: HeaderSync {
current_height: 1536, highest_height: 1992 } -> HeaderSync { current_height: 1992, highest_height:
1992 }
20190311 15:27:31.503 DEBUG grin_servers::common::types - sync_state: sync_status: HeaderSync {
current_height: 1992, highest_height: 1992 } -> TxHashsetDownload { start_time:
2019-03-11T18:27:31.503400991Z, downloaded_size: 0, total_size: 0 }
20190311 15:27:32.385 DEBUG grin_p2p::protocol - handle_payload: txhashset archive for 3ea8667af90e at
1971. size=208
20190311 15:27:32.426 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetDownload {
start_time: 2019-03-11T18:27:32.385793842Z, downloaded_size: 208, total_size: 208 } -> TxHashsetSetup
20190311 15:27:32.426 DEBUG grin_chain::chain - txhashset_write: body_head - 32c1193af373, 0,
header_head - 40e2f3bc79f7, 1992, sync_head - 40e2f3bc79f7, 1992
20190311 15:27:33.233 DEBUG grin_chain::chain - txhashset_write: need a state sync for txhashset.
oldest block which is not on local chain: d456d242ca3c at 1
20190311 15:27:37.242 ERROR grin_util::logger -
thread 'peer' panicked at 'attempt to add with overflow':
/home/u/.cargo/registry/src/github.com-1ecc6299db9ec823/zip-0.4.2/src/read.rs:205stack backtrace:
   0: grin_util::logger::send_panic_to_log::{{closure}}::h223b6a9e01ac8c08 (0x7fdd96971a07)
             at util/src/logger.rs:237
   1: std::panicking::rust_panic_with_hook::h8cbdfe43764887be (0x7fdd96e6d6e9)
             at src/libstd/panicking.rs:495
   2: std::panicking::continue_panic_fmt::h3d3c5a833c00a5e1 (0x7fdd96e6d191)
             at src/libstd/panicking.rs:398
   3: rust_begin_unwind (0x7fdd96e6d075)
             at src/libstd/panicking.rs:325
   4: core::panicking::panic_fmt::h4d67173bc68f6d5a (0x7fdd96e8a1fc)
             at src/libcore/panicking.rs:95
   5: core::panicking::panic::h6f50c0de2dcd7974 (0x7fdd96e8a12b)
             at src/libcore/panicking.rs:59
   6: <zip::read::ZipArchive<R>>::get_directory_counts::h5c3b2e5b9e6db52a (0x7fdd96769cfd)
             at /home/u/.cargo/registry/src/github.com-1ecc6299db9ec823/zip-0.4.2/src/read.rs:205
   7: <zip::read::ZipArchive<R>>::new::h95ac4dcc79c76be6 (0x7fdd9676a013)
             at /home/u/.cargo/registry/src/github.com-1ecc6299db9ec823/zip-0.4.2/src/read.rs:221
   8: grin_util::zip::decompress::h8ec6131787814ab4 (0x7fdd967a6b78)
             at /home/u/GRIN/grin/util/src/zip.rs:69
   9: grin_chain::txhashset::txhashset::zip_write::h116d29a9a141c431 (0x7fdd967302f3)
             at chain/src/txhashset/txhashset.rs:1432
  10: grin_chain::chain::Chain::txhashset_write::h5ddfc640845ea10c (0x7fdd96785690)
             at chain/src/chain.rs:868
  11: <grin_servers::common::adapters::NetToChainAdapter as
grin_p2p::types::ChainAdapter>::txhashset_write::h96375b5efa49becc (0x7fdd959c2c10)
```

```
             at servers/src/common/adapters.rs:352
  12: <grin_p2p::peers::Peers as grin_p2p::types::ChainAdapter>::txhashset_write::h8b00ccf90d98de01
(0x7fdd9661c001)
             at p2p/src/peers.rs:640
  13: <grin_p2p::peer::TrackingAdapter as
grin_p2p::types::ChainAdapter>::txhashset_write::h7d0b7762c4887eb7 (0x7fdd96637e65)
             at p2p/src/peer.rs:622
  14: <grin_p2p::protocol::Protocol as grin_p2p::conn::MessageHandler>::consume::ha647bc0e1dc4336f
(0x7fdd96675b94)
             at p2p/src/protocol.rs:337
  15: grin_p2p::conn::poll::{{closure}}::h9d2a4d299ac846f5 (0x7fdd966aecbf)
             at p2p/src/conn.rs:269
  16: std::sys_common::backtrace::__rust_begin_short_backtrace::ha78818f713c7badf (0x7fdd966b2d42)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/sys_common/backtrace.rs:136
  17: std::thread::Builder::spawn_unchecked::{{closure}}::{{closure}}::h6ffda85dc1433e11
(0x7fdd966b2a05)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:477
  18: <std::panic::AssertUnwindSafe<F> as
core::ops::function::FnOnce<()>>::call_once::h88c630326803d115 (0x7fdd966b2c45)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:319
  19: std::panicking::try::do_call::hb2136a08da364d92 (0x7fdd966b2ef9)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:310
  20: __rust_maybe_catch_panic (0x7fdd96e834d9)
             at src/libpanic_unwind/lib.rs:102
  21: std::panicking::try::hb8c8611d6c8b089f (0x7fdd966b2ddf)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:289
  22: std::panic::catch_unwind::h4a53d0dd05504755 (0x7fdd966b2c85)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:398
  23: std::thread::Builder::spawn_unchecked::{{closure}}::h7403e59a285d8b6f (0x7fdd966b27ed)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:476
  24: <F as alloc::boxed::FnBox<A>>::call_box::hb08ce337190283d8 (0x7fdd966b2ad7)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:673
  25: <alloc::boxed::Box<(dyn alloc::boxed::FnBox<A, Output=R> + 'a)> as
core::ops::function::FnOnce<A>>::call_once::hece536cf07b94f8d (0x7fdd96e7709d)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:683
      std::sys_common::thread::start_thread::h9605a7df0f911844
             at src/libstd/sys_common/thread.rs:24
      std::sys::unix::thread::Thread::new::thread_start::hca8e72c41fa9d291
             at src/libstd/sys/unix/thread.rs:90
  26: start_thread (0x7fdd93ebe183)
  27: clone (0x7fdd939d503c)
  28: <unknown> (0x0)
```

Even though the node continues operating normally, as a consequence of this panic, **the storage is left in an inconsistent state that prevents the node from synchronizing with the rest of its peers**:

```
20190311 16:08:18.570 ERROR grin_store::types - Corrupted storage, could not read an entry from data
file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190311 16:08:18.571 ERROR grin_chain::chain - body_sync: something is wrong! oldest_height is 0
20190311 16:08:21.331 ERROR grin_store::types - Corrupted storage, could not read an entry from data
file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190311 16:08:21.332 ERROR grin_chain::chain - body_sync: something is wrong! oldest_height is 0
20190311 16:08:24.039 INFO grin_servers::common::adapters - Received 1 block headers from
192.168.1.117:3414
20190311 16:08:26.264 INFO grin_servers::grin::sync::header_sync - sync: ban a fraud peer:
192.168.1.117:3414, claimed height: 2169, total difficulty: 119527
20190311 16:08:26.275 INFO grin_servers::grin::sync::syncer - synchronized at 3 @ 0 [32c1193af373]
20190311 16:09:06.321 INFO grin_servers::grin::sync::syncer - sync: total_difficulty 3,
peer_difficulty 119659, threshold 3 (last 5 blocks), enabling sync
20190311 16:09:13.774 INFO grin_servers::common::adapters - Received 6 block headers from
192.168.1.117:3414
20190311 16:09:16.630 ERROR grin_store::types - Corrupted storage, could not read an entry from data
file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190311 16:09:16.631 ERROR grin_chain::chain - body_sync: something is wrong! oldest_height is 0

20190311 16:11:31.759 DEBUG grin_servers::common::types - sync_state: sync_status: HeaderSync {
current_height: 2177, highest_height: 2178 } -> BodySync { current_height: 0, highest_height: 2178 }
20190311 16:11:32.764 DEBUG grin_servers::common::types - sync_state: sync_status: BodySync {
current_height: 0, highest_height: 2178 } -> HeaderSync { current_height: 2178, highest_height: 2178 }
```

Recommendations

Consider parsing performed by potentially unsafe external dependencies and guard against incomplete operations that leave the node in an inconsistent state.

Timeout if the synchronization is not successful after some time and restart the process with a different node.

Also, ban nodes that send malformed messages.

GRIN-004 CRoaring: memory corruption and DoS while processing bitmaps

Total Risk	Impact	Location
High	High	croaring-rs CRoaring/roaring.c:7917 CRoaring/roaring.c:7326 CRoaring/roaring.c:8988 CRoaring/roaring.c:8751
Fixed **Yes**	Likelihood High	

Description

Lack of input validation during the processing of CRoaring bitmaps enables remote attackers to, at least, crash Grin nodes.

This vulnerability was found after an analysis of the Grin code dependencies was performed. This library was identified as a promising target because:

1. A huge unsafe code attack surface
2. It is used to parse externally supplied binary files
3. Existent Github crash reports were identified

CRoaring bitmaps are utilized to store and manipulate multiple Grin data structures:

```
./core/src/pow/cuckatoo.rs:use croaring::Bitmap;
./core/src/pow/lean.rs:use croaring::Bitmap;
./core/src/pow/lean.rs:/// croaring which is likely sub-optimal for this task.
./core/src/core/pmmr/backend.rs:use croaring::Bitmap;
./core/src/core/pmmr/pmmr.rs:use croaring::Bitmap;
./chain/src/store.rs:use croaring::Bitmap;
./chain/src/txhashset/txhashset.rs:use croaring::Bitmap;
./store/tests/test_bitmap.rs:use croaring::Bitmap;
./store/tests/utxo_set_perf.rs:use croaring::Bitmap;
./store/tests/pmmr.rs:use croaring::Bitmap;
./store/src/leaf_set.rs:use croaring::Bitmap;
./store/src/lib.rs:use croaring::Bitmap;
./store/src/prune_list.rs:use croaring::Bitmap;
./store/src/pmmr.rs:use croaring::Bitmap;
```

The croaring-rs library wraps the C/C++ implementation located in https://github.com/RoaringBitmap/CRoaring. However, the Rust wrapper used by Grin does not link to the latest version of the C/C++ implementation. Coinspect consultants were able to easily find public test files that are known to crash older versions of the library, which are even included as part of the library's own test harness.

During the fast sync window, it is possible for an attacker to provide a malicious TxHashSet file, including corrupted bitmaps, resulting in a segmentation fault in unsafe code blocks. As a result, the node crashes and cannot be restarted until the chain data directory is manually removed.

The following stack traces exemplify different crash locations within the CRoaring library:

```
[1]
Program received signal SIGSEGV, Segmentation fault.
roaring_bitmap_add (r=0x0, val=1215) at CRoaring/roaring.c:7917
7917        const int i = ra_get_index(&r->high_low_container, hb);

[2]
Program received signal SIGSEGV, Segmentation fault.
0x00007ffff6e0ea5e in ?? () from /lib/x86_64-linux-gnu/libc.so.6
(gdb) x/i $pc
=> 0x7ffff6e0ea5e:      movdqu -0x10(%rsi,%rdx,1),%xmm8
(gdb) bt
#0  0x00007ffff6e0ea5e in ?? () from /lib/x86_64-linux-gnu/libc.so.6
#1  0x00005555572f1d75 in memcpy (__len=<optimized out>, __src=0x55555822b306, __dest=<optimized out>)
at /usr/include/x86_64-linux-gnu/bits/string3.h:51
#2 run_container_read (cardinality=cardinality@entry=1, container=container@entry=0x55555822b6f0,
buf=buf@entry=0x55555822b304 "\376\377") at CRoaring/roaring.c:7326
#3 0x00005555572f9c49 in ra_portable_deserialize (answer=answer@entry=0x5555581f0840,
buf=0x55555822b304 "\376\377", buf@entry=0x5555581f06b0 ";0;0\377",
maxbytes=maxbytes@entry=18446744073709551615,
    readbytes=readbytes@entry=0x7fffffffeb560) at CRoaring/roaring.c:11068
#4 0x00005555572f9dc7 in roaring_bitmap_portable_deserialize_safe (buf=0x5555581f06b0 ";0;0\377",
maxbytes=18446744073709551615) at CRoaring/roaring.c:8865
#5 0x000055555701a69e in
croaring::imp::_$LT$impl$u20$croaring..Bitmap$GT$::deserialize::ha96b992e9faf1831 (buffer=...) at
/home/u/.cargo/registry/src/github.com-1ecc6299db9ec823/croaring-0.3.8/src/imp.rs:779
#6 0x0000555557036877 in grin_store::read_bitmap::hd144388c9d0f8b78 (file_path=0x7fffffffebb80) at
/home/u/GRIN/grin/store/src/lib.rs:120
#7 0x0000555557034d5e in grin_store::prune_list::PruneList::open::h212063bb3b40e761
(path=0x7fffffffec788) at /home/u/GRIN/grin/store/src/prune_list.rs:69
#8 0x0000555556fe9936 in _$LT$grin_store..pmmr..PMMRBackend$LT$T$GT$$GT$::new::h81f278b9827293fd
(data_dir=..., prunable=true, header=...) at /home/u/GRIN/grin/store/src/pmmr.rs:219
#9 0x0000555556f68367 in
_$LT$grin_chain..txhashset..txhashset..PMMRHandle$LT$T$GT$$GT$::new::h45d8c699de9dc6a5 (root_dir=...,
sub_dir=..., file_name=..., prunable=true, header=...) at chain/src/txhashset/txhashset.rs:69
#10 0x0000555556f7eba0 in grin_chain::txhashset::txhashset::TxHashSet::open::hc3e3c6ab04000454
(root_dir=..., commit_index=..., header=...) at chain/src/txhashset/txhashset.rs:130
#11 0x0000555556f4b583 in grin_chain::chain::Chain::init::h8297f25047c202cd (db_root=..., db_env=...,
adapter=..., genesis=..., pow_verifier=0x5555572bded0
<grin_core::pow::verify_size::hd019bffbfd407afe>,
    verifier_cache=..., archive_mode=false, stop_state=...) at chain/src/chain.rs:183
#12 0x0000555556386d5c in grin_servers::grin::server::Server::new::h112171a8cdbbe207 (config=...) at
servers/src/grin/server.rs:179
#13 0x0000555555e39247 in grin_servers::grin::server::Server::start::h124d013591b00388 (config=...,
info_callback=...) at /home/u/GRIN/grin/servers/src/grin/server.rs:83
#14 0x0000555555f1a57f in grin::cmd::server::start_server_tui::h9b979ce79f882862 (config=...) at
src/bin/cmd/server.rs:62
#15 0x0000555555f1a055 in grin::cmd::server::start_server::h4927356618c1e453 (config=...) at
src/bin/cmd/server.rs:33
#16 0x0000555555f1ad00 in grin::cmd::server::server_command::he33efd3028c213d0 (server_args=...,
global_config=...) at src/bin/cmd/server.rs:145
#17 0x0000555555d32268 in grin::real_main::hc73dff343620e9d9 () at src/bin/grin.rs:193
#18 0x0000555555d30b16 in grin::main::hfb66e9a2739dc55d () at src/bin/grin.rs:68
#19 0x0000555555ee1800 in std::rt::lang_start::_$u7b$$u7b$closure$u7d$$u7d$::h0aaed896f4204d80 () at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/rt.rs:74
#20 0x00005555578609a3 in {{closure}} () at src/libstd/rt.rs:59
#21 std::panicking::try::do_call::h69790245ac2d03fe () at src/libstd/panicking.rs:310
#22 0x000055555787664a in __rust_maybe_catch_panic () at src/libpanic_unwind/lib.rs:102
#23 0x0000555557861374 in try<i32,closure> () at src/libstd/panicking.rs:289
#24 catch_unwind<closure,i32> () at src/libstd/panic.rs:398
```

