Nlnet NGI Review Security Evaluation

Pendulum Project ntpd-rs

V 1.1
Amsterdam, April 14th, 2023
PUBLIC

Document Properties

Client	Pendulum Project ntpd-rs
Title	NLnet NGI Review Security Evaluation
Targets	Security evaluation of https://github.com/pendulum-project/ntpd-rs Code audit of ntpd-rs Manual testing of ntpd-rs Dynamic fuzz testing of ntpd-rs Retesting of issues fixed during the review period
Version	1.1
Pentester	Christian Reitter
Authors	Christian Reitter, Peter Mosmans
Reviewed by	Peter Mosmans
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	April 11th, 2023	Christian Reitter	Initial draft
1.0-draft	April 12th, 2023	Peter Mosmans	Initial review, minor edits
1.0	April 13th, 2023	Peter Mosmans	Release 1.0
1.1	April 14th, 2023	Peter Mosmans	Public release

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Science Park 608 1098 XH Amsterdam The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Between March 13, 2023 and April 13, 2023, Radically Open Security B.V. carried out a code audit for Pendulum Project ntpd-rs. This report contains our findings as well as detailed explanations of exactly how ROS performed the code audit.

1.2 Scope of work

The scope of the code review was limited to the following target(s):

- Security evaluation of https://github.com/pendulum-project/ntpd-rs
- Code audit of ntpd-rs
- Manual testing of ntpd-rs
- Dynamic fuzz testing of ntpd-rs
- Retesting of issues fixed during the review period

The ntpd-rs project was analyzed mainly in revision fc7f5a8c29e8fa8c6f76cb296d39be8ccc1dc0d2.
We performed the ntpd-rs retest checks on revision 4b84ca5b8af59d1b952e009ea6584bf91ea57588.

The scoped services are broken down as follows:

- Time-boxed code audit of the targets including reporting, communication and all other relevant steps: 10 days
- **Total effort: 10 days**

1.3 Project objectives

ROS will perform a code audit of application code of Pendulum Project in order to assess the overall security and use of cryptography. To do so ROS will get access to the relevant source code and guide Pendulum Project in attempting to find vulnerabilities. ROS will document any such found, assess relevant impact and give recommendations.

1.4 Timeline

The Security Audit took place between March 13, 2023 and April 13, 2023.

1.5 Results In A Nutshell

During this crystal-box code audit we found 1 Elevated, 4 Moderate, 12 Low and 3 N/A severity issues.

The issues related to known vulnerabilities in dependencies, deviation from cryptographic best practices, problematic local file handling and permissions, rate limiting system weaknesses, lack of support for additional certificate verification mechanisms, concerns related to trust in NTP peers, insufficient handling of misbehaving NTP peers, insecure log handling, denial of service risks and problematic error handling.

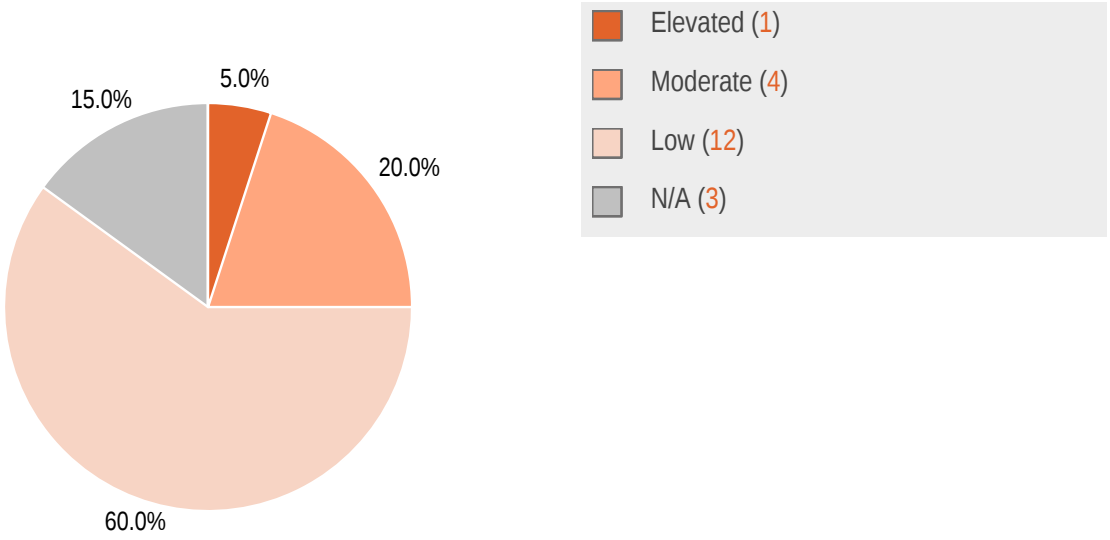
By exploiting these issues, an attacker might be able to crash the ntp-daemon or other ntpd-rs programs, remotely block successful time synchronization of ntp-daemon under some conditions, bypass rate limiting protections, manipulate the daemon configuration, learn of some ntp-daemon status information, pollute log files, or gain advantages during attacks on TLS certificates under some conditions.

1.6 Summary of Findings

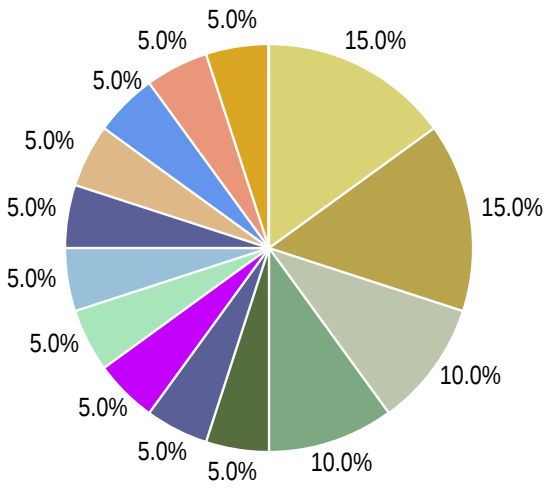
ID	Type	Description	Threat level
NTP-054	CWE-248: Uncaught Exception	The NTS client logic in ntpd-rs aborts on problematic network conditions that can be triggered by a remote attacker.	Elevated
NTP-051	CWE-693: Protection Mechanism Failure	ntpd-rs does not correctly handle peers that are unreachable or exhibit other behavior issues, increasing the risk of time synchronization failures or opportunities for manipulation.	Moderate
NTP-040	CWE-799: Improper Control of Interaction Frequency	In IPv6 environments, performing rate limiting for NTP based on individual IP addresses is difficult since attackers can send and receive packets from a large number of IPv6 addresses to trigger rate limiting exceptions.	Moderate
NTP-039	CWE-799: Improper Control of Interaction Frequency	The rate limiting implementation of ntpd-rs uses the UDP source port number and IP address combination to identify a peer, making it unable to correlate multiple requests from the same IP address that use a different source port.	Moderate
NTP-038	CWE-799: Improper Control of Interaction Frequency	ntpd-rs uses a hash table mechanism that insufficiently randomizes placement in the hash table and overwrites content on collisions. This allows attackers to predict table index locations and provoke collisions to bypass the rate limiting.	Moderate
NTP-062	CWE-361: Time and State	In the default configuration, protection mechanisms for initial system time changes are effectively disabled for forward clock adjustments, which can be leveraged for attacks.	Low
NTP-061	CWE-248: Uncaught Exception	read_json() does not correctly handle parsing failures, causing panics if malformed data is processed.	Low
NTP-060	CWE-779: Logging of Excessive Data	The ntpd-rs logs warnings per received packet for some unusual conditions, which represents an opportunity for attackers to trigger excessive logging.	Low
NTP-059	Certificate Handling	ntpd-rs does not allow pinning of specific TLS keys. If implemented, this feature could be used in some well-controlled client and server instances to provide additional protection.	Low
NTP-056	CWE-703: Improper Check or Handling of Exceptional Conditions	ntpd-rs starts up in a degraded state when encountering bootstrapping problems that obstruct the correct operation of subcomponents. This behavior may be unexpected and unwanted to operators.	Low
NTP-053	CWE-532: Insertion of Sensitive Information into Log File	Log operations which print the PeerNTSData reveal secrets in the form of NTS cookies.	Low

NTP-050	Peer handling and trust	ntpd-rs currently has no mechanism to rotate or re-evaluate established NTP peers which are sourced from an NTP pool as long as they're seen as active and well-behaved by the daemon.	Low
NTP-043	CWE-299: Improper Check for Certificate Revocation	ntpd-rs currently does not support Online Certificate Status Protocol (OCSP) stapling or verification due to missing features of the Rust TLS cryptography libraries.	Low
NTP-041	CWE-276: Incorrect Default Permissions	The ntpd-rs socket default file permissions are too lax.	Low
NTP-034	CWE-15: External Control of System or Configuration Setting	The ntp-daemon includes the ntp.toml configuration file preferably from the local working directory, if available. Under some edge conditions, this directory or file may not be sufficiently protected, giving unauthorized local users control over the configuration.	Low
NTP-029	CWE-1104: Use of Unmaintained Third Party Components	ntpd-rs directly depends on the webpki crate, which appears to be unmaintained and has at least one known issue.	Low
NTP-024	CWE-693: Protection Mechanism Failure	ntpd-rs does not explicitly overwrite cryptographic key material in memory after use. This increases the risk of compromise in the event that some other unspecified issue discloses program memory to attackers.	Low
NTP-058	CWE-703: Improper Check or Handling of Exceptional Conditions	An attacker with write access to the ntpd-rs configuration may leverage the daemon communication sockets to trash arbitrary files on the system under some conditions.	N/A
NTP-032	CWE-248: Uncaught Exception	Remote attackers can crash the demobilize-server with crafted network communications. This server is only for testing purposes and not part of the main ntpd-rs.	N/A
NTP-021	CWE-347 Improper Verification of Cryptographic Signature	The cargo tool in Rust versions before 1.66.1 does not verify SSH host keys, which allows machine-in-the-middle (MitM) attacks on repository cloning. A vulnerable version is used in the Continuous Integration (CI) infrastructure, but the affected functionality is not triggered.	N/A

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



- Cwe-248: uncaught exception (3)
- Cwe-799: improper control of interaction frequency (3)
- Cwe-703: improper check or handling of exceptional conditions (2)
- Cwe-693: protection mechanism failure (2)
- Cwe-361: time and state (1)
- Cwe-779: logging of excessive data (1)
- Certificate handling (1)
- Cwe-532: insertion of sensitive information into log file (1)
- Peer handling and trust (1)
- Cwe-299: improper check for certificate revocation (1)
- Cwe-276: incorrect default permissions (1)
- Cwe-15: external control of system or configuration setting (1)
- Cwe-1104: use of unmaintained third party components (1)
- Cwe-347 improper verification of cryptographic signature (1)

1.7 Summary of Recommendations

ID	Type	Recommendation
NTP-062	CWE-361: Time and State	• Consider implementing stricter limits for forward time changes in the default setting. • Make the current forward-unbounded limit a well-documented alternative configuration. • Evaluate potential additional opt-in safety mechanisms to logically restrict time adjustments and mitigate attacks. • For example, the timestamp of the ntp-daemon program compilation could be useful as a lower bound for time adjustments when running with production settings.
NTP-061	CWE-248: Uncaught Exception	• Make the JSON parsing handling more robust.
NTP-060	CWE-779: Logging of Excessive Data	• Lower the log verbosity level of edge cases that are in reach of a remote attacker.
NTP-059	Certificate Handling	• Consider adding TLS pinning functionality. • The rustls-pin crate may offer the required functionality.
NTP-058	CWE-703: Improper Check or Handling of Exceptional Conditions	• Change the socket creation logic to error out if the target file exists and is a regular file. • Evaluate if restricting the socket creation to `/run/` or other designated directories is acceptable for operators.
NTP-056	CWE-703: Improper Check or Handling of Exceptional Conditions	• Cleanly exit with error code exit status after encountering fatal problems during startup.
NTP-054	CWE-248: Uncaught Exception	• Ensure the NTS logic is robust against temporary unavailable peers or network failure. • Establish regular testing under non-optimal or adverse network conditions. • Consider adding automated tests running on simulated lossy or unreliable networks.
NTP-053	CWE-532: Insertion of Sensitive Information into Log File	• Reconfigure the log operation to not print `CookieStash` information by default. • If necessary, include an opt-in configuration setting (ideally at compile time) to print sensitive information for debugging.
NTP-051	CWE-693: Protection Mechanism Failure	• Correct or improve the peer handling logic with regards to observable peer issues. • Improve logging to warn operators of serious error conditions such as loss of time sync. • Improve logging to assist operators with identifying problematic peer servers. • This is partially related to NTP-050 (page 32).
NTP-050	Peer handling and trust	• We recommend further investigation into mechanisms and relevant trade-offs to re-acquire NTP peers from a pool. • Solution ideas include per-pool configuration of time-to-live settings for peers, combined with staggered peer refresh and some randomization.

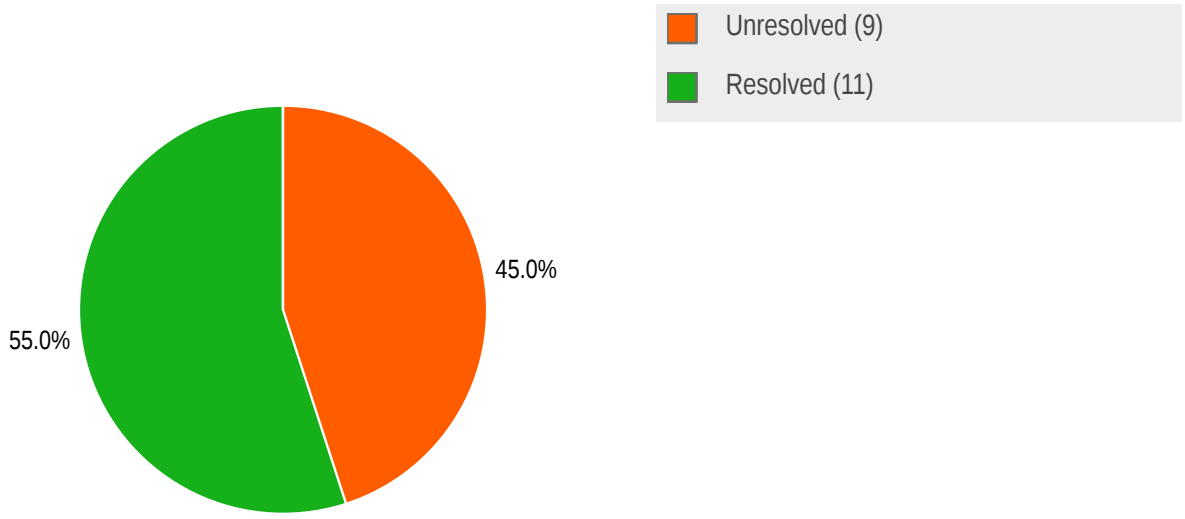
		• Since peer changes may temporarily influence the time sync precision, this behavior should be configurable.
NTP-043	CWE-299: Improper Check for Certificate Revocation	• Investigate documenting potential workarounds on the server side using TLS proxy systems that support OCSP stapling.
NTP-041	CWE-276: Incorrect Default Permissions	• Adopt a less permissive default configuration. • Investigate potential benefits of configurable socket file ownership to reduce the need of permissive read+write access for other local users.
NTP-040	CWE-799: Improper Control of Interaction Frequency	• Within the current system and protocol constraints, we have not been able to determine and recommend a clearly superior rate limiting design. • Increasing the rate limit bucket size to interactions per `/64` IPv6 subnet (or larger) can correlate individual addresses in an attack under some conditions, but risks overblocking legitimate peer interaction. • Improve the general visibility of attacks and attack characteristics to operators via logging and other methods. • Improve the documentation and handling of allow-list and deny-list functionality or recommend firewall-based alternatives.
NTP-039	CWE-799: Improper Control of Interaction Frequency	• Switch to rate limiting based only on `IP` address information without port numbers. • See also related challenges for rate limiting on IPv6-enabled servers in NTP-040 (page 36).
NTP-038	CWE-799: Improper Control of Interaction Frequency	• Ensure sufficient randomization of the hash table behavior to prevent targeted hash table collisions. • Randomize the hash table placement on each server startup with sufficient entropy. • Mitigations related to hash table size are subject to other technical trade-offs, see related findings. • Add optional rate limiting behavior to drop requests instead of replying with a `RATE` response to reduce outgoing traffic during untargeted attacks.
NTP-034	CWE-15: External Control of System or Configuration Setting	• Remove the local default inclusion of `./ntp.toml`. • Users can still use the existing configuration flag to emulate this behavior via `--config ./ntp.toml` if needed.
NTP-032	CWE-248: Uncaught Exception	• Improve the exception error handling. • Perform additional testing of the `demobilize-server` behavior.
NTP-029	CWE-1104: Use of Unmaintained Third Party Components	• Remove the explicit `webpki` dependency from `Cargo.toml` list of dependencies. • Switch to the `rustls-webpki` fork once possible. • Follow up on indirect dependency requirements for `webpki`.
NTP-024	CWE-693: Protection Mechanism Failure	• Consider using the `zeroize` Rust crate to implement memory sanitization actions for important program secrets.
NTP-021	CWE-347 Improper Verification of	• If possible, use security backports for Cargo or upgrade to patched Rust versions. • Document the issue to avoid accidental use of affected functionality.