```
#25 std::rt::lang_start_internal::h540c897fe52ba9c5 () at src/libstd/rt.rs:58
#26 0x0000555555ee17d9 in std::rt::lang_start::h144219d38079d061 (main=0x555555d30b10
<grin::main::hfb66e9a2739dc55d>, argc=1, argv=0x7fffffffdb58) at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/rt.rs:74
#27 0x0000555555d329ea in main ()

[3]
Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7ffd75663700 (LWP 32091)]
0x00007ffff6e0ea5e in ?? () from /lib/x86_64-linux-gnu/libc.so.6
(gdb) bt
#0  0x00007ffff6e0ea5e in ?? () from /lib/x86_64-linux-gnu/libc.so.6
#1 0x00005555572ecc9a in memcpy (__len=8192, __src=0x5555572ec79c <bitset_container_clear+28>,
__dest=<optimized out>) at /usr/include/x86_64-linux-gnu/bits/string3.h:51
#2 bitset_container_read (cardinality=cardinality@entry=64552,
container=container@entry=0x7ffd481a9470, buf=buf@entry=0x7ffd4841cf01 "") at CRoaring/roaring.c:3903
#3 0x00005555572f9ad1 in ra_portable_deserialize (answer=answer@entry=0x7ffd48315bd0,
buf=0x7ffd4841cf01 "", buf@entry=0x7ffd482176d0 ";000\375\177",
maxbytes=maxbytes@entry=18446744073709551615,
    readbytes=readbytes@entry=0x7ffd7564f210) at CRoaring/roaring.c:11039
#4 0x00005555572f9dc7 in roaring_bitmap_portable_deserialize_safe (buf=0x7ffd482176d0 ";000\375\177",
maxbytes=18446744073709551615) at CRoaring/roaring.c:8865
#5 0x000055555701a69e in
croaring::imp::_$LT$impl$u20$croaring..Bitmap$GT$::deserialize::ha96b992e9faf1831 (buffer=...) at
/home/u/.cargo/registry/src/github.com-1ecc6299db9ec823/croaring-0.3.8/src/imp.rs:779
#6 0x0000555557036877 in grin_store::read_bitmap::hd144388c9d0f8b78 (file_path=0x7ffd7564f830) at
/home/u/GRIN/grin/store/src/lib.rs:120
#7 0x0000555557034d5e in grin_store::prune_list::PruneList::open::h212063bb3b40e761
(path=0x7ffd75650438) at /home/u/GRIN/grin/store/src/prune_list.rs:69
#8 0x0000555556fe9936 in _$LT$grin_store..pmmr..PMMRBackend$LT$T$GT$$GT$::new::h81f278b9827293fd
(data_dir=..., prunable=true, header=...) at /home/u/GRIN/grin/store/src/pmmr.rs:219
#9 0x0000555556f68367 in
_$LT$grin_chain..txhashset..txhashset..PMMRHandle$LT$T$GT$$GT$::new::h45d8c699de9dc6a5 (root_dir=...,
sub_dir=..., file_name=..., prunable=true, header=...) at chain/src/txhashset/txhashset.rs:69
#10 0x0000555556f7eba0 in grin_chain::txhashset::txhashset::TxHashSet::open::hc3e3c6ab04000454
(root_dir=..., commit_index=..., header=...) at chain/src/txhashset/txhashset.rs:130
#11 0x0000555556f5760c in grin_chain::chain::Chain::txhashset_write::h790db0ae32cffafd
(self=0x5555581f46b0, h=..., txhashset_data=..., status=...) at chain/src/chain.rs:871
#12 0x00005555563d5ac1 in
_$LT$grin_servers..common..adapters..NetToChainAdapter$u20$as$u20$grin_p2p..types..ChainAdapter$GT$::t
xhashset_write::h0f4497da31f717d1 (self=0x5555581f4850, h=..., txhashset_data=...,
    _peer_addr=...) at servers/src/common/adapters.rs:359
#13 0x0000555556e9fdc2 in
_$LT$grin_p2p..peers..Peers$u20$as$u20$grin_p2p..types..ChainAdapter$GT$::txhashset_write::h0497557344
92d693 (self=0x5555584f79b0, h=..., txhashset_data=..., peer_addr=...) at p2p/src/peers.rs:623
#14 0x0000555556e803c6 in
_$LT$grin_p2p..peer..TrackingAdapter$u20$as$u20$grin_p2p..types..ChainAdapter$GT$::txhashset_write::h9
5f93b3fbb99844c (self=0x7ffd60000ab0, h=..., txhashset_data=..., peer_addr=...)
    at p2p/src/peer.rs:581
#15 0x0000555556ef0068 in
_$LT$grin_p2p..protocol..Protocol$u20$as$u20$grin_p2p..conn..MessageHandler$GT$::consume::hdc830875c29
c78c7 (self=0x7ffd756610c8, msg=..., writer=..., received_bytes=...)
    at p2p/src/protocol.rs:337
#16 0x0000555556ebe9e0 in grin_p2p::conn::poll::_$u7b$$u7b$closure$u7d$$u7d$::hcb003bb06128f8d7 () at
p2p/src/conn.rs:268
#17 0x0000555556f210c3 in std::sys_common::backtrace::__rust_begin_short_backtrace::h354fe58fb798d21f
(f=...) at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/sys_common/backtrace.rs:136
#18 0x0000555556f1f906 in
std::thread::Builder::spawn_unchecked::_$u7b$$u7b$closure$u7d$$u7d$::_$u7b$$u7b$closure$u7d$$u7d$::h56
a170e6183223d0 ()
    at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:477
```

```
#19 0x0000555556f1fa86 in
_$LT$std..panic..AssertUnwindSafe$LT$F$GT$$u20$as$u20$core..ops..function..FnOnce$LT$$LP$$RP$$GT$$GT$:
:call_once::h6e199459a8b512f0 (self=..., _args=0)
    at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:319
#20 0x0000555556f2128a in std::panicking::try::do_call::he59c50499165a66f (data=0x7ffd75661470 "") at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:310
#21 0x000055555787664a in __rust_maybe_catch_panic () at src/libpanic_unwind/lib.rs:102
#22 0x0000555556f21170 in std::panicking::try::he8612998811b7ebf (f=...) at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:289
#23 0x0000555556f1fac6 in std::panic::catch_unwind::h57e841abb0e53419 (f=...) at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:398
#24 0x0000555556f1f6ee in
std::thread::Builder::spawn_unchecked::_$u7b$$u7b$closure$u7d$$u7d$::hf4d6a7535a6deab7 () at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:476
#25 0x0000555556f1f9d8 in
_$LT$F$u20$as$u20$alloc..boxed..FnBox$LT$A$GT$$GT$::call_box::h586b545677055569 (self=0x7ffd60002750,
args=0) at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:673
#26 0x000055555786a20e in call_once<(),()> () at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:683
#27 start_thread () at src/libstd/sys_common/thread.rs:24
#28 std::sys::unix::thread::Thread::new::thread_start::hca8e72c41fa9d291 () at
src/libstd/sys/unix/thread.rs:90
#29 0x00007ffff735b184 in start_thread () from /lib/x86_64-linux-gnu/libpthread.so.0
#30 0x00007ffff6e7203d in clone () from /lib/x86_64-linux-gnu/libc.so.6

[4]
Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7ffd75864700 (LWP 32648)]
loadfirstvalue (newit=0x7ffd551fc8f0) at CRoaring/roaring.c:8988
8988                    newit->highbits |
(gdb) bt
#0  loadfirstvalue (newit=0x7ffd551fc8f0) at CRoaring/roaring.c:8988
#1 roaring_init_iterator (ra=ra@entry=0x7ffd54005860, newit=newit@entry=0x7ffd551fc8f0) at
CRoaring/roaring.c:9073
#2  0x00005555572f25b7 in roaring_create_iterator (ra=0x7ffd54005860) at CRoaring/roaring.c:9087
#3 0x00005555572eab07 in croaring::iter::BitmapIterator::new::h4e80945b49c49969
(bitmap=0x7ffd758508d8) at
/home/u/.cargo/registry/src/github.com-1ecc6299db9ec823/croaring-0.3.8/src/iter.rs:14
#4 0x00005555572eac83 in croaring::iter::_$LT$impl$u20$croaring..Bitmap$GT$::iter::hdb93dd51e075e9b5
(self=0x7ffd758508d8) at
/home/u/.cargo/registry/src/github.com-1ecc6299db9ec823/croaring-0.3.8/src/iter.rs:83
#5 0x00005555571ce9cb in grin_store::prune_list::PruneList::build_shift_cache::h662bdaf221c295c1
(self=0x7ffd758508c0) at store/src/prune_list.rs:160
#6 0x00005555571ce58e in grin_store::prune_list::PruneList::init_caches::h83b46f94f7695c94
(self=0x7ffd758508c0) at store/src/prune_list.rs:100
#7 0x000055555703505e in grin_store::prune_list::PruneList::open::h212063bb3b40e761
(path=0x7ffd75851438) at /home/u/GRIN/grin/store/src/prune_list.rs:83
#8 0x0000555556fe9936 in _$LT$grin_store..pmmr..PMMRBackend$LT$T$GT$$GT$::new::h81f278b9827293fd
(data_dir=..., prunable=true, header=...) at /home/u/GRIN/grin/store/src/pmmr.rs:219
#9 0x0000555556f68367 in
_$LT$grin_chain..txhashset..txhashset..PMMRHandle$LT$T$GT$$GT$::new::h45d8c699de9dc6a5 (root_dir=...,
sub_dir=..., file_name=..., prunable=true, header=...) at chain/src/txhashset/txhashset.rs:69
#10 0x0000555556f7eba0 in grin_chain::txhashset::txhashset::TxHashSet::open::hc3e3c6ab04000454
(root_dir=..., commit_index=..., header=...) at chain/src/txhashset/txhashset.rs:130
#11 0x0000555556f5760c in grin_chain::chain::Chain::txhashset_write::h790db0ae32cffafd
(self=0x5555581f46b0, h=..., txhashset_data=..., status=...) at chain/src/chain.rs:871
#12 0x00005555563d5ac1 in
_$LT$grin_servers..common..adapters..NetToChainAdapter$u20$as$u20$grin_p2p..types..ChainAdapter$GT$::t
xhashset_write::h0f4497da31f717d1 (self=0x5555581f4850, h=..., txhashset_data=...,
    _peer_addr=...) at servers/src/common/adapters.rs:359
```

```
#13                                    0x0000555556e9fdc2                                    in
_$LT$grin_p2p..peers..Peers$u20$as$u20$grin_p2p..types..ChainAdapter$GT$::txhashset_write::h0497557344
92d693 (self=0x5555584f79b0, h=..., txhashset_data=..., peer_addr=...) at p2p/src/peers.rs:623
#14                                    0x0000555556e803c6                                    in
_$LT$grin_p2p..peer..TrackingAdapter$u20$as$u20$grin_p2p..types..ChainAdapter$GT$::txhashset_write::h9
5f93b3fbb99844c (self=0x7ffd60000ab0, h=..., txhashset_data=..., peer_addr=...)
    at p2p/src/peer.rs:581
#15                                    0x0000555556ef0068                                    in
_$LT$grin_p2p..protocol..Protocol$u20$as$u20$grin_p2p..conn..MessageHandler$GT$::consume::hdc830875c29
c78c7 (self=0x7ffd758620c8, msg=..., writer=..., received_bytes=...)
    at p2p/src/protocol.rs:337
#16 0x0000555556ebe9e0 in grin_p2p::conn::poll::_$u7b$$u7b$closure$u7d$$u7d$::hcb003bb06128f8d7 () at
p2p/src/conn.rs:268
#17 0x0000555556f210c3 in std::sys_common::backtrace::__rust_begin_short_backtrace::h354fe58fb798d21f
(f=...) at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/sys_common/backtrace.rs:136
#18                                    0x0000555556f1f906                                    in
std::thread::Builder::spawn_unchecked::_$u7b$$u7b$closure$u7d$$u7d$::_$u7b$$u7b$closure$u7d$$u7d$::h56
a170e6183223d0 ()
    at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:477
#19                                    0x0000555556f1fa86                                    in
_$LT$std..panic..AssertUnwindSafe$LT$F$GT$$u20$as$u20$core..ops..function..FnOnce$LT$$LP$$RP$$GT$$GT$:
:call_once::h6e199459a8b512f0 (self=..., _args=0)
    at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:319
#20 0x0000555556f2128a in std::panicking::try::do_call::he59c50499165a66f (data=0x7ffd75862470 "") at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:310
#21 0x000055555787664a in __rust_maybe_catch_panic () at src/libpanic_unwind/lib.rs:102
#22      0x0000555556f21170      in      std::panicking::try::he8612998811b7ebf      (f=...)      at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:289
#23     0x0000555556f1fac6     in     std::panic::catch_unwind::h57e841abb0e53419     (f=...)     at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:398
#24                                    0x0000555556f1f6ee                                    in
std::thread::Builder::spawn_unchecked::_$u7b$$u7b$closure$u7d$$u7d$::hf4d6a7535a6deab7      ()      at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:476
#25                                    0x0000555556f1f9d8                                    in
_$LT$F$u20$as$u20$alloc..boxed..FnBox$LT$A$GT$$GT$::call_box::h586b545677055569  (self=0x7ffd60002750,
args=0) at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:673
#26          0x000055555786a20e          in          call_once<(),()>          ()          at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:683
#27 start_thread () at src/libstd/sys_common/thread.rs:24
#28        std::sys::unix::thread::Thread::new::thread_start::hca8e72c41fa9d291        ()        at
src/libstd/sys/unix/thread.rs:90
#29 0x00007ffff735b184 in start_thread () from /lib/x86_64-linux-gnu/libpthread.so.0
#30 0x00007ffff6e7203d in clone () from /lib/x86_64-linux-gnu/libc.so.6
(gdb) x/i $pc
=> 0x5555572f2524 <roaring_init_iterator+164>:  movzwl (%rax),%eax
(gdb) x/x $rax
0x0:    Cannot access memory at address 0x0
```