	Cryptographic Signature	

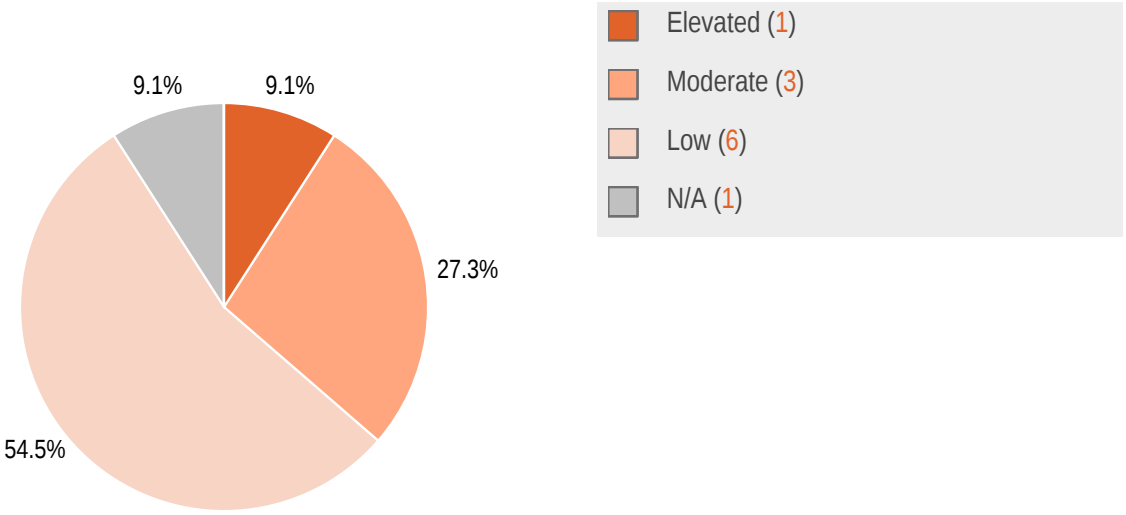
1.8 Summary of Retest

Notably, a significant number of findings were resolved during the initial engagement period.

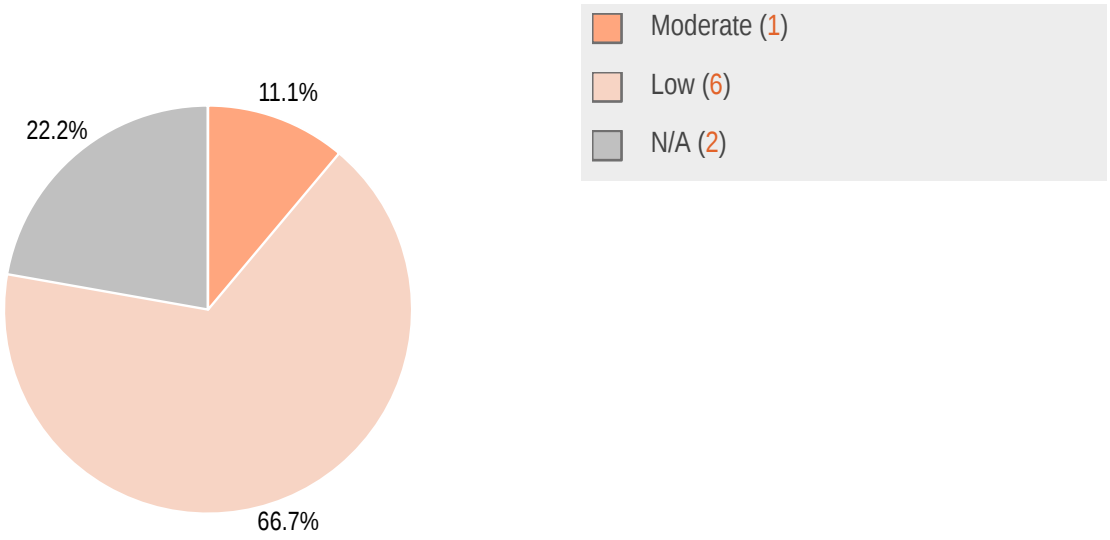
1.8.1 Overall Retest Status



1.8.2 Overview of Resolved Findings by Severity



1.8.3 Overview of Unresolved Findings by Severity



2 Methodology

2.1 Code Audit

ROS will perform a code audit. During this process we will verify if the proper security controls are present, work as intended and are implemented correctly. If vulnerabilities are found, we determine the threat level by assessing the likelihood of exploitation of this vulnerability and the impact on the Confidentiality, Integrity and Availability (CIA) of the system. We will describe how an attacker would exploit the vulnerability and suggest ways of fixing it.

This requires an extensive knowledge of the platform the application is running on, as well as the extensive knowledge of the language the application in written in and patterns that have been used. Therefore a code audit done by highly-trained specialists with a strong background in programming.

During the code audit, we take the following approach:

1. **Thorough comprehension of functionality**
   We try to get a thorough comprehension of how the application works and how it interacts with the user and other systems. Having detailed documentation (manuals, flow charts, system sequence diagrams, design documentation) at this stage is very helpful, as they aid the understanding of the application.
2. **Comprehensive code reading**
   Goals of the comprehensive code reading are:
   - To get an understanding of the whole code.
   - Identify adversary controlled inputs and trace their paths.
   - Identify issues.

3. **Static analysis**
   Using the understanding we gained in the previous step, we will use static code analysis to uncover any vulnerabilities. Static analysis means the specialist will analyze the code and implementation of security controls to get an understanding of the security of the application, rather than running the application to reach the same goal. This is primarily a manual process, where the specialist relies on his knowledge and expertise to find the flaws in the application. The specialist may be aided in this process by automatic analysis tools, but his or her skills are the driving force.
   Depending on the type of application, we will identify the endpoints. In this case, it means where data enters and leaves the application. The data is then followed through the application and is leading in determining if assessing the quality of the security measures.
4. **Dynamic analysis**
   Dynamic analysis can also be performed. In this case, the program is run and actively exploited by the specialist. This is usually done to confirm a vulnerability and as such follows the result of the static analysis.

5. **Fuzzing**

   Fuzz testing or Fuzzing is a software testing technique which in essence consists of finding implementation bugs using malformed/semi-malformed data injection in an automated fashion.

2.2 Risk Classification

Throughout the report, vulnerabilities or risks are labeled and categorized according to the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting

These categories are:

- **Extreme**

  Extreme risk of security controls being compromised with the possibility of catastrophic financial/reputational losses occurring as a result.
- **High**

  High risk of security controls being compromised with the potential for significant financial/reputational losses occurring as a result.
- **Elevated**

  Elevated risk of security controls being compromised with the potential for material financial/reputational losses occurring as a result.
- **Moderate**

  Moderate risk of security controls being compromised with the potential for limited financial/reputational losses occurring as a result.
- **Low**

  Low risk of security controls being compromised with measurable negative impacts as a result.

Additionally, the report contains finding with **N/A** severity level that represent informational findings below the level of **Low** to better distinguish between risk levels of minor security concerns.

3 Findings

We have identified the following issues:

3.1 NTP-062 — ntp-daemon Allows Unbounded Forward Clock Changes at Startup

Vulnerability ID: NTP-062 **Status:** Unresolved

Vulnerability type: CWE-361: Time and State

Threat level: Low

Description:

In the default configuration, protection mechanisms for initial system time changes are effectively disabled for forward clock adjustments, which can be leveraged for attacks.

Technical description:

As outlined in the paper `Attacking the Network Time Protocol` by Malhotra, Aanchal, et al. (Cryptology ePrint Archive, 2015), it is publicly known that NTP client implementations which allow larger time adjustments either at general runtime or directly after daemon startup inadvertently provide significant manipulation opportunities to attackers. By manipulating the time of a victim system in the order of months or years away from the actual time, other security mechanisms such as TLS certificate validations and other cryptography mechanisms can be broken or subverted.

The current `ntpd-rs` configuration distinguishes between the maximum allowed time changes at startup (`startup-panic-threshold`) and runtime (`panic-threshold`), with individual control over time changes forward and backward in time for each setting.

Although the default `ntpd-rs` NTP configuration correlates time server responses from multiple peers, a well-placed Machine-in-the-Middle (MitM) attacker on the network path towards the peers or with equivalent network access (such as control over DNS server replies) can plausibly manipulate all peer responses and trigger arbitrary clock changes in the client (within the configured thresholds). Similarly, NTS clients relying on malicious NTS servers could be tricked into adjusting their system clock to incorrect time values.

We expect that a time manipulation forward in time may not be as severe as one backward in time since it does not allow bypassing essential TLS protections via the re-use of previously compromised TLS certificates by the MitM attacker. However, we still see a significant risk of Denial-of-Service (DoS) attacks that go beyond the scope of the NTP service itself, for example by moving the victim system clock forward beyond the point of expiry of common TLS certificates and therefore disrupting other TLS-protected communications of the host. To give another example, under some conditions,

attacked systems may exhibit other problematic behavior when manipulated in a way that triggers the Year 2038 problem.

We acknowledge the intention of the `ntpd-rs` developers in CONFIGURATION.md for the current default value to be compatible with systems that start up with system clocks in the past:

> By default, this is unrestricted as we may be the initial source of time for systems without a hardware backed clock.

However, it may be possible to disallow unbounded changes by default and turn the current permissive settings into an alternative configuration that is only recommended for systems which require it due to their hardware configuration.

Impact:

- The current default setting results in weakened system clock integrity protection for time changes at startup.
- A Machine-in-the-Middle (MitM) attacker can trigger denial of service impacts via unusually large system time changes under certain conditions.
- The issue can be mitigated via the existing configuration options.

Recommendation:

- Consider implementing stricter limits for forward time changes in the default setting.
- Make the current forward-unbounded limit a well-documented alternative configuration.
- Evaluate potential additional opt-in safety mechanisms to logically restrict time adjustments and mitigate attacks.
- For example, the timestamp of the ntp-daemon program compilation could be useful as a lower bound for time adjustments when running with production settings.

3.2 NTP-061 — Insufficient Handling of Parsing Failures in read_json()

Vulnerability ID: NTP-061 **Status:** Resolved

Vulnerability type: CWE-248: Uncaught Exception

Threat level: Low

Description:

`read_json()` does not correctly handle parsing failures, causing panics if malformed data is processed.