```
[5]
Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7ffd75a65700 (LWP 31751)]
```
roaring_bitmap_is_empty (ra=0x0) at CRoaring/roaring.c:8751
```
8751        return ra->high_low_container.size == 0;
(gdb) bt
#0  roaring_bitmap_is_empty (ra=0x0) at CRoaring/roaring.c:8751
#1                                             0x00005555571e0bb7                                    in
croaring::imp::_$LT$impl$u20$croaring..Bitmap$GT$::is_empty::hec82076e9e465164   (self=0x7ffd75a518d8)
at /home/u/.cargo/registry/src/github.com-1ecc6299db9ec823/croaring-0.3.8/src/imp.rs:872
#2    0x00005555571ce970   in   grin_store::prune_list::PruneList::build_shift_cache::h662bdaf221c295c1
(self=0x7ffd75a518c0) at store/src/prune_list.rs:155
#3      0x00005555571ce58e    in    grin_store::prune_list::PruneList::init_caches::h83b46f94f7695c94
(self=0x7ffd75a518c0) at store/src/prune_list.rs:100
```

```
#4          0x000055555703505e      in      grin_store::prune_list::PruneList::open::h212063bb3b40e761
(path=0x7ffd75a52438) at /home/u/GRIN/grin/store/src/prune_list.rs:83
#5     0x0000555556fe9936   in   _$LT$grin_store..pmmr..PMMRBackend$LT$T$GT$$GT$::new::h81f278b9827293fd
(data_dir=..., prunable=true, header=...) at /home/u/GRIN/grin/store/src/pmmr.rs:219
#6                                                0x0000555556f68367                               in
_$LT$grin_chain..txhashset..txhashset..PMMRHandle$LT$T$GT$$GT$::new::h45d8c699de9dc6a5  (root_dir=...,
sub_dir=..., file_name=..., prunable=true, header=...) at chain/src/txhashset/txhashset.rs:69
#7      0x0000555556f7eba0   in   grin_chain::txhashset::txhashset::TxHashSet::open::hc3e3c6ab04000454
(root_dir=..., commit_index=..., header=...) at chain/src/txhashset/txhashset.rs:130
#8          0x0000555556f5760c     in     grin_chain::chain::Chain::txhashset_write::h790db0ae32cffafd
(self=0x5555581f46b0, h=..., txhashset_data=..., status=...) at chain/src/chain.rs:871
#9                                                0x00005555563d5ac1                               in
_$LT$grin_servers..common..adapters..NetToChainAdapter$u20$as$u20$grin_p2p..types..ChainAdapter$GT$::t
xhashset_write::h0f4497da31f717d1 (self=0x5555581f4850, h=..., txhashset_data=...,
    _peer_addr=...) at servers/src/common/adapters.rs:359
#10                                  0x0000555556e9fdc2                                          in
_$LT$grin_p2p..peers..Peers$u20$as$u20$grin_p2p..types..ChainAdapter$GT$::txhashset_write::h0497557344
92d693 (self=0x5555584f79b0, h=..., txhashset_data=..., peer_addr=...) at p2p/src/peers.rs:623
#11                                  0x0000555556e803c6                                          in
_$LT$grin_p2p..peer..TrackingAdapter$u20$as$u20$grin_p2p..types..ChainAdapter$GT$::txhashset_write::h9
5f93b3fbb99844c (self=0x7ffd60000ab0, h=..., txhashset_data=..., peer_addr=...)
    at p2p/src/peer.rs:581
#12                                  0x0000555556ef0068                                          in
_$LT$grin_p2p..protocol..Protocol$u20$as$u20$grin_p2p..conn..MessageHandler$GT$::consume::hdc830875c29
c78c7 (self=0x7ffd75a630c8, msg=..., writer=..., received_bytes=...)
    at p2p/src/protocol.rs:337
#13  0x0000555556ebe9e0  in  grin_p2p::conn::poll::_$u7b$$u7b$closure$u7d$$u7d$::hcb003bb06128f8d7  ()  at
p2p/src/conn.rs:268
#14  0x0000555556f210c3  in  std::sys_common::backtrace::__rust_begin_short_backtrace::h354fe58fb798d21f
(f=...) at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/sys_common/backtrace.rs:136
#15                                  0x0000555556f1f906                                          in
std::thread::Builder::spawn_unchecked::_$u7b$$u7b$closure$u7d$$u7d$::_$u7b$$u7b$closure$u7d$$u7d$::h56
a170e6183223d0 ()
    at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:477
#16                                  0x0000555556f1fa86                                          in
_$LT$std..panic..AssertUnwindSafe$LT$F$GT$$u20$as$u20$core..ops..function..FnOnce$LT$$LP$$RP$$GT$$GT$:
:call_once::h6e199459a8b512f0 (self=..., _args=0)
    at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:319
#17  0x0000555556f2128a  in  std::panicking::try::do_call::he59c50499165a66f  (data=0x7ffd75a63470  "")  at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:310
#18 0x000055555787664a in __rust_maybe_catch_panic () at src/libpanic_unwind/lib.rs:102
#19       0x0000555556f21170       in       std::panicking::try::he8612998811b7ebf       (f=...)       at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:289
#20      0x0000555556f1fac6      in      std::panic::catch_unwind::h57e841abb0e53419      (f=...)      at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:398
#21                                  0x0000555556f1f6ee                                          in
std::thread::Builder::spawn_unchecked::_$u7b$$u7b$closure$u7d$$u7d$::hf4d6a7535a6deab7      ()      at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:476
#22                                  0x0000555556f1f9d8                                          in
_$LT$F$u20$as$u20$alloc..boxed..FnBox$LT$A$GT$$GT$::call_box::h586b545677055569  (self=0x7ffd60002750,
args=0) at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:673
#23            0x000055555786a20e            in            call_once<(),()>            ()            at
/rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:683
#24 start_thread () at src/libstd/sys_common/thread.rs:24
#25         std::sys::unix::thread::Thread::new::thread_start::hca8e72c41fa9d291         ()            at
src/libstd/sys/unix/thread.rs:90
#26 0x00007ffff735b184 in start_thread () from /lib/x86_64-linux-gnu/libpthread.so.0
#27 0x00007ffff6e7203d in clone () from /lib/x86_64-linux-gnu/libc.so.6
```

Coinspect believes that, given enough time, this attack vector could potentially lead to remote code execution. After Coinspect reported this initial finding to the Grin team members, they immediately started working on a series of remediations. Thus, further research was abandoned in order to focus on other components.

Recommendations

Maintain croaring-rs up to date with the latest version of the C/C++ implementation in order to keep known vulnerabilities from affecting Grin.

Consider replacing the CRoaring library with an alternative implementation. If that is not possible, Coinspect recommends a full audit of the library in order to ensure no unknown vulnerabilities exist that could lead to arbitrary code execution and/or node crashes.

Also, Coinspect suggests the establishment of a due diligence process to evaluate existing and new project dependencies.

GRIN-005 prune_list panic after processing an invalid TxHashSet leaves node unable to sync and restart

Total Risk	Impact	Location
Medium	Medium	store/src/prune_list.rs:162
Fixed	Likelihood	
Yes	High	

Description

Improper error handling during the processing of non-trusted PruneList data structures results in a panic in a peer thread, which leaves the node in an inconsistent state and unable to synchronize.

The next code is used to initialize the shift cache from an externally supplied file:

```
fn build_shift_cache(&mut self) {
  if self.bitmap.is_empty() {
    return;
  }

  self.shift_cache.clear();
  for pos in self.bitmap.iter() {
    let pos = pos as u64;
    let prev_shift = self.get_shift(pos - 1);

    let curr_shift = if self.is_pruned_root(pos) {
      let height = bintree_postorder_height(pos);
      2 * ((1 << height) - 1)
    } else {
      0
    };

    self.shift_cache.push(prev_shift + curr_shift);
  }
}
```

The above highlighted code line can be abused to trigger a subtract overflow and panic the thread.

Then, during the fast sync window, it is possible for an attacker to provide a malicious TxHashSet file and crash the peer thread:

```
20190315 22:52:23.954 ERROR grin_util::logger -
thread 'peer' panicked at 'attempt to subtract with overflow': store/src/prune_list.rs:162stack
backtrace:
   0: grin_util::logger::send_panic_to_log::{{closure}}::hcfe2f0cc5d0c7575 (0x7f443467db07)
             at util/src/logger.rs:240
   1: std::panicking::rust_panic_with_hook::h8cbdfe43764887be (0x7f4434b74079)
```

```
             at src/libstd/panicking.rs:495
   2: std::panicking::continue_panic_fmt::h3d3c5a833c00a5e1 (0x7f4434b73b21)
             at src/libstd/panicking.rs:398
   3: rust_begin_unwind (0x7f4434b73a05)
             at src/libstd/panicking.rs:325
   4: core::panicking::panic_fmt::h4d67173bc68f6d5a (0x7f4434b9038c)
             at src/libcore/panicking.rs:95
   5: core::panicking::panic::h6f50c0de2dcd7974 (0x7f4434b902bb)
             at src/libcore/panicking.rs:59
   6: grin_store::prune_list::PruneList::build_shift_cache::h662bdaf221c295c1 (0x7f44344e1c43)
             at store/src/prune_list.rs:162
   7: grin_store::prune_list::PruneList::init_caches::h83b46f94f7695c94 (0x7f44344e158d)
             at store/src/prune_list.rs:100
   8: grin_store::prune_list::PruneList::open::h212063bb3b40e761 (0x7f443434805d)
             at /home/u/GRIN/grin/store/src/prune_list.rs:83
   9: <grin_store::pmmr::PMMRBackend<T>>::new::h81f278b9827293fd (0x7f44342fc935)
             at /home/u/GRIN/grin/store/src/pmmr.rs:219
  10: <grin_chain::txhashset::txhashset::PMMRHandle<T>>::new::h45d8c699de9dc6a5 (0x7f443427b366)
             at chain/src/txhashset/txhashset.rs:69
  11: grin_chain::txhashset::txhashset::TxHashSet::open::hc3e3c6ab04000454 (0x7f4434291b9f)
             at chain/src/txhashset/txhashset.rs:130
  12: grin_chain::chain::Chain::txhashset_write::h790db0ae32cffafd (0x7f443426a60b)
             at chain/src/chain.rs:871
  13: <grin_servers::common::adapters::NetToChainAdapter as
grin_p2p::types::ChainAdapter>::txhashset_write::h0f4497da31f717d1 (0x7f44336e8ac0)
             at servers/src/common/adapters.rs:359
  14: <grin_p2p::peers::Peers as grin_p2p::types::ChainAdapter>::txhashset_write::h049755734492d693
(0x7f44341b2dc1)
             at p2p/src/peers.rs:623
  15: <grin_p2p::peer::TrackingAdapter as
grin_p2p::types::ChainAdapter>::txhashset_write::h95f93b3fbb99844c (0x7f44341933c5)
             at p2p/src/peer.rs:581
  16: <grin_p2p::protocol::Protocol as grin_p2p::conn::MessageHandler>::consume::hdc830875c29c78c7
(0x7f4434203067)
             at p2p/src/protocol.rs:337
  17: grin_p2p::conn::poll::{{closure}}::hcb003bb06128f8d7 (0x7f44341d19df)
             at p2p/src/conn.rs:268
  18: std::sys_common::backtrace::__rust_begin_short_backtrace::h354fe58fb798d21f (0x7f44342340c2)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/sys_common/backtrace.rs:136
  19: std::thread::Builder::spawn_unchecked::{{closure}}::{{closure}}::h56a170e6183223d0
(0x7f4434232905)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:477
  20: <std::panic::AssertUnwindSafe<F> as
core::ops::function::FnOnce<()>>::call_once::h6e199459a8b512f0 (0x7f4434232a85)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:319
  21: std::panicking::try::do_call::he59c50499165a66f (0x7f4434234289)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:310
  22: __rust_maybe_catch_panic (0x7f4434b89649)
             at src/libpanic_unwind/lib.rs:102
  23: std::panicking::try::he8612998811b7ebf (0x7f443423416f)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:289
  24: std::panic::catch_unwind::h57e841abb0e53419 (0x7f4434232ac5)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:398
  25: std::thread::Builder::spawn_unchecked::{{closure}}::hf4d6a7535a6deab7 (0x7f44342326ed)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:476
  26: <F as alloc::boxed::FnBox<A>>::call_box::h586b545677055569 (0x7f44342329d7)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:673
  27: <alloc::boxed::Box<(dyn alloc::boxed::FnBox<A, Output=R> + 'a)> as
core::ops::function::FnOnce<A>>::call_once::hece536cf07b94f8d (0x7f4434b7d20d)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:683
      std::sys_common::thread::start_thread::h9605a7df0f911844
             at src/libstd/sys_common/thread.rs:24
      std::sys::unix::thread::Thread::new::thread_start::hca8e72c41fa9d291
             at src/libstd/sys/unix/thread.rs:90
  28: start_thread (0x7f4431bc3183)
  29: clone (0x7f44316da03c)
  30: <unknown> (0x0)