Technical description:

This issue derives from the use of `unwrap()` on `serde_json::from_slice()` in sockets.rs.

The `serde_json` crate documentation lists the following:

```
/// This conversion can fail if the structure of the input does not match the
/// structure expected by `T`, for example if `T` is a struct type but the input
/// contains something other than a JSON map. It can also fail if the structure
/// is correct but `T`'s implementation of `Deserialize` decides that something
/// is wrong with the data, for example required struct fields are missing from
/// the JSON map or some number is too big to fit in the expected primitive
/// type.
```

Proof-of-concept no.1 via modified unit test:

`async fn write_then_read_is_identity()`

```
-  write_json(&mut writer, &object).await.unwrap();
+  // write data that cannot be parsed
+  let mut data = [0; 24];
+  writer.write_all(&data).await;
```

```
running 1 test
thread 'sockets::tests::write_then_read_is_identity' panicked at 'called `Result::unwrap()` on an
 `Err` value: Error("expected value", line: 1, column: 1)', ntp-daemon/src/sockets.rs:27:39
```

Proof-of-concept no.2:

1. Start the ntp-daemon with the following configuration file:

```
# dummy peer
[[peers]]
addr = "localhost:123"

[configure]
path = "/run/ntpd-rs/configure"
mode = 0o666
```

2. Trigger the vulnerability with the following Python program:

```
import socket

sock = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
sock.connect("/run/ntpd-rs/configure")
# crash the target by sending malformed json
sock.send(b'poc')
sock.close()
```

3. Observe crash:

```
thread 'tokio-runtime-worker' panicked at 'called `Result::unwrap()` on an `Err` value:
 Error("expected value", line: 1, column: 1)', [...]/ntp-daemon/src/sockets.rs:27:39
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
Aborted
```

Retest Status

During the review period, the developers fixed this issue via PR664, which makes the exception behavior non-fatal.

Log example:

```
ERROR ntp_daemon::config::dynamic: could not parse data on socket: Custom { kind: InvalidInput,
 error: Error("expected value", line: 1, column: 1) }
```

Impact:

- A local attacker with write access to the `configure` socket can crash ntp-daemon.
- Note that write access to the `configure` socket implies the ability to shut down the server via other means.
- This issue also applies to ntp-ctl and ntp-metrics-exporter.

Recommendation:

- Make the JSON parsing handling more robust.

3.3 NTP-060 — Log Pollution Issues

Vulnerability ID: NTP-060 **Status:** Unresolved

Vulnerability type: CWE-779: Logging of Excessive Data

Threat level: Low

Description:

The ntpd-rs logs warnings per received packet for some unusual conditions, which represents an opportunity for attackers to trigger excessive logging.

Technical description:

Variant 1

Receiving NTP packets with over 1024 bytes of UDP payload in length results in one warning log per packet.

```
WARN serve{rate_limiting_cutoff=0ns addr=1.2.3.4:123}:recv{local_addr=1.2.3.4:123 peer_addr=None
 buf_size=1024}: ntp_udp::raw_socket::recv_message: truncated packet because it was larger than
 expected max_len=1024
```

Packet payload data definition in Python:

```
data = b"\x00" * 1024
```

Overall length of the packet is 1068 byte including TCP and UDP headers.

Variant 2

Receiving packets with invalid version number header results in an `INFO` level log entry per packet.

```
serve{rate_limiting_cutoff=0ns addr=1.2.3.4:123}: ntp_daemon::server: received invalid packet:
 Invalid version 0
```

Variant 3

```
sudo hping3 127.0.0.1 --udp -a 10.1.0.1 -n -p 123 --data 48 --file ntp_one.raw -i u1000
```

```
hd ntp_one.raw -v
00000000  e3 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030
```

Observed with and without rate limiting

Config:

```
log-filter = "warn"
log-format = "full"

[[server]]
addr = "127.0.0.1:123"
rate-limiting-cache-size = 10000
rate-limiting-cutoff-ms = 1000

[[peers]]
mode = "pool"
addr = "pool.ntp.org"
max-peers = 4
```

```
[...]
WARN ntp_daemon::server: Could not send response packet error=Os { code: 22, kind: InvalidInput,
 message: "Invalid argument" }
WARN ntp_daemon::server: Could not send response packet error=Os { code: 22, kind: InvalidInput,
 message: "Invalid argument" }
```

```
WARN ntp_daemon::server: Could not send response packet error=Os { code: 22, kind: InvalidInput,
 message: "Invalid argument" }
[...]
```

Currently only observed if server is listening on localhost and the spoofed packets come in from a non-localhost address. Due to the required raw socket, this attack requires elevated privileges.

Other notes

`log-format` variations don't help mitigating this issue since there's still one log entry per event.

In the log excerpts, the local ntp-daemon server IP address was replaced with `1.2.3.4`.

Retest Status

During the review period, the ntpd-rs developers proposed PR675 which improves the behavior for one of the identified issues by lowering the log level of a message. Since this PR was not yet merged and the other issues were not yet addressed, the issue is marked as unresolved.

Impact:

- An remote attacker may fill system logs with meaningless information.
- Under some conditions, this can obstruct or prevent the detection of problem conditions.

Recommendation:

- Lower the log verbosity level of edge cases that are in reach of a remote attacker.

3.4 NTP-059 — TLS Certificate Pinning not Available

Vulnerability ID: NTP-059 **Status:** Unresolved

Vulnerability type: Certificate Handling

Threat level: Low

Description:

ntpd-rs does not allow pinning of specific TLS keys. If implemented, this feature could be used in some well-controlled client and server instances to provide additional protection.

Technical description:

`rustls` lists certificate pinning under the currently not implemented but possible future features. The `rustls` implementation ticket suggests that native support for this feature will not be rolled out soon and requires additional crates for the foreseeable future.

Due to the operational challenges of pinning certificates, particularly when pinning individual server certificates and not intermediary certificates, we expect that this feature could be used primarily for internal deployments where both the NTS client- and server configuration are controlled by one entity.

Impact:

- At the moment, NTS client trust in TLS certificates can not be limited to a particular TLS certificate or intermediary Certificate Authority certificate.
- This may simplify attacks under some conditions, for example after a subdomain takeover of a timeserver domain.

Recommendation:

- Consider adding TLS pinning functionality.
- The rustls-pin crate may offer the required functionality.

3.5 NTP-058 — File Deletion Risk via Socket File Configuration

Vulnerability ID: NTP-058	**Status:** Resolved
Vulnerability type: CWE-703: Improper Check or Handling of Exceptional Conditions	
Threat level: N/A	

Description:

An attacker with write access to the ntpd-rs configuration may leverage the daemon communication sockets to trash arbitrary files on the system under some conditions.

Technical description:

Proof-of-concept

Configuration:

```
# one peer is needed to satisfy startup conditions
[[peers]]
addr = "localhost:123"

[observe]
# replace the targeted file with a socket
path = "/etc/proof-of-concept"
```

Step 1 - Run the following commands as root to create a root-owned file target with some content:

```
echo "important content" > /etc/proof-of-concept
chmod 600 /etc/proof-of-concept
```

```
ls -la --time-style=+ /etc/proof-of-concept
-rw------- 1 root root 18  /etc/proof-of-concept
```

Step 2 - Start ntp-daemon with root permissions and the provided attack configuration.

Step 3 - Observe impact of deletion/overwrite:

```
ls -la --time-style=+ /etc/proof-of-concept
srwxrwxrwx 1 root root 0  /etc/proof-of-concept
```

Since the resulting file is still a socket, the wide `0o777` permissions are not sufficient for other lower-privileged users on the system to further modify the file. This obstructs attacks which aim to replace system files with specific crafted content.

Note that at the time of testing, the attack does not succeed if ntpd-rs runs as a systemd service with reduced user permissions. It is unclear if the socket mechanism is generally non-functional under this daemon startup situation or if this is due to effective mitigations.

Retest Status

During the review period, the developers added a mitigation via PR665, which prevents the deletion of files that aren't sockets.

Log excerpt of new behavior:

```
WARN ntp_daemon::observer: Abnormal termination of the state observer: path "/etc/proof-of-concept"
 exists but is not a socket
WARN ntp_daemon::observer: The state observer will not be available
```

Impact:

- An attacker with write permissions to the ntpd-rs configuration file may misuse the socket functionality to delete arbitrary files on the filesystem.
- For maximum effect, the attack requires ntpd-rs to run as root.

- Issue NTP-034 (page 40) may provide opportunities for attacker-controlled configuration files.

Recommendation:

- Change the socket creation logic to error out if the target file exists and is a regular file.
- Evaluate if restricting the socket creation to `/run/` or other designated directories is acceptable for operators.

3.6 NTP-056 — Daemon Runs Despite Unrecoverable Errors at Startup

Vulnerability ID: NTP-056 **Status:** Resolved

Vulnerability type: CWE-703: Improper Check or Handling of Exceptional Conditions

Threat level: Low

Description:

ntpd-rs starts up in a degraded state when encountering bootstrapping problems that obstruct the correct operation of subcomponents. This behavior may be unexpected and unwanted to operators.