Thread 'peer' panicked with message:
"attempt to subtract with overflow"
```

As a consequence of this panic, **the synchronization process is left in an inconsistent state that prevents the node from synchronizing with the rest of its peers. Furthermore, restarting the node is not possible**:

```
20190315 23:00:31.637 ERROR grin_util::logger -
thread 'main' panicked at 'attempt to subtract with overflow': store/src/prune_list.rs:162stack
backtrace:
   0: grin_util::logger::send_panic_to_log::{{closure}}::hcfe2f0cc5d0c7575 (0x7f28d2691b07)
             at util/src/logger.rs:240
   1: std::panicking::rust_panic_with_hook::h8cbdfe43764887be (0x7f28d2b88079)
             at src/libstd/panicking.rs:495
   2: std::panicking::continue_panic_fmt::h3d3c5a833c00a5e1 (0x7f28d2b87b21)
             at src/libstd/panicking.rs:398
   3: rust_begin_unwind (0x7f28d2b87a05)
             at src/libstd/panicking.rs:325
   4: core::panicking::panic_fmt::h4d67173bc68f6d5a (0x7f28d2ba438c)
             at src/libcore/panicking.rs:95
   5: core::panicking::panic::h6f50c0de2dcd7974 (0x7f28d2ba42bb)
             at src/libcore/panicking.rs:59
   6: grin_store::prune_list::PruneList::build_shift_cache::h662bdaf221c295c1 (0x7f28d24f5c43)
             at store/src/prune_list.rs:162
   7: grin_store::prune_list::PruneList::init_caches::h83b46f94f7695c94 (0x7f28d24f558d)
             at store/src/prune_list.rs:100
   8: grin_store::prune_list::PruneList::open::h212063bb3b40e761 (0x7f28d235c05d)
             at /home/u/GRIN/grin/store/src/prune_list.rs:83
   9: <grin_store::pmmr::PMMRBackend<T>>::new::h81f278b9827293fd (0x7f28d2310935)
             at /home/u/GRIN/grin/store/src/pmmr.rs:219
  10: <grin_chain::txhashset::txhashset::PMMRHandle<T>>::new::h45d8c699de9dc6a5 (0x7f28d228f366)
             at chain/src/txhashset/txhashset.rs:69
  11: grin_chain::txhashset::txhashset::TxHashSet::open::hc3e3c6ab04000454 (0x7f28d22a5b9f)
             at chain/src/txhashset/txhashset.rs:130
  12: grin_chain::chain::Chain::init::h8297f25047c202cd (0x7f28d2272582)
             at chain/src/chain.rs:183
  13: grin_servers::grin::server::Server::new::h112171a8cdbbe207 (0x7f28d16add5b)
             at servers/src/grin/server.rs:179
  14: grin_servers::grin::server::Server::start::h124d013591b00388 (0x7f28d1160246)
             at /home/u/GRIN/grin/servers/src/grin/server.rs:83
  15: grin::cmd::server::start_server_tui::h9b979ce79f882862 (0x7f28d124157e)
             at src/bin/cmd/server.rs:62
  16: grin::cmd::server::start_server::h4927356618c1e453 (0x7f28d1241054)
             at src/bin/cmd/server.rs:33
  17: grin::cmd::server::server_command::he33efd3028c213d0 (0x7f28d1241cff)
             at src/bin/cmd/server.rs:145
  18: grin::real_main::hc73dff343620e9d9 (0x7f28d1059267)
             at src/bin/grin.rs:193
  19: grin::main::hfb66e9a2739dc55d (0x7f28d1057b15)
             at src/bin/grin.rs:68
  20: std::rt::lang_start::{{closure}}::h0aaed896f4204d80 (0x7f28d12087ff)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/rt.rs:74
```

```
21: std::rt::lang_start_internal::{{closure}}::hdfc28107b5be47c9 (0x7f28d2b879a2)
           at src/libstd/rt.rs:59
    std::panicking::try::do_call::h69790245ac2d03fe
           at src/libstd/panicking.rs:310
22: __rust_maybe_catch_panic (0x7f28d2b9d649)
           at src/libpanic_unwind/lib.rs:102
23: std::panicking::try::h9c1cbc5599e1efbf (0x7f28d2b88373)
           at src/libstd/panicking.rs:289
    std::panic::catch_unwind::h0562757d03ff60b3
           at src/libstd/panic.rs:398
    std::rt::lang_start_internal::h540c897fe52ba9c5
           at src/libstd/rt.rs:58
24: std::rt::lang_start::h144219d38079d061 (0x7f28d12087d8)
           at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/rt.rs:74
25: main (0x7f28d10599e9)
26: __libc_start_main (0x7f28cf611f44)
27: <unknown> (0x7f28d1054c8c)
28: <unknown> (0x0)
```

Recommendations

In order to prevent panics, handle malformed prune lists gracefully.

GRIN-006 Disk space DoS via TxHashSetRequest p2p messages

Total Risk	Impact	Location
High	Medium	p2p/src/protocol.rs:247 chain/txhashset/txhashset.rs:1413 chain/src/chain.rs:676
Fixed	Likelihood	
Yes	High	

Description

Grin nodes answer TxHashSetRequest p2p protocol requests with a ZIP encoded file containing data for three MMRs (output, rangeproof, and kernel). Each time, the target node creates a new random name directory to copy the files required and creates a new ZIP encoded file with this directory content; neither this directory nor the archive file is deleted from the disk after the request has been served. Then, for each TxHashSetRequest processed, the size of the current txhashset directory multiplied by two (as ZIP compression is not supported) is added to the filesystem.

Because these messages can be sent from any peer at any time, one or more attacking nodes can keep making requests until the target node filesystem is full, making it unable to add new blocks to the storage layer and forcing it out of the network.

The txhashset_read function is used to respond to the TxHashSetRequest p2p protocol message:

```
/// Provides a reading view into the current txhashset state as well as
/// the required indexes for a consumer to rewind to a consistent state
/// at the provided block hash.
pub fn txhashset_read(&self, h: Hash) -> Result<(u64, u64, File), Error> {
  // now we want to rewind the txhashset extension and
  // sync a "rewound" copy of the leaf_set files to disk
  // so we can send these across as part of the zip file.
  // The fast sync client does *not* have the necessary data
  // to rewind after receiving the txhashset zip.
  let header = self.get_block_header(&h)?;
  {
    let mut txhashset = self.txhashset.write();
    txhashset::extending_readonly(&mut txhashset, |extension| {
      extension.rewind(&header)?;
      extension.snapshot()?;
      Ok(())
    })?;
  }

  // prepares the zip and return the corresponding Read
  let txhashset_reader = txhashset::zip_read(self.db_root.clone(), &header, None)?;
  Ok((
    header.output_mmr_size,
    header.kernel_mmr_size,
    txhashset_reader,
  ))
}
```

It relies on the zip_read function:

```
/// Packages the txhashset data files into a zip and returns a Read to the
/// resulting file
pub fn zip_read(root_dir: String, header: &BlockHeader, rand: Option<u32>) -> Result<File, Error> {
  let ts = if let None = rand {
    let now = SystemTime::now();
    now.duration_since(UNIX_EPOCH).unwrap().subsec_micros()
  } else {
    rand.unwrap()
  };
  let txhashset_zip = format!("{}_{}.zip", TXHASHSET_ZIP, ts);

  let txhashset_path = Path::new(&root_dir).join(TXHASHSET_SUBDIR);
  let zip_path = Path::new(&root_dir).join(txhashset_zip);
  // create the zip archive
  {
    // Temp txhashset directory
    let temp_txhashset_path =
      Path::new(&root_dir).join(format!("{}_zip_{}", TXHASHSET_SUBDIR, ts));
    // Remove temp dir if it exist
    if temp_txhashset_path.exists() {
      fs::remove_dir_all(&temp_txhashset_path)?;
    }
    // Copy file to another dir
    file::copy_dir_to(&txhashset_path, &temp_txhashset_path)?;
    // Check and remove file that are not supposed to be there
    check_and_remove_files(&temp_txhashset_path, header)?;
    // Compress zip
    zip::compress(&temp_txhashset_path, &File::create(zip_path.clone())?)
      .map_err(|ze| ErrorKind::Other(ze.to_string()))?;
  }

  // open it again to read it back
  let zip_file = File::open(zip_path)?;

  Ok(zip_file)
}
```

This is how the chain_data directory looks after processing 345 messages from an attacking node. The target node in this example is at height 2575 (each temporary directory and its corresponding ZIP archive size is 3.4M):

```
du -hs chain_data
2.3G    chain_data
```

```
grin.lock                       txhashset_snapshot_343086.zip  txhashset_snapshot_618941.zip
txhashset_snapshot_874587.zip  txhashset_zip_222936  txhashset_zip_504611  txhashset_zip_76014
header                          txhashset_snapshot_344645.zip  txhashset_snapshot_61959.zip
txhashset_snapshot_875713.zip  txhashset_zip_223005  txhashset_zip_51145   txhashset_zip_760190
lmdb                            txhashset_snapshot_346062.zip  txhashset_snapshot_621014.zip
txhashset_snapshot_881453.zip  txhashset_zip_225128  txhashset_zip_51422   txhashset_zip_76210
peer                            txhashset_snapshot_350446.zip  txhashset_snapshot_6225.zip
txhashset_snapshot_884378.zip  txhashset_zip_225352  txhashset_zip_514421  txhashset_zip_764025
```

```
txhashset                     txhashset_snapshot_356494.zip  txhashset_snapshot_628277.zip
txhashset_snapshot_884528.zip  txhashset_zip_225965  txhashset_zip_519044  txhashset_zip_765769
txhashset_snapshot_10150.zip   txhashset_snapshot_358810.zip  txhashset_snapshot_633683.zip
txhashset_snapshot_886489.zip  txhashset_zip_227119  txhashset_zip_519645  txhashset_zip_7659
txhashset_snapshot_110109.zip  txhashset_snapshot_365703.zip  txhashset_snapshot_636054.zip
txhashset_snapshot_88852.zip   txhashset_zip_227390  txhashset_zip_520277  txhashset_zip_767998
txhashset_snapshot_112067.zip  txhashset_snapshot_368054.zip  txhashset_snapshot_636570.zip
txhashset_snapshot_892970.zip  txhashset_zip_232249  txhashset_zip_522283  txhashset_zip_771570
txhashset_snapshot_11289.zip   txhashset_snapshot_375492.zip  txhashset_snapshot_644590.zip
txhashset_snapshot_893106.zip  txhashset_zip_234840  txhashset_zip_527956  txhashset_zip_776577
txhashset_snapshot_113011.zip  txhashset_snapshot_381684.zip  txhashset_snapshot_644790.zip
txhashset_snapshot_893807.zip  txhashset_zip_234951  txhashset_zip_528304  txhashset_zip_777464
txhashset_snapshot_114989.zip  txhashset_snapshot_386513.zip  txhashset_snapshot_647225.zip
txhashset_snapshot_903033.zip  txhashset_zip_239580  txhashset_zip_530546  txhashset_zip_780869
txhashset_snapshot_115573.zip  txhashset_snapshot_392350.zip  txhashset_snapshot_648533.zip
txhashset_snapshot_903276.zip  txhashset_zip_24996   txhashset_zip_53093   txhashset_zip_787130
txhashset_snapshot_118292.zip  txhashset_snapshot_393542.zip  txhashset_snapshot_648961.zip
txhashset_snapshot_904710.zip  txhashset_zip_251782  txhashset_zip_533841  txhashset_zip_78838
txhashset_snapshot_1238.zip    txhashset_snapshot_395386.zip  txhashset_snapshot_649420.zip

[...]

txhashset_zip_18275           txhashset_zip_469649  txhashset_zip_732162  txhashset_zip_969265
txhashset_snapshot_313078.zip  txhashset_snapshot_591485.zip  txhashset_snapshot_836030.zip
txhashset_zip_18340           txhashset_zip_472236  txhashset_zip_736689  txhashset_zip_969365
txhashset_snapshot_315843.zip  txhashset_snapshot_597416.zip  txhashset_snapshot_838647.zip
txhashset_zip_184087          txhashset_zip_473804  txhashset_zip_738919  txhashset_zip_974185
txhashset_snapshot_316302.zip  txhashset_snapshot_597752.zip  txhashset_snapshot_838751.zip
txhashset_zip_186659          txhashset_zip_477702  txhashset_zip_742051  txhashset_zip_974775
txhashset_snapshot_316620.zip  txhashset_snapshot_597864.zip  txhashset_snapshot_842302.zip
txhashset_zip_187249          txhashset_zip_479135  txhashset_zip_744402  txhashset_zip_975542
txhashset_snapshot_316954.zip  txhashset_snapshot_597947.zip  txhashset_snapshot_843973.zip
txhashset_zip_193982          txhashset_zip_479416  txhashset_zip_74568   txhashset_zip_982205
txhashset_snapshot_319836.zip  txhashset_snapshot_601360.zip  txhashset_snapshot_845694.zip
txhashset_zip_202382          txhashset_zip_480288  txhashset_zip_749399  txhashset_zip_98351
txhashset_snapshot_32798.zip   txhashset_snapshot_602038.zip  txhashset_snapshot_847955.zip
txhashset_zip_20446           txhashset_zip_487557  txhashset_zip_750486  txhashset_zip_995163
txhashset_snapshot_328222.zip  txhashset_snapshot_604552.zip  txhashset_snapshot_848217.zip
txhashset_zip_206282          txhashset_zip_49640   txhashset_zip_751863  txhashset_zip_997234
txhashset_snapshot_329378.zip  txhashset_snapshot_60487.zip   txhashset_snapshot_849913.zip
txhashset_zip_208661          txhashset_zip_496660  txhashset_zip_753407
txhashset_snapshot_332572.zip  txhashset_snapshot_605326.zip  txhashset_snapshot_85797.zip
txhashset_zip_211740          txhashset_zip_497034  txhashset_zip_753811
txhashset_snapshot_33829.zip   txhashset_snapshot_612983.zip  txhashset_snapshot_864324.zip
txhashset_zip_213215          txhashset_zip_499086  txhashset_zip_755787
txhashset_snapshot_338715.zip  txhashset_snapshot_613098.zip  txhashset_snapshot_86986.zip
txhashset_zip_217148          txhashset_zip_500264  txhashset_zip_756792
txhashset_snapshot_340745.zip  txhashset_snapshot_615890.zip  txhashset_snapshot_871823.zip
txhashset_zip_217449          txhashset_zip_501975  txhashset_zip_757945
```

It is worth noting that Grin implements a rate-limiting mechanism (peer.rs and rate_counter.rs) that considers a peer abusive when it sends messages per minute above a certain threshold (MAX_PEER_MSG_PER_MIN, currently 500).

As the moment of the writing of this report, the size of the txhashset directory for a mainnet node is ~166M, so each processed message will result in an additional ~332M in the filesystem, resulting in a theoretical maximum of new 162G stored per minute per attacking host (limited by network bandwidth).

Recommendations

Remove temporary `txhashset` directories right after the ZIP archive has been created. Then remove the temporary ZIP archive after the node finishes serving the request.

Consider decreasing the current rate limit for this particular message.

GRIN-007 Nodes can be indefinitely prevented from synchronizing the blockchain via unsolicited TxHashSetArchive p2p messages

Total Risk	Impact	Location
High	Medium	servers/src/common/adapters.rs:315 p2p/src/protocol.rs:275 chain/src/chain.rs:859
Fixed	Likelihood	
Yes	High	

Description

Lack of validation during the state synchronization process enables remote attackers to prevent a node from ever finishing the process.

Grin nodes only accept TxHashSetArchive messages from their peers when their synchronization status is TxHashsetDownload as checked in the following code:

```
fn txhashset_receive_ready(&self) -> bool {
  match self.sync_state.status() {
    SyncStatus::TxHashsetDownload { .. } => true,
    _ => false,
  }
}
```

The txhashset_receive_ready function is called from the code in charge of processing a received TxHashSetArchive message:

```
Type::TxHashSetArchive => {
  let sm_arch: TxHashSetArchive = msg.body()?;
  debug!(
    "handle_payload: txhashset archive for {} at {}. size={}",
    sm_arch.hash, sm_arch.height, sm_arch.bytes,
  );

  if !self.adapter.txhashset_receive_ready() {
    error!(
      "handle_payload: txhashset archive received but SyncStatus not on TxHashsetDownload",
    );
    return Err(Error::BadMessage);
  }

  let download_start_time = Utc::now();
  self.adapter
    .txhashset_download_update(download_start_time, 0, sm_arch.bytes);

  let mut tmp = env::temp_dir();
  tmp.push("txhashset.zip");


  let mut save_txhashset_to_file = |file| -> Result<(), Error> {
    let mut tmp_zip = BufWriter::new(File::create(file)?);
    let total_size = sm_arch.bytes as usize;
```

```
        let mut downloaded_size: usize = 0;
        let mut request_size = cmp::min(48_000, total_size);
        while request_size > 0 {
            let size = msg.copy_attachment(request_size, &mut tmp_zip)?;
            downloaded_size += size;
            request_size = cmp::min(48_000, total_size - downloaded_size);
            self.adapter.txhashset_download_update(
                download_start_time,
                downloaded_size as u64,
                total_size as u64,
            );

            // Increase received bytes quietly (without affecting the counters).
            // Otherwise we risk banning a peer as "abusive".
            {
                let mut received_bytes = received_bytes.write();
                received_bytes.inc_quiet(size as u64);
            }
        }
        tmp_zip
            .into_inner()
            .map_err(|_| Error::Internal)?
            .sync_all()?;
        Ok(())
    };

    if let Err(e) = save_txhashset_to_file(tmp.clone()) {
        error!(
            "handle_payload: txhashset archive save to file fail. err={:?}",
            e
        );
        return Err(e);
    }
```

However, there is no check to guarantee the sending peer is the one from which the hashset was requested.

As a consequence, malicious nodes can send unsolicited TxHashsetArchive messages with a non-existent hash value (while a node is synchronizing the blockchain with a well-behaved node) and interrupt the process:

```
pub fn txhashset_write(
    &self,
    h: Hash,
    txhashset_data: File,
    status: &dyn TxHashsetWriteStatus,
) -> Result<(), Error> {
    status.on_setup();

    // Initial check whether this txhashset is needed or not
    let mut hashes: Option<Vec<Hash>> = None;
    if !self.check_txhashset_needed("txhashset_write".to_owned(), &mut hashes) {
        warn!("txhashset_write: txhashset received but it's not needed! ignored.");
        return Err(ErrorKind::InvalidTxHashSet("not needed".to_owned()).into());
    }

    let header = self.get_block_header(&h)?;
```

This is what an attack would look like:

1. The target node starts, connects to seed nodes.
2. The target node picks a random node from the ones in the most-work list; it synchronizes headers.
3. The target node requests txhashset from a random node in the most-work list.
4. The well-behaved node answers the requests with a txhashset archive.
5. While this download is taking place, an evil node discovers the new node and *sends a* TxHashSetArchive *message with an invalid hash value*.
6. The target node tries to process the attacker's message but fails as the invalid hash provided is not found in the storage. The target node decides to restart the state sync process. This takes the process back to point 3.

The following log shows an attack taking place:

1. The target node requests the state archive from the benign node at `127.0.0.1:3414.`

```
20190402 11:56:03.766 DEBUG grin_chain::chain - body_sync: body_head - 32c1193af373, 0, header_head -
cf57210e01ab, 2583, sync_head - cf57210e01ab, 2583
20190402 11:56:04.404 DEBUG grin_chain::chain - body_sync: need a state sync for txhashset. oldest
block which is not on local chain: d456d242ca3c at 1
20190402 11:56:04.404 DEBUG grin_servers::grin::sync::body_sync - body_sync: cannot sync full blocks
earlier than horizon. will request txhashset
20190402 11:56:04.407 DEBUG grin_servers::grin::sync::state_sync - state_sync: before txhashset
request, header head: 2583 / cf57210e01ab, txhashset_head: 2562 / 53c52127a7f4
20190402 11:56:04.407 DEBUG grin_p2p::peer - Asking 127.0.0.1:3414 for txhashset archive at 2562
53c52127a7f4.
20190402 11:56:04.408 DEBUG grin_servers::common::types - sync_state: sync_status: HeaderSync {
current_height: 2560, highest_height: 2583 } -> HeaderSync { current_height: 2583, highest_height:
2583 }
20190402 11:56:04.408 DEBUG grin_servers::common::types - sync_state: sync_status: HeaderSync {
current_height: 2583, highest_height: 2583 } -> TxHashsetDownload { start_time:
2019-04-02T14:56:04.408106596Z, prev_update_time: 2019-04-02T14:56:04.408108177Z, update_time:
2019-04-02T14:56:04.408108564Z, prev_downloaded_size: 0, downloaded_size: 0, total_size: 0 }
20190402 11:56:05.009 DEBUG grin_p2p::protocol - handle_payload: txhashset archive for 53c52127a7f4 at
2562. size=5273845535
```

2. Then an evil node at `192.168.1.117:5414` connects and sends an unsolicited txhashset message with an invalid hash value: `000000000000`

```
20190402 11:56:38.793 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.117:5414))
20190402 11:56:41.575 DEBUG grin_p2p::protocol - handle_payload: txhashset archive for 000000000000 at
123. size=622
20190402 11:56:41.695 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetDownload {
start_time: 2019-04-02T14:56:05.010154764Z, prev_update_time: 2019-04-02T14:56:41.686224813Z,
update_time: 2019-04-02T14:56:41.691984710Z, prev_downloaded_size: 319296000, downloaded_size:
319344000, total_size: 5273845535 } -> TxHashsetSetup
20190402 11:56:41.695 DEBUG grin_chain::chain - txhashset_write: body_head - 32c1193af373, 0,
header_head - cf57210e01ab, 2583, sync_head - cf57210e01ab, 2583
```

```
20190402 11:56:42.678 DEBUG grin_chain::chain - txhashset_write: need a state sync for txhashset.
oldest block which is not on local chain: d456d242ca3c at 1
20190402 11:56:42.678 ERROR grin_servers::common::adapters - Failed to save txhashset archive: Store
Error: chain get header, reason: DB Not Found Error: BLOCK HEADER: 000000000000
 Cause: Unknown
 Backtrace:
20190402 11:56:42.679 DEBUG grin_p2p::protocol - handle_payload: txhashset archive for 000000000000 at
 123, DONE. Data Ok: true
```

3. The malformed archive fails, and the state synchronization process is restarted.

```
20190402 11:56:42.688 ERROR grin_servers::grin::sync::state_sync - state_sync: error = Chain(Error {
inner:

Store Error: chain get header, reason: DB Not Found Error: BLOCK HEADER: 000000000000 }). restart fast
sync
20190402 11:56:42.696 DEBUG grin_servers::grin::sync::state_sync - state_sync: before txhashset
request, header head: 2583 / cf57210e01ab, txhashset_head: 2562 / 53c52127a7f4
20190402 11:56:42.696 DEBUG grin_p2p::peer - Asking 127.0.0.1:3414 for txhashset archive at 2562
53c52127a7f4.
20190402 11:56:42.696 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetSetup ->
TxHashsetDownload { start_time: 2019-04-02T14:56:42.696843063Z, prev_update_time:
2019-04-02T14:56:42.696845973Z, update_time: 2019-04-02T14:56:42.696846773Z, prev_downloaded_size: 0,
downloaded_size: 0, total_size: 0 }
```

4. And again . . .

```
20190402 11:57:11.587 DEBUG grin_p2p::protocol - handle_payload: txhashset archive for 000000000000 at
123. size=622
20190402 11:57:11.726 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetDownload {
start_time: 2019-04-02T14:56:05.010154764Z, prev_update_time: 2019-04-02T14:57:11.719278712Z,
update_time: 2019-04-02T14:57:11.723421935Z, prev_downloaded_size: 543504000, downloaded_size:
543552000, total_size: 5273845535 } -> TxHashsetSetup
20190402 11:57:11.726 DEBUG grin_chain::chain - txhashset_write: body_head - 32c1193af373, 0,
header_head - cf57210e01ab, 2583, sync_head - cf57210e01ab, 2583
20190402 11:57:12.886 DEBUG grin_chain::chain - txhashset_write: need a state sync for txhashset.
oldest block which is not on local chain: d456d242ca3c at 1
20190402 11:57:12.886 ERROR grin_servers::common::adapters - Failed to save txhashset archive: Store
Error: chain get header, reason: DB Not Found Error: BLOCK HEADER: 000000000000
 Cause: Unknown
 Backtrace:
20190402 11:57:12.887 DEBUG grin_p2p::protocol - handle_payload: txhashset archive for 000000000000 at
123, DONE. Data Ok: true
20190402 11:57:12.890 ERROR grin_servers::grin::sync::state_sync - state_sync: error = Chain(Error {
inner:

Store Error: chain get header, reason: DB Not Found Error: BLOCK HEADER: 000000000000 }). restart fast
sync
20190402 11:57:12.897 DEBUG grin_servers::grin::sync::state_sync - state_sync: before txhashset
request, header head: 2583 / cf57210e01ab, txhashset_head: 2562 / 53c52127a7f4
20190402 11:57:12.897 DEBUG grin_p2p::peer - Asking 127.0.0.1:3414 for txhashset archive at 2562
53c52127a7f4.
20190402 11:57:12.897 ERROR grin_servers::grin::sync::state_sync - state_sync: send_txhashset_request
err! Send("sending on a full channel")
20190402 11:57:12.897 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetSetup ->
TxHashsetDownload { start_time: 2019-04-02T14:57:12.897875223Z, prev_update_time:
2019-04-02T14:57:12.897877826Z, update_time: 2019-04-02T14:57:12.897878482Z, prev_downloaded_size: 0,
downloaded_size: 0, total_size: 0 }
20190402 11:57:12.908 ERROR grin_servers::grin::sync::state_sync - state_sync: error =
P2P(Send("sending on a full channel")). restart fast sync
```

```
20190402 11:57:12.912 DEBUG grin_servers::grin::sync::state_sync - state_sync: before txhashset
request, header head: 2583 / cf57210e01ab, txhashset_head: 2562 / 53c52127a7f4
20190402 11:57:12.912 DEBUG grin_p2p::peer - Asking 127.0.0.1:3414 for txhashset archive at 2562
53c52127a7f4.
20190402 11:57:12.913 ERROR grin_servers::grin::sync::state_sync - state_sync: send_txhashset_request
err! Send("sending on a full channel")
20190402 11:57:12.913 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetDownload {
start_time: 2019-04-02T14:56:05.010154764Z, prev_update_time: 2019-04-02T14:57:12.907223002Z,
update_time: 2019-04-02T14:57:12.911900747Z, prev_downloaded_size: 551568000, downloaded_size:
551616000, total_size: 5273845535 } -> TxHashsetDownload { start_time: 2019-04-02T14:57:12.913317745Z,
prev_update_time: 2019-04-02T14:57:12.913319807Z, update_time: 2019-04-02T14:57:12.913320260Z,
prev_downloaded_size: 0, downloaded_size: 0, total_size: 0 }
20190402 11:57:12.924 ERROR grin_servers::grin::sync::state_sync - state_sync: error =
P2P(Send("sending on a full channel")). restart fast sync
20190402 11:57:12.929 DEBUG grin_servers::grin::sync::state_sync - state_sync: before txhashset
request, header head: 2583 / cf57210e01ab, txhashset_head: 2562 / 53c52127a7f4
20190402 11:57:12.929 DEBUG grin_p2p::peer - Asking 127.0.0.1:3414 for txhashset archive at 2562
53c52127a7f4.
20190402 11:57:12.930 ERROR grin_servers::grin::sync::state_sync - state_sync: send_txhashset_request
err! Send("sending on a full channel")
20190402 11:57:12.930 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetDownload {
start_time: 2019-04-02T14:56:05.010154764Z, prev_update_time: 2019-04-02T14:57:12.917250369Z,
update_time: 2019-04-02T14:57:12.928000284Z, prev_downloaded_size: 551664000, downloaded_size:
551712000, total_size: 5273845535 } -> TxHashsetDownload { start_time: 2019-04-02T14:57:12.930145533Z,
prev_update_time: 2019-04-02T14:57:12.930148475Z, update_time: 2019-04-02T14:57:12.930149177Z,
prev_downloaded_size: 0, downloaded_size: 0, total_size: 0 }
20190402 11:57:12.940 ERROR grin_servers::grin::sync::state_sync - state_sync: error =
P2P(Send("sending on a full channel")). restart fast sync
20190402 11:57:12.945 DEBUG grin_servers::grin::sync::state_sync - state_sync: before txhashset
request, header head: 2583 / cf57210e01ab, txhashset_head: 2562 / 53c52127a7f4
20190402 11:57:12.945 DEBUG grin_p2p::peer - Asking 127.0.0.1:3414 for txhashset archive at 2562
53c52127a7f4.
20190402 11:57:12.945 ERROR grin_servers::grin::sync::state_sync - state_sync: send_txhashset_request
err! Send("sending on a full channel")
20190402 11:57:12.945 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetDownload {
start_time: 2019-04-02T14:56:05.010154764Z, prev_update_time: 2019-04-02T14:57:12.934590607Z,
update_time: 2019-04-02T14:57:12.940368583Z, prev_downloaded_size: 551760000, downloaded_size:
551808000, total_size: 5273845535 } -> TxHashsetDownload { start_time: 2019-04-02T14:57:12.945777952Z,
prev_update_time: 2019-04-02T14:57:12.945780213Z, update_time: 2019-04-02T14:57:12.945780754Z,
prev_downloaded_size: 0, downloaded_size: 0, total_size: 0 }
```

Recommendations

Coinspect recommends that nodes only accept TxHashsetArchive messages from the peer they originally requested it from. Also, ban nodes that send unsolicited and/or out-of-sequence TxHashsetArchive messages.

GRIN-008 Insecure file handling local privilege escalation

Total Risk	Impact	Location
Medium	High	p2p/src/protocol.rs:247 chain/txhashset/txhashset.rs:1413 chain/src/chain.rs:676
Fixed	Likelihood	
Yes	Low	

Description

The Grin node insecurely creates a file in a predictable location writable by any user in the system.

Any user of the host running the node can create a symbolic link in /tmp/txhashset.zip to any file writable by the Grin node.

The code below, located in protocol.rs, is used to process TxHashSetArchive messages from the p2p network:

```
Type::TxHashSetArchive => {
  let sm_arch: TxHashSetArchive = msg.body()?;
  debug!(
    "handle_payload: txhashset archive for {} at {}. size={}",
    sm_arch.hash, sm_arch.height, sm_arch.bytes,
  );

  if !self.adapter.txhashset_receive_ready() {
    error!(
      "handle_payload: txhashset archive received but SyncStatus not on TxHashsetDownload",
    );
    return Err(Error::BadMessage);
  }

  let download_start_time = Utc::now();
  self.adapter
    .txhashset_download_update(download_start_time, 0, sm_arch.bytes);

  let mut tmp = env::temp_dir();
  tmp.push("txhashset.zip");
  let mut save_txhashset_to_file = |file| -> Result<(), Error> {
    let mut tmp_zip = BufWriter::new(File::create(file)?);
    let total_size = sm_arch.bytes as usize;
    let mut downloaded_size: usize = 0;
    let mut request_size = cmp::min(48_000, total_size);
    while request_size > 0 {
      let size = msg.copy_attachment(request_size, &mut tmp_zip)?;
      downloaded_size += size;
      request_size = cmp::min(48_000, total_size - downloaded_size);
      self.adapter.txhashset_download_update(
        download_start_time,
        downloaded_size as u64,
        total_size as u64,
      );
```

```
        // Increase received bytes quietly (without affecting the counters).
        // Otherwise we risk banning a peer as "abusive".
        {
          let mut received_bytes = received_bytes.write();
          received_bytes.inc_quiet(size as u64);
        }
      }
      tmp_zip
        .into_inner()
        .map_err(|_| Error::Internal)?
        .sync_all()?;
      Ok(())
    };

    if let Err(e) = save_txhashset_to_file(tmp.clone()) {
      error!(
        "handle_payload: txhashset archive save to file fail. err={:?}",
        e
      );
      return Err(e);
    }
```

File::Create truncates existing files and follows symbolic links.

Then, during the fast sync window, an attacker with an account on the host running the node can provide a malicious TxHashSet file and overwrite any file the node has write permissions to.

For example, node configuration files could be overwritten to make a miner use the attacker's wallet as the coinbase target, or if the wallet is running in the same host (and owned by the same account), the seed could be overwritten and lost.

Recommendations

- Randomize temporary file names.
- Only write to files in directories owned and writable by the node user.
- Check that the target file is not a symbolic link.

Coinspect suggests using the create_new option from std::fs::OpenOptions, which guarantees the target file does not exist or is a symbolic link, in an atomic way, to prevent race conditions as documented in https://doc.rust-lang.org/std/fs/struct.OpenOptions.html.

More information regarding atomic actions in the filesystem can be found at http://tldp.org/HOWTO/Secure-Programs-HOWTO/avoid-race.html.

GRIN-009 Nodes can be tricked into banning well-behaved peers (temporary file shared among peer threads)

Total Risk	Impact	Location
High	Medium	p2p/src/protocol.rs:296
Fixed	Likelihood	
Yes	High	

Description

Reuse of a temporary file between peer threads during the state synchronization process enables remote attackers to abuse the peer-banning mechanism to ban well-behaved peers.

This is the code in charge of processing TxHashSetArchive messages:

```
Type::TxHashSetArchive => {
 let sm_arch: TxHashSetArchive = msg.body()?;
 debug!(
   "handle_payload: txhashset archive for {} at {}. size={}",
   sm_arch.hash, sm_arch.height, sm_arch.bytes,
 );

 if !self.adapter.txhashset_receive_ready() {
   error!(
     "handle_payload: txhashset archive received but SyncStatus not on TxHashsetDownload",
   );
   return Err(Error::BadMessage);
 }

 let download_start_time = Utc::now();
 self.adapter
   .txhashset_download_update(download_start_time, 0, sm_arch.bytes);

 let mut tmp = env::temp_dir();
 tmp.push("txhashset.zip");


 let mut save_txhashset_to_file = |file| -> Result<(), Error> {
   let mut tmp_zip = BufWriter::new(File::create(file)?);
   let total_size = sm_arch.bytes as usize;
   let mut downloaded_size: usize = 0;
   let mut request_size = cmp::min(48_000, total_size);
   while request_size > 0 {
     let size = msg.copy_attachment(request_size, &mut tmp_zip)?;
     downloaded_size += size;
     request_size = cmp::min(48_000, total_size - downloaded_size);
     self.adapter.txhashset_download_update(
       download_start_time,
       downloaded_size as u64,
       total_size as u64,
     );

     // Increase received bytes quietly (without affecting the counters).
     // Otherwise we risk banning a peer as "abusive".
```

```
      {
        let mut received_bytes = received_bytes.write();
        received_bytes.inc_quiet(size as u64);
      }
    }
    tmp_zip
      .into_inner()
      .map_err(|_| Error::Internal)?
      .sync_all()?;
    Ok(())
  };

  if let Err(e) = save_txhashset_to_file(tmp.clone()) {
    error!(
      "handle_payload: txhashset archive save to file fail. err={:?}",
      e
    );
    return Err(e);
  }
```

The same file path is used to store txhashset archives, which means that when a second peer thread processes a message from a malicious node, the file gets overwritten.

Malicious nodes can send unsolicited TxHashsetArchive messages while a node is synchronizing the blockchain with a well-behaved node. **As a consequence, the downloaded archives get corrupted, resulting in the innocent node being banned.**

This is what an attack would look like:

1. The target node starts and connects to seed nodes.
2. The target node picks a random node from the ones in the most-work list; it synchronizes headers.
3. The target node requests txhashset from a random node in the most-work list.
4. The well-behaved node answers the requests with a txhashset archive.
5. While this download is taking place, an evil node discovers the new node and sends a TxHashSetArchive message with random data.
6. The target node processes the evil node message, accepts the new txhashset archive, and overwrites the one being downloaded from the well-behaved node.
7. When the archive from the well-behaved node finishes downloading, the target node tries to process what now is a corrupted txhashset archive, finds out it is broken, and decides to restart the state sync process.
8. The well-behaved node gets banned.

The following log shows an attack taking place:

1. The target node requests a txhashset archive from a well-behaved peer at `127.0.0.1:3414.`

```
20190404 09:52:00.531 DEBUG grin_chain::chain - body_sync: need a state sync for txhashset. oldest
block which is not on local chain: d456d242ca3c88e1c9d325cb6f9a8407 at 1
```

```
20190404 09:52:00.531 DEBUG grin_servers::grin::sync::body_sync - body_sync: cannot sync full blocks
earlier than horizon. will request txhashset
20190404 09:52:00.534 DEBUG grin_servers::grin::sync::state_sync - state_sync: before txhashset
request, header head: 2583 / cf57210e01ab0ba31aca2f8b827f13d6, txhashset_head: 2562 /
53c52127a7f430da079a9b5ca59f61d8
20190404 09:52:00.534 DEBUG grin_p2p::peer - Asking 127.0.0.1:3414 for txhashset archive at 2562
53c52127a7f430da079a9b5ca59f61d8.
20190404 09:52:00.534 DEBUG grin_servers::common::types - sync_state: sync_status: HeaderSync {
current_height: 2583, highest_height: 2583 } -> TxHashsetDownload { start_time:
2019-04-04T12:52:00.534932853Z, prev_update_time: 2019-04-04T12:52:00.534934350Z, update_time:
2019-04-04T12:52:00.534934668Z, prev_downloaded_size: 0, downloaded_size: 0, total_size: 0 }
20190404 09:52:01.177 DEBUG grin_p2p::protocol - handle_payload: txhashset archive for
53c52127a7f430da079a9b5ca59f61d8 at 2562. size=3469942
20190404 09:52:01.177 DEBUG grin_p2p::protocol - CUCO: re-creating /tmp/txhashset.zip
20190404 09:52:17.829 DEBUG grin_servers::grin::seed - monitor_peers: on 0.0.0.0:4414, 1 connected (1
most_work). all 28 = 4 healthy + 1 banned + 23 defunct
```

2. An evil peer (127.0.0.1:5414) connects to our target node and sends an unsolicited txhashset archive. Our target node fails to process it properly as it gets corrupted:

```
20190404 09:52:18.959 DEBUG grin_p2p::peers - Saving newly connected peer 127.0.0.1:5414.
20190404 09:52:18.959 DEBUG grin_p2p::store - save_peer: PeerAddr(V4(127.0.0.1:5414)) marked Healthy
20190404 09:53:06.293 DEBUG grin_p2p::peers - Saving newly connected peer 127.0.0.1:5414.
20190404 09:53:06.294 DEBUG grin_p2p::store - save_peer: PeerAddr(V4(127.0.0.1:5414)) marked Healthy
20190404 09:53:06.483 DEBUG grin_p2p::protocol - handle_payload: txhashset archive for
00000000000000000000000000000000 at 666. size=5273845535
20190404 09:53:06.483 DEBUG grin_p2p::protocol - CUCO: re-creating /tmp/txhashset.zip
20190404 09:53:15.641 DEBUG grin_p2p::protocol - handle_payload: txhashset archive save to file
"/tmp/txhashset.zip" success
20190404 09:53:15.641 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetDownload {
start_time: 2019-04-04T12:53:06.483951296Z, prev_update_time: 2019-04-04T12:53:15.610356249Z,
update_time: 2019-04-04T12:53:15.619496668Z, prev_downloaded_size: 73200000, downloaded_size:
73248000, total_size: 5273845535 } -> TxHashsetSetup
20190404 09:53:15.641 DEBUG grin_chain::chain - txhashset_write: body_head -
32c1193af3731db9daafe0ec2c53ce2a, 0, header_head - cf57210e01ab0ba31aca2f8b827f13d6, 2583, sync_head -
cf57210e01ab0ba31aca2f8b827f13d6, 2583
20190404 09:53:16.585 DEBUG grin_chain::chain - txhashset_write: need a state sync for txhashset.
oldest block which is not on local chain: d456d242ca3c88e1c9d325cb6f9a8407 at 1
20190404 09:53:16.586 DEBUG grin_util::zip - new zip
20190404 09:53:16.680 ERROR grin_servers::common::adapters - Failed to save txhashset archive: Other
Error: Invalid Zip archive: Could not find central directory end
 Cause: Unknown
 Backtrace:
20190404 09:53:16.680 DEBUG grin_p2p::protocol - handle_payload: txhashset archive for
53c52127a7f430da079a9b5ca59f61d8 at 2562, DONE. Data Ok: true
20190404 09:53:16.689 ERROR grin_servers::grin::sync::state_sync - state_sync: error = Chain(Error {
inner:
Other Error: Invalid Zip archive: Could not find central directory end }). restart fast sync
```

3. And again . . .