Technical description:

Example 1

NTS related issues result in a running daemon without NTS server:

```
ERROR ntp_daemon::keyexchange: Abnormal termination of NTS KE server: error reading cert_chain_path
 at `"test-keys/end.fullchain.pem"`: Os { code: 2, kind: NotFound, message: "No such file or
 directory" }
```

Example 2

Socket related issues result in a running daemon without sockets:

```
ERROR ntp_daemon::observer: Abnormal termination of state observer: Could not create observe socket
 at "/run/ntpd-rs/observe" because its parent directory does not exist
```

Note that ntpd-rs correctly detects some configuration file issues, leading to informative errors and safe abort handling in those cases:

```
There was an error loading the config: config toml parsing error: TOML parse error at line 8, column
 1
  |
8 | [[peers]]
```

```
  | ^^^^^^^^^
error while parsing certificate file "test-keys/testca.pem": Os { code: 2, kind: NotFound, message:
 "No such file or directory" }
```

Exit code:

```
echo $?
78
```

Retest Status

The ntpd-rs developers resolved this issue during the review period via PR673.

Impact:

- Operating the daemon with a subset of the requested functionality may lead to degraded service with unreliable or unsafe operations.

Recommendation:

- Cleanly exit with error code exit status after encountering fatal problems during startup.

3.7 NTP-054 — NTS Client Panics on Refused Connection

Vulnerability ID: NTP-054	**Status:** Resolved
Vulnerability type: CWE-248: Uncaught Exception	
Threat level: Elevated	

Description:

The NTS client logic in ntpd-rs aborts on problematic network conditions that can be triggered by a remote attacker.

Technical description:

The ntpd-rs NTS key exchange logic of the reviewed version has a fragile TCP connection handling. The malicious or accidental interference with TCP connections between the NTS client and the NTS server leads to a complete crash of the daemon on the client side. This can be leveraged by a Machine-in-the-Middle (MitM) attacker for reliable attacks.

Example log:

```
DEBUG ntp_daemon::system: updating peer msg=NetworkIssue(PeerId(3))
[ntp-daemon/src/keyexchange.rs:28] "connecting" = "connecting"
thread 'tokio-runtime-worker' panicked at 'called `Result::unwrap()` on an `Err` value: Os { code:
 111, kind: ConnectionRefused, message: "Connection refused" }', ntp-daemon/src/keyexchange.rs:31:10
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
Aborted
```

Proof-of-Concept:

1. Set up a `ntpd-rs` NTS daemon acting as a peer towards some external server (e.g., `time.cloudflare.com:4460`)
2. Ensure NTS traffic will fail on the network path, for example with an ICMP Destination Unreachable response via `sudo iptables -I OUTPUT -p tcp --dport 4460 -j REJECT`
3. Start the ntp-daemon.
4. Observe ntp-daemon program abort via panic.

Note that the vulnerability is triggered by other reasons for `Err value: Os { code: 101, kind: NetworkUnreachable, message: "Network is unreachable" }'` such as dropped packets (`-j DROP`) instead of ICMP Destination Unreachable responses (`-j REJECT`). For the proof-of-concept, this method is simply faster to trigger the problematic result.

Example configuration:

```
[[peers]]
mode = "NtsServer"
ke-addr = "time.cloudflare.com:4460"
```

Additional log information for crash on bootstrapping:

```
INFO ntp_daemon::system: Running spawner id=SpawnerId(1) ty="nts" addr="localhost:4460"
[ntp-daemon/src/keyexchange.rs:28] "connecting" = "connecting"
thread 'tokio-runtime-worker' panicked at 'called `Result::unwrap()` on an `Err` value: Os { code:
 111, kind: ConnectionRefused, message: "Connection refused" }', ntp-daemon/src/keyexchange.rs:31:10
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
Aborted
```

This issue was resolved during the security review via improved exception handling in PR646.

The client behavior is now more robust and retries failing NTP connections every second:

```
WARN ntp_daemon::spawn::nts: error while attempting key exchange error=Io(Os { code: 111, kind:
 ConnectionRefused, message: "Connection refused" })
WARN ntp_daemon::spawn::nts: error while attempting key exchange error=Io(Os { code: 111, kind:
 ConnectionRefused, message: "Connection refused" })
```

Impact:

- Persistent denial of service of the ntpd-rs daemon running the NTS client.

- A machine-in-the-middle (MitM) attacker on the network path can use this to crash ntp-daemon instances.
- This issue can be triggered accidentally and likely reduces overall daemon robustness when using NTS.

Recommendation:

- Ensure the NTS logic is robust against temporary unavailable peers or network failure.
- Establish regular testing under non-optimal or adverse network conditions.
- Consider adding automated tests running on simulated lossy or unreliable networks.

3.8 NTP-053 — NTS Client Logs Disclose Sensitive Security Cookies

Vulnerability ID: NTP-053 **Status:** Resolved

Vulnerability type: CWE-532: Insertion of Sensitive Information into Log File

Threat level: Low

Description:

Log operations which print the PeerNTSData reveal secrets in the form of NTS cookies.

Technical description:

Example log (shortened)

```
WARN spawn{index=PeerId(2) addr=[::1]:123 network_wait_period=1s nts=Some(PeerNtsData { cookies:
 CookieStash { cookies: [[0, 0, 0, 0, 0, 146, 214, 93, 85, 90, 176, 23, 51, 189, 180,
[...]
]], read: 0, valid: 8 } })}: ntp_daemon::peer: poll message could not be sent error=Os { code: 111,
 kind: ConnectionRefused, message: "Connection refused" }
```

Retest Status

During the review period, the ntpd-rs developers fixed this issue via PR645, which only prints the number of available cookies in the `CookieStash` instead of their content.

Improved log behavior:

```
DEBUG spawn{index=PeerId(1) addr=[::1]:123 interface=None enable_timestamps=EnableTimestamps
 { rx_software: true, tx_software: false, rx_hardware: false, tx_hardware: false }
```

```
network_wait_period=1s nts=Some(PeerNtsData { cookies: CookieStash { cookies: 8, read: 0, valid:
8 } })}: ntp_daemon::peer: wait completed
```

Impact:

- An attacker with access to the log information may theoretically leverage the cookies to perform attacks against the client or the server.
- We are not aware of practical attacks that significantly benefit from access to the security cookies.

Recommendation:

- Reconfigure the log operation to not print `CookieStash` information by default.
- If necessary, include an opt-in configuration setting (ideally at compile time) to print sensitive information for debugging.

3.9 NTP-051 — Unresponsive Peers are not Handled Correctly

Vulnerability ID: NTP-051	**Status:** Resolved
Vulnerability type: CWE-693: Protection Mechanism Failure	
Threat level: Moderate	

Description:

ntpd-rs does not correctly handle peers that are unreachable or exhibit other behavior issues, increasing the risk of time synchronization failures or opportunities for manipulation.

Technical description:

Scenario: During normal ntpd-rs operation, one or multiple NTP reference sources of a pool go unresponsive (for benign or malicious reasons).

Expectation: ntpd-rs should perform a failover to other sources, namely by attempting to reach new trusted upstream peers from the pool via DNS requests. Additionally, ntpd-rs should log and warn about the situation appropriately.

Setup:

- ntpd-rs allowed to start up normally and acquire its peers from the pool.

- Unresponsive peers simulated by dropping all incoming packets `iptables -I INPUT 1 -s 0.0.0.0/0 -j DROP` during runtime
- Network communication monitored with `wireshark` to watch for changes in behavior, namely new DNS requests to the pool to fetch new peer addresses for failover.
- IPv4 only setup on a local VM.

Configuration:

```
# Other values include trace, debug, warn and error
log-filter = "info"

# Allowed values: full, compact, pretty, json
log-format = "full"

[...]

[[peers]]
mode = "pool"
addr = "pool.ntp.org"
max-peers = 4

[system]
min-intersection-survivors = 1
```

Observations:

- ntpd-rs generically warns about unreachable peers after some time passes on unanswered requests.
- No failover occurs within several hours of observation time despite the minimum survivors number not being met and no possibility for timesync events (!).
- This behavior was present both when triggering the iptables rule during startup before the chosen peers are firmly established (stratum 16 status on the daemon), as well as after startup once the peers were fully initialized.

Log:

```
2023-03-28T17:27:53.218602Z  WARN ntp_proto::peer: Peer unreachable
2023-03-28T17:28:17.236714Z  WARN ntp_proto::peer: Peer unreachable
[...]
2023-03-28T19:31:49.010147Z  WARN ntp_proto::peer: Peer unreachable
2023-03-28T19:32:23.581694Z  WARN ntp_proto::peer: Peer unreachable
[...]
2023-03-28T23:20:38.975546Z  WARN ntp_proto::peer: Peer unreachable
```

Observability socket information (different run):

```
[
  {
    "Observable": {
      "offset": 0.0010424726645095444,
      "uncertainty": 0.0004805072677509176,
      "delay": 0.017931963786932632,
      "remote_delay": 0.012207031252842171,
```

```
      "remote_uncertainty": 0.027816772467414097,
      "last_update": {
        "timestamp": 16703466299683936427
      },
      "reachability": 0,
      "poll_interval": 4,
      "peer_id": 2297678454,
      "address": "pool.ntp.org:123"
    }
  },
[...]
```

- The `"reachability": 0,` implies the peers are internally recognized as unresponsive.

Problematic behavior:

- No log information (via INFO/WARN/ERROR) on which peer is misbehaving (Peer ID, addr).
- No log information that the required minimum of peers isn't responding.
- No log information that time sync is effectively lost since no `Filter progressed` event occurs.
- No attempt at re-establishing time sync via new DNS lookups from the pool.

Additional remarks:

- Note that the behavior is different if packets are dropped immediately at ntpd-rs startup and the initial peer acquisition via DNS queries is affected.

This issue was resolved during the review via PR649. After detecting that a peer from a particular NTP pool becomes unavailable, failover to a new peer from this pool is performed. Note that at the time of the retest, the logging handling has not been improved.

Impact:

- Due to typical requirements on the minimum number of timeserver peers, this issue may simplify Denial of Service (DoS) attacks.
- This issue can be triggered by non-malicious conditions, therefore we suspect it reduces the overall timesync resilience if `ntpd-rs` doesn't automatically recover from peer issues.

Recommendation:

- Correct or improve the peer handling logic with regards to observable peer issues.
- Improve logging to warn operators of serious error conditions such as loss of time sync.
- Improve logging to assist operators with identifying problematic peer servers.

- This is partially related to NTP-050 (page 32).

3.10 NTP-050 — Consider Adding a Mechanism to Cycle Pool Servers

Vulnerability ID: NTP-050 **Status:** Unresolved

Vulnerability type: Peer handling and trust

Threat level: Low

Description:

ntpd-rs currently has no mechanism to rotate or re-evaluate established NTP peers which are sourced from an NTP pool as long as they're seen as active and well-behaved by the daemon.

Technical description:

NTP pools are a commonly used concept to cluster multiple available NTP servers.

For the purposes of this finding, a pool address can be understood as a Domain Name Server (DNS) domain that allows clients to query a subset of IPv4 or IPv6 addresses of available NTP servers in the pool cluster via standard DNS methods.

In our observation, `ntpd-rs` triggers initial DNS queries (`A`/`AAAA` records) when attempting to acquire the minimum number of NTP peers from a configured pool during bootstrapping at startup. However, there currently is no built-in mechanism or configuration to ask the ntpd-rs daemon to re-acquire new valid addresses from the configured pool and switch peers at runtime without forcing a daemon restart. Therefore, once a peer connection is established, a long-running ntpd-rs daemon will continue using it forever unless severe reachability issues are detected.

Unlike other uses of DNS, the originally fetched name records are effectively used beyond their specified DNS time to live (TTL) field on purpose. By our understanding, this TTL field is not available to signal the intended use time of an NTP server associated with the pool due to typical use Round-robin DNS load balancing mechanisms.

The popular `pool.ntp.org` project operates with volunteer-provided servers. Servers could get removed from the pool for a number of reasons such as discontinuation of NTP service, planned maintenance or as security measure after the detection of malicious behavior. An initial public assignment of a given NTP server by the pool's DNS server towards clients implies some level of trust and fitness, but is only a snapshot in time and trust or fitness are not implied forever. For load balancing reasons, pools are expected to regularly switch announced NTP server addresses for a given pool address. This intentional lack of stability for IP resolution mean that additional DNS queries to the pool and comparisons of resolved IP addresses cannot be used efficently to establish whether a given NTP server IP address is still served by the pool.

This conflict raises the question of security- and reliability implications of indefinite peering with NTP servers without a mechanism to re-evaluate the trust and fitness indicated by the pool authority into a given peer.

Impact:

- ntpd-rs instances may continue using an NTP peer with publicly known issues or problematic behavior that has been removed from a pool.

Recommendation:

- We recommend further investigation into mechanisms and relevant trade-offs to re-acquire NTP peers from a pool.
- Solution ideas include per-pool configuration of time-to-live settings for peers, combined with staggered peer refresh and some randomization.
- Since peer changes may temporarily influence the time sync precision, this behavior should be configurable.

3.11 NTP-043 — Lack of OCSP Handling for NTS Endpoint

Vulnerability ID: NTP-043 **Status:** Unresolved

Vulnerability type: CWE-299: Improper Check for Certificate Revocation

Threat level: Low

Description:

ntpd-rs currently does not support Online Certificate Status Protocol (OCSP) stapling or verification due to missing features of the Rust TLS cryptography libraries.

Technical description:

Based on available documentation, ntpd-rs does not support Online Certificate Status Protocol (OCSP) stapling in a server role or OCSP verification in an NTS client role. If correct, this introduces some risk in case NTS TLS certificates are compromised and revoked, but the revocations are not acted upon.

Documentation and related tickets:

- https://github.com/rustls/rustls/issues/31
- https://github.com/briansmith/webpki/issues/26

- https://github.com/rustls/rustls/issues/888
- https://github.com/khonsulabs/fabruic/issues/1

Note that according to the changelog and list of current features, the `rustls` library does support `Allow OCSP and SCT stapling for servers.` since `0.10.0 (2017-08-12)`. However, the existing ticket 31 suggests the implementation is not complete from a usability perspective. Furthermore, OCSP related improvements on the Rust side are dependent on future standard TLS library improvements.

"OCSP verification by clients" is not supported by `rustls` at the moment and officially listed under "Possible future features".

During discussion, the `ntpd-rs` developers outlined the additional difficulty associated with TLS certificate related verification steps such as OCSP verification when done from NTS clients that have an inaccurate system time clock during bootstrapping. This may add some additional complexity to the task of integrating OCSP verification on the client side.

Impact:

- The ntpd-rs NTS client functionality cannot evaluate the revocation status of TLS certificates received from peers, increasing risks.
- The ntpd-rs NTS server cannot, by our understanding, provide NTS peers with recently signed assurances of their TLS certificate's validity via OCSP stapling.

Recommendation:

- Investigate documenting potential workarounds on the server side using TLS proxy systems that support OCSP stapling.

3.12 NTP-041 — Too Broad Default Socket File Permissions

Vulnerability ID: NTP-041	**Status:** Unresolved
Vulnerability type: CWE-276: Incorrect Default Permissions	
Threat level: Low	

Description:

The `ntpd-rs` socket default file permissions are too lax.

Technical description:

For configuration sockets, we recommend switching the default value from `770` to `660` or `600` (preferred, if ownership is adjustable).

For observation sockets, we recommend switching the default value from `777` to `666`, `660` or `600` (preferred, if ownership is adjustable).

See the CONFIGURATION.md documentation for details.

We recommend additional validation to ensure the recommended change is compatible with other systems such as FreeBSD and MacOS.

Retest Status

During the review period, the `ntpd-rs` developers implemented PR670, which removes the executable permission from the socket permissions. At the time of review, the exact handling of folder permissions, the daemon system user and other details are not specified yet, making it difficult to assess whether the current "group" and "other" permissions are still too lax.

We leave this issue marked unresolved since additional future retest steps are advised.

Impact:

- A local attacker may have access to the socket endpoints under limited conditions.

Recommendation:

- Adopt a less permissive default configuration.
- Investigate potential benefits of configurable socket file ownership to reduce the need of permissive read+write access for other local users.

3.13 NTP-040 — Weak Rate Limiting System on IPv6

Vulnerability ID: NTP-040	**Status:** Unresolved
Vulnerability type: CWE-799: Improper Control of Interaction Frequency	
Threat level: Moderate	

Description:

In IPv6 environments, performing rate limiting for NTP based on individual IP addresses is difficult since attackers can send and receive packets from a large number of IPv6 addresses to trigger rate limiting exceptions.

Technical description:

In public IPv4 networks, remote attackers likely only have access to a very limited number of individual addresses they can both send and receive UDP packets from.

On IPv6 networks, a remote attacker can plausibly control a large number of globally routed individual addresses via a `/64` subnet (`2^64` addresses), `/56` subnet (`2^72` addresses) or even `/48` subnet (`2^80` addresses) that is allocated to them by their provider.

Spoofing UDP packets with fake sender addresses is therefore not strictly necessary to bypass per-IP-address based rate limiting mechanisms on IPv6-enabled targets. We think that it is difficult to effectively rate-limit incoming NTP packets at the server without additional costly network lookups or potentially resource-intensive memory representations. We recommend further evaluation of potential solutions and server-side statistics, e.g., with bounded memory options for rate limiting buckets or lists, due to the increasing importance of safe operation on public IPv6 networks with growing worldwide IPv6 adoption.

Impact:

- The currently chosen rate limiting design does not scale well against attacks from IPv6 networks.
- An attacker with access to a basic `/64` subnet can trigger and observe hash table collisions in the victim server without the need to spoof UDP packets originating from non-controlled networks.
- By doing so, they can bypass the current rate limiting mechanism even if other reported implementation weaknesses are resolved.

Recommendation:

- Within the current system and protocol constraints, we have not been able to determine and recommend a clearly superior rate limiting design.
- Increasing the rate limit bucket size to interactions per `/64` IPv6 subnet (or larger) can correlate individual addresses in an attack under some conditions, but risks overblocking legitimate peer interaction.
- Improve the general visibility of attacks and attack characteristics to operators via logging and other methods.
- Improve the documentation and handling of allow-list and deny-list functionality or recommend firewall-based alternatives.

3.14 NTP-039 — Weak Rate Limiting System due to Port Number Inclusion

Vulnerability ID: NTP-039	**Status:** Resolved
Vulnerability type: CWE-799: Improper Control of Interaction Frequency	
Threat level: Moderate	

Description:

The rate limiting implementation of `ntpd-rs` uses the UDP source port number and IP address combination to identify a peer, making it unable to correlate multiple requests from the same IP address that use a different source port.