```
20190404 09:53:16.692 DEBUG grin_servers::grin::sync::state_sync - state_sync: before txhashset
request, header head: 2583 / cf57210e01ab0ba31aca2f8b827f13d6, txhashset_head: 2562 /
53c52127a7f430da079a9b5ca59f61d8
20190404 09:53:16.693 DEBUG grin_p2p::peer - Asking 127.0.0.1:3414 for txhashset archive at 2562
53c52127a7f430da079a9b5ca59f61d8.
20190404 09:53:16.693 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetSetup ->
TxHashsetDownload { start_time: 2019-04-04T12:53:16.693269724Z, prev_update_time:
2019-04-04T12:53:16.693271572Z, update_time: 2019-04-04T12:53:16.693271997Z, prev_downloaded_size: 0,
downloaded_size: 0, total_size: 0 }
```

```
20190404 09:53:17.300 DEBUG grin_p2p::protocol - handle_payload: txhashset archive for
53c52127a7f430da079a9b5ca59f61d8 at 2562. size=3469942
20190404 09:53:17.300 DEBUG grin_p2p::protocol - CUCO: re-creating /tmp/txhashset.zip

20190404 09:54:04.075 ERROR grin_p2p::protocol - handle_payload: txhashset archive save to file fail.
err=Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190404 09:54:04.083 DEBUG grin_p2p::peer - Client 127.0.0.1:5414 connection lost: Connection(Custom
{ kind: ConnectionAborted, error: StringError("read_exact") })
```

4. When the archive from the well-behaved node at `127.0.0.1:3414` finishes downloading, the target node tries to process it in a corrupted context and decides to ban the node:

```
20190404 09:54:30.984 DEBUG grin_p2p::protocol - handle_payload: txhashset archive save to file
"/tmp/txhashset.zip" success
20190404 09:54:30.984 DEBUG grin_servers::common::types - sync_state: sync_status: TxHashsetDownload {
start_time: 2019-04-04T12:53:17.300537132Z, prev_update_time: 2019-04-04T12:54:29.905171419Z,
update_time: 2019-04-04T12:54:30.907664119Z, prev_downloaded_size: 3456000, downloaded_size: 3469942,
total_size: 3469942 } -> TxHashsetSetup
20190404 09:54:30.984 DEBUG grin_chain::chain - txhashset_write: body_head -
32c1193af3731db9daafe0ec2c53ce2a, 0, header_head - cf57210e01ab0ba31aca2f8b827f13d6, 2583, sync_head -
cf57210e01ab0ba31aca2f8b827f13d6, 2583
20190404 09:54:31.605 ERROR grin_store::types - Corrupted storage, could not read an entry from data
file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190404 09:54:31.605 ERROR grin_chain::chain - txhashset_write: something is wrong! oldest_height is
0
20190404 09:54:31.605 WARN grin_chain::chain - txhashset_write: txhashset received but it's not
needed! ignored.
20190404 09:54:31.606 ERROR grin_servers::common::adapters - Failed to save txhashset archive: Invalid
TxHashSet: not needed
 Cause: Unknown
 Backtrace:
20190404 09:54:31.606 DEBUG grin_p2p::peers - Received a bad txhashset data from 127.0.0.1:3414, the
peer will be banned
```

Recommendations

Coinspect recommends properly randomizing temporary file names in order to isolate storage used by threads processing remote peers.

GRIN-010 Node crashes when ulimit is reached with many incoming peer connections

Total Risk	Impact	Location
High	High	p2p/src/serv.rs:83
Fixed	**Likelihood**	
Yes	High	

Description

The Grin node panics with a “too many open files” exception when the process reaches the configured maximum file descriptors limit. An attacker can force this scenario by quickly establishing many simultaneous connections to the target node from different IP addresses.

This is a sample crash log:

```
20190501 22:25:05.611 DEBUG grin_p2p::peers - Saving newly connected peer 192.168.1.190:4414.
20190501 22:25:05.611 DEBUG grin_p2p::store - save_peer: PeerAddr(V4(192.168.1.190:4414)) marked
Healthy
20190501 22:25:05.650 DEBUG grin_p2p::peer - accept: handshaking from Ok(V4(192.168.1.190:48416))
20190501 22:25:05.651 DEBUG grin_p2p::peers - Saving newly connected peer 192.168.1.190:4414.
20190501 22:25:05.651 DEBUG grin_p2p::store - save_peer: PeerAddr(V4(192.168.1.190:4414)) marked
Healthy
20190501 22:25:05.702 DEBUG grin_p2p::peer - accept: handshaking from Ok(V4(192.168.1.190:48418))
20190501 22:25:05.708 DEBUG grin_p2p::peers - Saving newly connected peer 192.168.1.190:4414.
20190501 22:25:05.709 DEBUG grin_p2p::store - save_peer: PeerAddr(V4(192.168.1.190:4414)) marked
Healthy
20190501 22:25:05.746 DEBUG grin_p2p::peer - accept: handshaking from Ok(V4(192.168.1.190:48420))
20190501 22:25:05.753 DEBUG grin_p2p::peers - Saving newly connected peer 192.168.1.190:4414.
20190501 22:25:05.756 DEBUG grin_p2p::store - save_peer: PeerAddr(V4(192.168.1.190:4414)) marked
Healthy
20190501 22:25:05.792 DEBUG grin_p2p::peer - accept: handshaking from Ok(V4(192.168.1.190:48422))
20190501 22:25:05.795 DEBUG grin_p2p::peers - Saving newly connected peer 192.168.1.190:4414.
20190501 22:25:05.795 DEBUG grin_p2p::store - save_peer: PeerAddr(V4(192.168.1.190:4414)) marked
Healthy
20190501 22:25:05.825 DEBUG grin_p2p::peer - accept: handshaking from Ok(V4(192.168.1.190:48424))
...
20190501 22:25:05.850 ERROR grin_util::logger -
thread 'p2p-server' panicked at 'clone conn for reader failed: Os { code: 24, kind: Other, message:
"Too many open files" }': src/libcore/result.rs:1009stack backtrace:
   0: <no info> (0x7fc86c2f99b7)
   1: <no info> (0x7fc86c7eff29)
   2: <no info> (0x7fc86c7ef9d1)
   3: <no info> (0x7fc86c7ef8b5)
   4: <no info> (0x7fc86c80c23c)
   5: <no info> (0x7fc86c3b6e9f)
   6: <no info> (0x7fc86be84ed2)
   7: <no info> (0x7fc86be4c121)
   8: <no info> (0x7fc86be4bc59)
   9: <no info> (0x7fc86be0898c)
  10: <no info> (0x7fc86be1510b)
  11: <no info> (0x7fc86be12a38)
  12: <no info> (0x7fc86b2f92c0)
  13: <no info> (0x7fc86b39dba9)
  14: <no info> (0x7fc86b37a19b)
  15: <no info> (0x7fc86b38e03b)
  16: <no info> (0x7fc86b39fecc)
  17: <no info> (0x7fc86c8054f9)
  18: <no info> (0x7fc86b39f133)
  19: <no info> (0x7fc86b38e26b)
  20: <no info> (0x7fc86b379bae)
  21: <no info> (0x7fc86b37a88a)
  22: <no info> (0x7fc86c7f90bd)
  23: <no info> (0x7fc869841183)
  24: <no info> (0x7fc86935803c)
  25: <no info> (0x0)


20190501 22:25:19.014 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.034 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.036 DEBUG grin_p2p::handshake - Connected! Cumulative 234 offered from
PeerAddr(V4(192.168.1.190:4414)) "MW/Grin 1.0.2" HEADER_HIST | TXHASHSET_HIST | PEER_LIST |
TX_KERNEL_HASH | FULL_NODE
20190501 22:25:19.038 ERROR grin_util::logger -
thread 'peer_connect' panicked at 'clone conn for reader failed: Os { code: 24, kind: Other, message:
"Too many open files" }': src/libcore/result.rs:1009stack backtrace:
   0: <no info> (0x7fc86c2f99b7)
   1: <no info> (0x7fc86c7eff29)
   2: <no info> (0x7fc86c7ef9d1)
   3: <no info> (0x7fc86c7ef8b5)
   4: <no info> (0x7fc86c80c23c)
   5: <no info> (0x7fc86c3b6e9f)
   6: <no info> (0x7fc86be84ed2)
   7: <no info> (0x7fc86be4c121)
   8: <no info> (0x7fc86be4bc59)
   9: <no info> (0x7fc86be0898c)
  10: <no info> (0x7fc86be14995)
  11: <no info> (0x7fc86b383343)
  12: <no info> (0x7fc86b39db81)
  13: <no info> (0x7fc86b37a320)
  14: <no info> (0x7fc86b38e010)
  15: <no info> (0x7fc86b39fd76)
  16: <no info> (0x7fc86c8054f9)
  17: <no info> (0x7fc86b39f53f)
  18: <no info> (0x7fc86b38e342)
  19: <no info> (0x7fc86b379f3d)
  20: <no info> (0x7fc86b37ab45)
  21: <no info> (0x7fc86c7f90bd)
  22: <no info> (0x7fc869841183)
  23: <no info> (0x7fc86935803c)
  24: <no info> (0x0)


20190501 22:25:19.042 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.046 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.047 DEBUG grin_p2p::peer - Error shutting down conn: Os { code: 107, kind:
NotConnected, message: "Transport endpoint is not connected" }
20190501 22:25:19.047 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.063 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.067 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.085 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
```

```
20190501 22:25:19.088 DEBUG grin_p2p::peer - Error shutting down conn: Os { code: 107, kind:
NotConnected, message: "Transport endpoint is not connected" }
20190501 22:25:19.104 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.113 DEBUG grin_p2p::peer - connect: handshaking with Err(Os { code: 107, kind:
NotConnected, message: "Transport endpoint is not connected" }) failed with error: Connection(Custom {
kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.117 DEBUG grin_p2p::peer - Error shutting down conn: Os { code: 107, kind:
NotConnected, message: "Transport endpoint is not connected" }
20190501 22:25:19.126 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.136 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.144 DEBUG grin_p2p::peer - Error shutting down conn: Os { code: 107, kind:
NotConnected, message: "Transport endpoint is not connected" }
20190501 22:25:19.146 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.147 DEBUG grin_p2p::peer - Error shutting down conn: Os { code: 107, kind:
NotConnected, message: "Transport endpoint is not connected" }
20190501 22:25:19.152 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.188 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.243 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.268 DEBUG grin_p2p::peer - Error shutting down conn: Os { code: 107, kind:
NotConnected, message: "Transport endpoint is not connected" }
20190501 22:25:19.284 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.300 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.301 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.351 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.355 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.363 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.367 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.369 DEBUG grin_p2p::peer - Error shutting down conn: Os { code: 107, kind:
NotConnected, message: "Transport endpoint is not connected" }
20190501 22:25:19.370 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.409 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
20190501 22:25:19.418 DEBUG grin_p2p::peer - connect: handshaking with Err(Os { code: 107, kind:
NotConnected, message: "Transport endpoint is not connected" }) failed with error: Connection(Custom {
kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.429 DEBUG grin_p2p::peer - Error shutting down conn: Os { code: 107, kind:
NotConnected, message: "Transport endpoint is not connected" }
20190501 22:25:19.432 DEBUG grin_servers::grin::seed - monitor_peers: on 0.0.0.0:3414, 1 connected (1
most_work). all 1 = 0 healthy + 0 banned + 1 defunct
20190501 22:25:19.433 DEBUG grin_p2p::peer - connect: handshaking with Ok(V4(192.168.1.190:4414))
failed with error: Connection(Custom { kind: ConnectionAborted, error: StringError("read_exact") })
20190501 22:25:19.438 DEBUG grin_p2p::store - save_peer: PeerAddr(V4(192.168.1.117:3414)) marked
Healthy
20190501 22:25:24.865 DEBUG grin_servers::grin::seed - monitor_peers: no dandelion relay updating
20190501 22:25:24.866 DEBUG grin_p2p::peers - Could not update dandelion relay
20190501 22:25:44.873 DEBUG grin_servers::grin::seed - monitor_peers: on 0.0.0.0:3414, 1 connected (1
most_work). all 2 = 1 healthy + 0 banned + 1 defunct
20190501 22:25:44.897 DEBUG grin_servers::grin::seed - monitor_peers: no dandelion relay updating
20190501 22:25:44.898 DEBUG grin_p2p::peers - Could not update dandelion relay
20190501 22:25:54.022 DEBUG grin_servers::mining::test_miner - (Server ID: Port 3414) No solution
found after 53 iterations, continuing...
20190501 22:25:54.023 DEBUG grin_servers::mining::test_miner - setting pubkey in miner to pubkey from
block_fees - BlockFees { fees: 0, height: 85, key_id:
Some(Identifier(0300000000000000000000260500000000)) }
20190501 22:25:54.085 ERROR grin_util::logger -
```

```
thread 'test_miner' panicked at 'called `Result::unwrap()` on an `Err` value: Os { code: 24, kind:
Other, message: "Too many open files" }': src/libcore/result.rs:1009stack backtrace:
   0: <no info> (0x7fc86c2f99b7)
   1: <no info> (0x7fc86c7eff29)
   2: <no info> (0x7fc86c7ef9d1)
   3: <no info> (0x7fc86c7ef8b5)
   4: <no info> (0x7fc86c80c23c)
   5: <no info> (0x7fc86bc08950)
   6: <no info> (0x7fc86b6492af)
   7: <no info> (0x7fc86b6f66dc)
   8: <no info> (0x7fc86b42b693)
   9: <no info> (0x7fc86b42e177)
  10: <no info> (0x7fc86b5452ba)
  11: <no info> (0x7fc86b54471e)
  12: <no info> (0x7fc86b3725db)
  13: <no info> (0x7fc86b3704cf)
  14: <no info> (0x7fc86b36ebdb)
  15: <no info> (0x7fc86b36b0e2)
  16: <no info> (0x7fc86b2f93f6)
  17: <no info> (0x7fc86b39db42)
  18: <no info> (0x7fc86b37a1e5)
  19: <no info> (0x7fc86b38e0a5)
  20: <no info> (0x7fc86b39fc29)
  21: <no info> (0x7fc86c8054f9)
  22: <no info> (0x7fc86b39f9df)
  23: <no info> (0x7fc86b38e225)
  24: <no info> (0x7fc86b37951f)
  25: <no info> (0x7fc86b37a797)
  26: <no info> (0x7fc86c7f90bd)
  27: <no info> (0x7fc869841183)
  28: <no info> (0x7fc86935803c)
  29: <no info> (0x0)
```

As a consequence of the file descriptor exhaustion, any thread attempting to open a new file descriptor panics; in the logs above, **the miner, peer, and p2p server threads can be observed crashing**. The node must be restarted manually to resume normal operation.

Recommendations

1. Enforce incoming connection rate limits in the network layer (before the peer handshake takes place).
2. Recommend proper *ulimit* values in the node installations guide, Wiki, etc.

GRIN-011 High CPU usage when handling many incoming peer connections results in stuck miner and unresponsive node

Total Risk	Impact	Location
Medium	High	servers/src/grin/seed.rs:141
Fixed	**Likelihood**	
Yes	Low	

Description

The Grin node process can be forced to consume 100% of the host's available CPU power.

An attacker can force this scenario by establishing many simultaneous connections to the target node from different IP addresses. Additionally, the attacker can stall each connection handshake as much as allowed by the reader timeout in order to be able to pool more connections before the peers are cleaned.

This is a sample log obtained during an attack simulation in which the attacker connects 1000 times to the target node:

```
20190502 15:48:20.124 WARN grin_servers::grin::server - Grin server started.
20190502 15:48:20.124 INFO grin_servers::grin::server - start_test_miner - start