Technical description:

The reviewed ntpd-rs rate limiting implementation determines peers based on origin IP address and origin port number (`peer_addr`) instead of just the origin IP address (`peer_addr.ip()`).

This implementation detail severely limits the effectiveness of the rate limiting of clients which pick different UDP source addresses for individual requests. An attacker on a host with a dedicated single IP address has about 65k ports, and therefore the ability to be treated as 65k different peers by ntpd-rs. (On Unix hosts, this figure is close to 64.5k or 65.5k addresses depending on whether the attacker has network-related system privileges on the attacking host).

This was fixed in PR647 by switching from `TimestampedCache<SocketAddr>` to `TimestampedCache<IpAddr>`.

Impact:

- Remote peers that cycle through outgoing UDP ports between requests can evade or interfere with the rate limiting mechanism more easily.

- This attack mechanism can be combined with NTP-038 (page 38) for greater effect.

Recommendation:

- Switch to rate limiting based only on `IP` address information without port numbers.
- See also related challenges for rate limiting on IPv6-enabled servers in NTP-040 (page 36).

3.15 NTP-038 — Weak Rate Limiting System due to Insufficient Hashtable Index Randomization

Vulnerability ID: NTP-038	**Status:** Resolved
Vulnerability type: CWE-799: Improper Control of Interaction Frequency	
Threat level: Moderate	

Description:

`ntpd-rs` uses a hash table mechanism that insufficiently randomizes placement in the hash table and overwrites content on collisions. This allows attackers to predict table index locations and provoke collisions to bypass the rate limiting.

Technical description:

The reviewed version of ntpd-rs uses `std::collections::hash_map::DefaultHasher::default()` to place rate limit information of network peers in a hash map.

The Rust documentation for `DefaultHasher` states that `The internal algorithm is not specified, and so it and its hashes should not be relied upon over releases.`. At the time of writing the implementation uses `DefaultHasher(SipHasher13::new_with_keys(0, 0))`. The hasher logic therefore uses SipHash 1-3 with fixed initialization values.

Additionally, each hash table index calculation re-uses the initial hasher state to ensure deterministic placement of identical peers in identical locations.

This results in the problematic property that all `ntpd-rs` instances share identical hashing configurations, which allows an attacker to accurately pre-determine the expected hash table placement if the hash table size of the victim is known (e.g., default values or commonly picked sizes).

Since the hash table in use overwrites content on collisions and applies no rate limiting in this situation due to missing information on the new peer, this construction allows for targeted rate limiting bypasses.

Attack concept (simplified):

- The attacker picks two outgoing IP addresses `A` and `B` that map to the same hash table index `I`, which can be pre-computed due to the weakness.
- The attacker alternates between sending UDP NTP requests to the victim with the spoofed address `A` and `B`.
- Since addresses `A` and `B` collide in the hash map but are not identical, the victim server keeps accepting the received packets without rate limiting and re-writing the hash table entry at the index `I`.
- Rate limitation is bypassed for the described attacker traffic.

Known attack constraints:

- The attacker has to guess `rate-limiting-cache-size` hash table size (configured by the operator in `ntp.toml`). Under some conditions, an attacker may be able to determine the configured hash table size by testing different configurations and observing the victim server's responses for successful collisions.
- Theoretically, the rust DefaultHasher algorithm could change across `ntpd-rs` versions since it's not guaranteed to be stable by the standard.

Note that at the time of discovery of the issue, UDP source port numbers of NTP packets are factored into the index computation, which is an additional advantage to attackers. For more information, see NTP-039 (page 37) .

Note that the reviewed ntpd-rs source code contains the following rationale on hash collision handling:

```
[...]
/// The likelyhood of hash collisions can be controlled by changing the size of the cache. Overall,
/// hash collisions are not a big problem because problematic IPs will continue to show up, so if
/// we don't deny them on the first attempt because the previous entry was evicted, we're very
/// likely to succeed on the next request it makes.
```

However, this implicitly assumes that collisions only happen accidentally and that an attacker does not seek them out deliberately. Due to the birthday problem characteristics of the collision behavior in the hash table and well as the possibility of attackers to spoof (on IPv4) or directly control (on IPv6) NTP traffic from many different IP addresses, we think this assumption does not hold.

After discussion with the ntpd-rs developers, they've revised the implementation and documentation to perform per-startup hash table index randomization via `RandomState` in PR647 and document the remaining weakness of the rate limiting system against targeted attacks.

Note that the randomization of the hasher prevents an attacker from pre-computing two requests that will cause a collision, but does not reduce the likelihood of collisions or change their outcome. In scenarios where the attacker can observe the decisions of the rate limiting system, collisions can still be found by brute force and then used for rate

limiting bypasses until the next server restart of the targeted instance. This was confirmed experimentally during the retest.

Impact:

- The rate limit system can generally be circumvented by targeted attacks due to design limitations.
- The described lack of randomization makes the attack easier and more reliable to perform.

Recommendation:

- Ensure sufficient randomization of the hash table behavior to prevent targeted hash table collisions.
- Randomize the hash table placement on each server startup with sufficient entropy.
- Mitigations related to hash table size are subject to other technical trade-offs, see related findings.
- Add optional rate limiting behavior to drop requests instead of replying with a `RATE` response to reduce outgoing traffic during untargeted attacks.

3.16 NTP-034 — Problematic ntp.toml Local File Preference

Vulnerability ID: NTP-034 **Status:** Resolved

Vulnerability type: CWE-15: External Control of System or Configuration Setting

Threat level: Low

Description:

The `ntp-daemon` includes the `ntp.toml` configuration file preferably from the local working directory, if available. Under some edge conditions, this directory or file may not be sufficiently protected, giving unauthorized local users control over the configuration.

Technical description:

`ntp-daemon` first attempts to load the `ntp.toml` file from the local working directory. The absolute path `/etc/ntp.toml` is only checked if no local file exists.

The file search is done in the directory `ntp-daemon` is called from, and not relative to where the `ntp-daemon` binary is located. Users may not realize the potential negative security implications if the relevant `$PWD` is writeable by

other users. This could create opportunities for a local attacker to place a new `ntp.toml` file or overwrite an existing `ntp.toml` file in that path if the directory is not sufficiently protected.

Retest Status

The issue was fixed during the security review via PR651, which is summarized as `Change configuration paths to only consider /etc/ntpd-rs/ntp.toml by default`.

Impact:

- Under some edge conditions, a local attacker may be able to manipulate the server configuration.

Recommendation:

- Remove the local default inclusion of `./ntp.toml`.
- Users can still use the existing configuration flag to emulate this behavior via `--config ./ntp.toml` if needed.

3.17 NTP-032 — Remote Denial of Service in demobilize-server Test Binary

Vulnerability ID: NTP-032	**Status:** Unresolved
Vulnerability type: CWE-248: Uncaught Exception	
Threat level: N/A	

Description:

Remote attackers can crash the `demobilize-server` with crafted network communications. This server is only for testing purposes and not part of the main `ntpd-rs`.

Technical description:

Steps to reproduce:

1. Within the source code, change the listening port to `UdpSocket::bind("0.0.0.0:123")` so that standard NTP tools can talk to the demobilize test server.
2. Start the `demobilize-server`.
3. Run `ntpdate -q localhost`.

4. Observe crash.

Log output:

```
48 bytes received from 127.0.0.1:35444
thread 'main' panicked at 'called `Result::unwrap()` on an `Err` value: Os { code: 89, kind:
 Uncategorized, message: "Destination address required" }', test-binaries/src/bin/demobilize-
server.rs:28:44
stack backtrace:
   0: rust_begin_unwind
             at /rustc/2c8cc343237b8f7d5a3c3703e3a87f2eb2c54a74/library/std/src/panicking.rs:575:5
   1: core::panicking::panic_fmt
             at /rustc/2c8cc343237b8f7d5a3c3703e3a87f2eb2c54a74/library/core/src/panicking.rs:64:14
   2: core::result::unwrap_failed
             at /rustc/2c8cc343237b8f7d5a3c3703e3a87f2eb2c54a74/library/core/src/result.rs:1790:5
   3: core::result::Result<T,E>::unwrap
             at /rustc/2c8cc343237b8f7d5a3c3703e3a87f2eb2c54a74/library/core/src/result.rs:1112:23
   4: demobilize_server::main::{{closure}}
             at ./test-binaries/src/bin/demobilize-server.rs:28:19
[...]
```

Impact:

- Remote Denial of Service (DoS) via `panic!`.
- The affected program is part of `test-binaries` and unrelated to the normal `ntpd-rs`.

Recommendation:

- Improve the exception error handling.
- Perform additional testing of the `demobilize-server` behavior.

3.18 NTP-029 — webpki Dependency Apparently Unmaintained and with Known Issue

Vulnerability ID: NTP-029 **Status:** Resolved

Vulnerability type: CWE-1104: Use of Unmaintained Third Party Components

Threat level: Low

Description:

`ntpd-rs` directly depends on the `webpki` crate, which appears to be unmaintained and has at least one known issue.

Technical description:

`ntpd-rs` uses the `webpki` crate in the latest release `0.22.0` from 10.4.2021.