20190502 15:48:51.138 WARN grin_servers::grin::sync::syncer - sync: no peers available, disabling sync
20190502 15:48:55.126 INFO grin_servers::mining::test_miner - (Server ID: Port 3414) Starting test
miner loop.
20190502 15:49:01.149 WARN grin_servers::grin::sync::syncer - sync: no peers available, disabling sync
20190502 15:49:03.103 INFO grin_servers::mining::test_miner - (Server ID: Port 3414) Found valid proof
of work, adding block 3c54ea777821480354f6e7ae46105dcd (prev_root 23d6a7462901ab722b1b1b1bf81ae23d).
20190502 15:49:10.072 INFO grin_servers::mining::test_miner - (Server ID: Port 3414) Found valid proof
of work, adding block 6d6aaf68bcdf7a65117314e5a33bfd62 (prev_root 9675ac5c126a1fa524121c18f82ca448).
20190502 15:49:11.160 WARN grin_servers::grin::sync::syncer - sync: no peers available, disabling sync
20190502 15:49:21.171 WARN grin_servers::grin::sync::syncer - sync: no peers available, disabling sync
20190502 15:49:23.290 INFO grin_servers::mining::test_miner - (Server ID: Port 3414) Found valid proof
of work, adding block 9150edecd9064ff9eb073abef8fe6b3f (prev_root 33325160c737978551dabeea1bac212d).
20190502 15:49:31.181 WARN grin_servers::grin::sync::syncer - sync: no peers available, disabling sync
20190502 15:49:32.644 INFO grin_servers::mining::test_miner - (Server ID: Port 3414) Found valid proof
of work, adding block 6e895d14672c36e9589bbec1abe4f59e (prev_root 859c8755a7fb0ff8a36a45d185098ae3).
20190502 15:49:41.192 WARN grin_servers::grin::sync::syncer - sync: no peers available, disabling sync
20190502 15:49:51.203 WARN grin_servers::grin::sync::syncer - sync: no peers available, disabling sync
20190502 15:49:56.406 INFO grin_servers::mining::test_miner - (Server ID: Port 3414) Found valid proof
of work, adding block aaabceb2d3ca7a7bb73c4767cc89c0cf (prev_root 9b609d4bff449865aeddd9b58cf76d38).
20190502 15:50:01.214 WARN grin_servers::grin::sync::syncer - sync: no peers available, disabling sync
20190502 15:50:11.225 WARN grin_servers::grin::sync::syncer - sync: no peers available, disabling sync
20190502 15:50:21.235 WARN grin_servers::grin::sync::syncer - sync: no peers available, disabling sync
20190502 15:50:29.650 INFO grin_servers::mining::test_miner - (Server ID: Port 3414) Found valid proof
of work, adding block 0ec6500c17d1f2c031ee77e3b602cca1 (prev_root 5f812f1b38df4c39438fe12b7b1b764d).

[attack started]

^C20190502 15:59:40.189 WARN grin::cmd::server - Received SIGINT (Ctrl+C) or SIGTERM (kill).
```

```
20190502 15:59:41.191 WARN grin::cmd::server - Shutting down...
20190502 15:59:42.192 WARN grin::cmd::server - Shutdown complete.
```

As seen above, **an attacker can prevent the miner from creating new blocks.**

Recommendations

Coinspect suggests enforcing incoming connection limits at the network layer *before* a new peer is accepted. Also, the networking code should be reviewed to clarify why the current code consumes so much CPU for handling idle peer connections.

GRIN-012 Miner thread panic after long chain reorganization

Total Risk	Impact	Location
Low	High	core/pmmr/pmmr.rs:144
Fixed	Likelihood	
Yes	Low	

Description

Miner (only tested with test_miner) panics after a chain reorganization.

The observed behavior takes place after a new node with a longer chain forces the target miner to synchronize (from height ~96 to ~2600). Chain data directories for both nodes is available for further investigation upon request.

In pmmr.rs root function:

```
/// Computes the root of the MMR. Find all the peaks in the current
/// tree and "bags" them to get a single peak.
pub fn root(&self) -> Hash {
  if self.is_empty() {
    return ZERO_HASH;
  }
  let mut res = None;
  for peak in self.peaks().iter().rev() {
    res = match res {
      None => Some(*peak),
      Some(rhash) => Some((*peak, rhash).hash_with_index(self.unpruned_size())),
    }
  }
  // miner panic
  res.expect("no root, invalid tree")
}
```

In mine_block.rs build_block function:

```
debug!(
  "Built new block with {} inputs and {} outputs, block difficulty: {}, cumulative difficulty {}",
  b.inputs().len(),
  b.outputs().len(),
  difficulty.difficulty,
  b.header.total_difficulty().to_num(),
);

// Now set txhashset roots and sizes on the header of the block being built.
match chain.set_txhashset_roots(&mut b) {
  Ok(_) => Ok((b, block_fees)),
```

This is the resulting test_miner thread panic:

```
20190320 21:07:09.260 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:11.240 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:13.424 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:15.325 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:17.118 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:19.014 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:20.899 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:22.560 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:24.345 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:26.030 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:28.002 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:29.895 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:31.767 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:33.784 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:35.709 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:37.007 INFO grin_servers::common::adapters - Received 32 block headers from 127.0.0.1:5414
20190320 21:07:38.595 INFO grin_servers::common::adapters - Received 2 block headers from 127.0.0.1:5414
20190320 21:07:42.883 INFO grin_servers::mining::test_miner - (Server ID: Port 3414) Found valid proof of work, adding block e7196e18e889 (prev_root 4389e6a6817e).
20190320 21:09:37.856 ERROR grin_store::types - Corrupted storage, could not read an entry from data file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190320 21:09:37.856 ERROR grin_store::types - Corrupted storage, could not read an entry from data file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190320 21:09:37.856 ERROR grin_store::types - Corrupted storage, could not read an entry from data file: IOErr("failed to fill whole buffer", UnexpectedEof)
20190320 21:09:45.587 ERROR grin_util::logger -
thread 'test_miner' panicked at 'no root, invalid tree': src/libcore/option.rs:1008stack backtrace:
   0: grin_util::logger::send_panic_to_log::{{closure}}::hcfe2f0cc5d0c7575 (0x7f43519e7177)
             at util/src/logger.rs:240
   1: std::panicking::rust_panic_with_hook::h8cbdfe43764887be (0x7f4351edd6e9)
             at src/libstd/panicking.rs:495
   2: std::panicking::continue_panic_fmt::h3d3c5a833c00a5e1 (0x7f4351edd191)
             at src/libstd/panicking.rs:398
   3: rust_begin_unwind (0x7f4351edd075)
             at src/libstd/panicking.rs:325
   4: core::panicking::panic_fmt::h4d67173bc68f6d5a (0x7f4351ef99fc)
             at src/libcore/panicking.rs:95
   5: core::option::expect_failed::h2f881c519f1d8001 (0x7f4351ef9a72)
             at src/libcore/option.rs:1008
   6: <core::option::Option<T>>::expect::hde28c366cb78588a (0x7f435168ef06)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libcore/option.rs:322
   7: <grin_core::core::pmmr::pmmr::PMMR<'a, T, B>>::root::h3fb3abea0594044d (0x7f435169ced2)
             at /home/u/GRIN/grin/core/src/core/pmmr/pmmr.rs:145
   8: grin_chain::txhashset::txhashset::Extension::header_root::h7c6eac04ee0763d5 (0x7f4351606df2)
             at chain/src/txhashset/txhashset.rs:1098
   9: grin_chain::chain::Chain::set_txhashset_roots::{{closure}}::h531328909fde73f2 (0x7f43516a25e0)
             at chain/src/chain.rs:618
  10: grin_chain::txhashset::txhashset::extending_readonly::h610bd43e04e95278 (0x7f43515e7876)
             at chain/src/txhashset/txhashset.rs:343
  11: grin_chain::chain::Chain::set_txhashset_roots::hea484c2661df7cd4 (0x7f43515d0271)
             at chain/src/chain.rs:616
  12: grin_servers::mining::mine_block::build_block::h14f13133a04194bf (0x7f4350a5e2e2)
             at servers/src/mining/mine_block.rs:154
  13: grin_servers::mining::mine_block::get_block::ha8710b6a5847cccd (0x7f4350a5be2b)
             at servers/src/mining/mine_block.rs:46
  14: grin_servers::mining::test_miner::Miner::run_loop::hd242da1c310bd97d (0x7f4350a582c2)
             at servers/src/mining/test_miner.rs:149
          15:     grin_servers::grin::server::Server::start_test_miner::{{closure}}::h291d7a8b77b65a0c
(0x7f43509e57b6)
             at servers/src/grin/server.rs:387
  16: std::sys_common::backtrace::__rust_begin_short_backtrace::hcea30122bbdc7c77 (0x7f4350a8ba22)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/sys_common/backtrace.rs:136
            17:      std::thread::Builder::spawn_unchecked::{{closure}}::{{closure}}::h3bbc155d44e34f47
(0x7f4350a67435)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:477
                             18:                <std::panic::AssertUnwindSafe<F>                as
core::ops::function::FnOnce<()>>::call_once::h73229e6018e88cfe (0x7f4350a7b675)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:319
  19: std::panicking::try::do_call::h046bad1ff27ead0a (0x7f4350a8dcc9)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:310
  20: __rust_maybe_catch_panic (0x7f4351ef2cb9)
             at src/libpanic_unwind/lib.rs:102
  21: std::panicking::try::hb7c4e13120d6e52c (0x7f4350a8da7f)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panicking.rs:289
  22: std::panic::catch_unwind::h6c816707ed1cb335 (0x7f4350a7b7f5)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/panic.rs:398
  23: std::thread::Builder::spawn_unchecked::{{closure}}::ha325740b92a20994 (0x7f4350a6676f)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/libstd/thread/mod.rs:476
  24: <F as alloc::boxed::FnBox<A>>::call_box::h042c42b77d46ff6a (0x7f4350a679e7)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:673
          25:     <alloc::boxed::Box<(dyn     alloc::boxed::FnBox<A,     Output=R>     +     'a)>     as
core::ops::function::FnOnce<A>>::call_once::hece536cf07b94f8d (0x7f4351ee687d)
             at /rustc/9fda7c2237db910e41d6a712e9a2139b352e558b/src/liballoc/boxed.rs:683
      std::sys_common::thread::start_thread::h9605a7df0f911844
             at src/libstd/sys_common/thread.rs:24
      std::sys::unix::thread::Thread::new::thread_start::hca8e72c41fa9d291
             at src/libstd/sys/unix/thread.rs:90
  26: start_thread (0x7f434ef2d183)
  27: clone (0x7f434ea4403c)
  28: <unknown> (0x0)


Thread 'test_miner' panicked with message:
"no root, invalid tree"
See /home/u/GRIN/grin/config.jp.UserTesting/n1/grin-server.log for further details.
20190320  21:09:45.639  INFO  grin_servers::common::adapters  -  Received  valid  txhashset  data  for
85f9a2cbe1b1.
20190320  21:09:59.710  INFO  grin_servers::grin::sync::syncer  -  synchronized  at  147594  @  2562
[53c52127a7f4]
```

This finding has not been fully researched because no directly exploitable scenario not requiring massive hashing power was found.

GRIN-013 Arbitrary orphan blocks can be used to flush out legitimate ones from the OrphanBlockPool

Total Risk	Impact	Location
Medium	Low	chain/src/chain.rs:96 chain/src/chain.rs:293 chain/src/pipe.rs:114
Fixed	Likelihood	
Yes	High	

Description

The *OrphanBlockPool* data structure holds orphan blocks received from the network. When the *MAX_ORPHAN_SIZE* (currently 200 blocks) is reached, blocks that are too old or too far ahead are evicted.

The code responsible for the chain block acceptance pipeline fails to validate the block when the block is an orphan, as shown below:

```
/// Runs the block processing pipeline, including validation and finding a
/// place for the new block in the chain.
/// Returns new head if chain head updated.
pub fn process_block(b: &Block, ctx: &mut BlockContext<'_>) -> Result<Option<Tip>, Error> {
  // TODO should just take a promise for a block with a full header so we don't
  // spend resources reading the full block when its header is invalid

  debug!(
    "pipe: process_block {} at {} [in/out/kern: {}/{}/{}]",
    b.hash(),
    b.header.height,
    b.inputs().len(),
    b.outputs().len(),
    b.kernels().len(),
  );

  // Check if we have already processed this block previously.
  check_known(b, ctx)?;

  // Delay hitting the db for current chain head until we know
  // this block is not already known.
  let head = ctx.batch.head()?;
  let is_next = b.header.prev_hash == head.last_block_h;

  let prev = prev_header_store(&b.header, &mut ctx.batch)?;

  // Block is an orphan if we do not know about the previous full block.
  // Skip this check if we have just processed the previous block
  // or the full txhashset state (fast sync) at the previous block height.
  if !is_next && !ctx.batch.block_exists(&prev.hash())? {
    return Err(ErrorKind::Orphan.into());
  }

  // This is a fork in the context of both header and block processing
  // if this block does not immediately follow the chain head.
```

```
let is_fork = !is_next;

// Check the header is valid before we proceed with the full block.
process_header_for_block(&b.header, is_fork, ctx)?;

// Validate the block itself, make sure it is internally consistent.
// Use the verifier_cache for verifying rangeproofs and kernel signatures.
validate_block(b, ctx)?;
```

As a consequence, an attacker:

1. Can abuse the eviction mechanism to flush legitimate blocks out of the pool, imposing higher bandwidth consumption (as previously received orphan blocks will need to be re-requested) and longer network latency for block propagation.
2. Can grow the orphan pool to use around 1.2 Gb, potentially crashing nodes running in hosts with low resources: 200 elements * 6 Mb; block max size is ~1.5 Mb, but a 4x growth factor is accepted as hardcoded in the *read_header* function implementation in *msg.rs.*

Recommendations

Coinspect recommends properly validating orphan blocks, including their proof of work, before accepting them into the orphans pool.

GRIN-014 Known headers replay can be abused to clog victim node CPU with PoW computations

Total Risk **Medium**	Impact Medium	Location
Fixed **Yes**	Likelihood High	

Description

Malicious peers can continuously replay old headers through the *Header* message, and those old headers will be validated, including their proof of work.

Peers replaying already known headers are not banned and can abuse this fact to **clog the victim's CPU with PoW computations indefinitely.**

The only limiting measure currently implemented is the message count rate limit, which bans a node as abusive if more than 500 messages per minute are received. Using many peers and staying below that threshold suffice to execute a successful attack.

Recommendations

Coinspect recommends checking that headers are present in the store before further validation takes place. Additionally, consider banning nodes replaying known headers.