Known Security Issue:

- `TLS-01-003 Webpki: Support for Non-Contiguous Subnet Masks (Low)` reported in Security Review & Audit Report rustls 05.-06.2020 by Cure53.
- Vulnerable code for `TLS-01-003` is still present in newest webpki version src/name/ip_address.rs.
- This problem is fixed in the `rustls-webpki` fork, see rustls webpki PR18.

The `rustls` developers forked `webpki` as `rustls-webpki`, released the first stable version `0.100.0 (2023-03-13)` and switched to it for rustls `0.21.0`. This confirms the general security concern about the `webpki` maintenance status and offers a straightforward solution.

Retest Status

During the review, the ntpd-rs developers switched to the `rustls 0.21.0` version via PR643 and removed the direct `webpki` dependency entry in `Cargo.toml` via commit 577cac17c4b47995a8b07536af7d5286879337a3. Note that other `ntpd-rs` dependencies currently still require `webpki` transitively, but the overall reliance on it has decreased.

Impact:

- General risk of using unmaintained cryptography-related software.
- Overall practical impact is expected to be low to none.

Recommendation:

- Remove the explicit `webpki` dependency from `Cargo.toml` list of dependencies.
- Switch to the `rustls-webpki` fork once possible.
- Follow up on indirect dependency requirements for `webpki`.

3.19 NTP-024 — Perform Safe Memory Cleanup of Key Material after Use

Vulnerability ID: NTP-024 **Status:** Resolved

Vulnerability type: CWE-693: Protection Mechanism Failure

Threat level: Low

Description:

`ntpd-rs` does not explicitly overwrite cryptographic key material in memory after use. This increases the risk of compromise in the event that some other unspecified issue discloses program memory to attackers.

Technical description:

We recommend usage of the Rust crate `zeroize` functionality to improve this behavior. The library is part of `RustCrypto/utils` and also in use by other essential Rust crypto crates that ntpd-rs depends on.

Cryptographic material:

- NTS TLS private key and intermediary copies.
- Ephemeral AES-GCM-SIV keys.
- Security cookie values (limited).

Tracked upstream as Issue 617. During the review period, the ntpd-rs developers implemented PR623. Note that we did not evaluate the coverage level of the current protections in depth, in part since the information in question is stored in cleartext on the filesystem or transferred in cleartext over the network as part of normal operation.

We expect some remaining difficulty for `zeroize`-based memory cleanup in panic conditions, since ntpd-rs defaults to a quick abort that does not allow for unwinding actions.

Additional references

- https://benma.github.io/2020/10/16/rust-zeroize-move.html

Impact:

- It is best practice to securely overwrite cryptographic secrets after use.
- This defense-in-depth measure limits the impact of information leaks of process memory.

Recommendation:

- Consider using the `zeroize` Rust crate to implement memory sanitization actions for important program secrets.

3.20 NTP-021 — Cargo SSH Host Key Verification Vulnerable to CVE-2022-46176

Vulnerability ID: NTP-021	**Status:** Unresolved
Vulnerability type: CWE-347 Improper Verification of Cryptographic Signature	
Threat level: N/A	

Description:

The cargo tool in Rust versions before `1.66.1` does not verify SSH host keys, which allows machine-in-the-middle (MitM) attacks on repository cloning. A vulnerable version is used in the Continuous Integration (CI) infrastructure, but the affected functionality is not triggered.

Technical description:

- https://github.com/advisories/GHSA-r5w3-xm58-jv6j
- https://ubuntu.com/security/CVE-2022-46176
- The `ntpd-rs` Continuous Integration (CI) uses cargo version `1.65.0` in .github/workflows/build.yaml.

Note the mitigating circumstances, which reduce the attack surface. Additionally, this specific Rust version with the vulnerable Cargo software is tested since it represents the current Minimum Supported Rust Version (MSRV) of the `ntpd-rs` project.

Impact:

- A vulnerable cargo version is used in some CI configurations, however the affected SSH functionality is not triggered.
- No practical impact expected.

Recommendation:

- If possible, use security backports for Cargo or upgrade to patched Rust versions.
- Document the issue to avoid accidental use of affected functionality.

4 Non-Findings

In this section we list some of the things that were tried but turned out to be dead ends, as well as further documentation on security considerations and bugs.

4.1 NF-063 — False Positive Warning for Peer Number on Default Config

The logic check for the configured number of NTP peers does not correctly take NTP pool configurations into account. This leads to a false positive warning of `Fewer peers configured than are required to agree on the current time. Daemon will not do anything.`, which is a bug.

Log example:

```
INFO ntp_daemon::config: using global config file at default location `/etc/ntpd-rs/ntp.toml`
WARN ntp_daemon::config: Fewer peers configured than are required to agree on the current time.
 Daemon will not do anything.
INFO ntp_daemon::system: Running spawner id=SpawnerId(1) ty="pool" addr="pool.ntp.org:123 (4)"
INFO ntp_daemon::system: new peer peer_id=PeerId(1) addr=176.221.42.125:123 spawner=SpawnerId(1)
INFO ntp_daemon::system: new peer peer_id=PeerId(2) addr=188.68.59.79:123 spawner=SpawnerId(1)
INFO ntp_daemon::system: new peer peer_id=PeerId(3) addr=213.209.109.45:123 spawner=SpawnerId(1)
INFO ntp_daemon::system: new peer peer_id=PeerId(4) addr=45.9.61.155:123 spawner=SpawnerId(1)
```

Example config file:

```
[[peers]]
mode = "Pool"
addr = "pool.ntp.org"
max-peers = 4
```

Note the absence of `min-intersection-survivors` parameter. The minimum required number of peers for successful synchronization should therefore be `3` via the default value, which is satisfied by four peers from the pool.

Unrelated to this particular bug, we have several minor recommendations related to this functionality:

- Consider increasing the log level of the `Fewer peers configured [...]` to `ERROR` due to the significant implications of non-functional time synchronization.
- Clarify expectations beyond `Daemon will not do anything.` in the warning, since the ntp-daemon will in fact perform many steps such as peer communication when started in this mode.
- Increase the log level of `INFO ntp_proto::algorithm::kalman: No concensus cluster found` to make time synchronization problems at runtime more visible.
- Consider logging a reminder to operators in case `min-intersection-survivors` is equal to the number of available peers. While this is not a fatal configuration, any single server failure will obstruct time synchronization, making it less robust in error cases.

During the review period, the ntpd-rs developers resolved the false positive warning via PR666.

4.2 NF-052 — NTP Client Poll Interval is Documented Incorrectly

The documentation specifies `poll-interval-limits` as the configuration flag which influences the poll interval.

During evaluation, it was noticed that this configuration value is ineffective and not acted upon by the daemon. Further analysis together with the `ntpd-rs` developers revealed that the parameter name was incorrectly documented and `poll-limits` should be used instead.

We recommend correcting of the documentation and clarifying the parameter implications on NTP client- or server connections.

Additionally, we recommend providing additional documentation guidance for operators regarding valid and reasonable minimum/maximum values for this parameter with regards to NTP specification and practical server behavior, for example with the goal of accelerating time synchronization on startup without being too aggressive on NTP servers.

During the review period, the ntpd-rs developers fixed the documentation issue via PR669.

4.3 NF-049 — Log Output for systemd Service Contains Escape Sequences

When using the recommended `ntpd-rs.service.example` systemd service configuraton file, the ntpd-rs log output redirected to `/var/log/syslog` contains ANSI escape codes for text formatting.

Example:

```
localhost ntp-daemon[22685]: #033[2m2023-03-28T15:02:43.054778Z#033[0m #033[33m WARN#033[0m
 #033[2mntp_proto::peer#033[0m#033[2m:#033[0m Peer rejected due to invalid stratum
 #033[3mstratum#033[0m#033[2m=#033[0m16
```

This leads to a less readable log output and should be avoided. Potentially, the daemon can format the log output dependent on whether the output is to a real terminal or redirected via systemd.

4.4 NF-047 — Consider Implementing Mechanism for NTS TLS Key Reloading

At the moment, `ntpd-rs` does not support reloading the TLS private key and certificate at daemon runtime. This leads to difficulties with uninterrupted operation on deployments that use a TLS key of limited time validity. On a server which regularly fetches new keys via the Let's Encrypt Certificate Authority (CA), key reload events are expected to happen about every two months on average.

At the moment, key rotation requires a full server restart, which may lead to some temporary disruption that could be avoided with a dedicated reload mechanism.

Other server processes like as the `nginx` webserver implement externally triggered reload actions via POSIX signals such as SIGHUP:

> Daemon programs sometimes use SIGHUP as a signal to restart themselves, the most common reason for this being to re-read a configuration file that has been changed.

We recommend documenting the expected procedure and available mechanisms to make `ntpd-rs` aware of private key or certificate chain changes. Ideally, this should cover how to configure common tooling such as certbot and their built-in functionality (such as hook mechanisms) to work with ntpd-rs installations.

4.5 NF-046 — Continuous Integration Fuzzing Improvement Recommendations

- Configure the GitHub CI workflow to start with an existing selection of fuzzer test inputs (corpus).
- Consider adding a mechanism to store and fetch discovered corpus files automatically between CI runs.
- Run the fuzzer multi-threaded to make use of the second CPU core available on the runner environment.
- Increase the fuzzing time per target above the current `10s`, or add a second CI configuration with regularly scheduled runs with increased time limits.
- Investigate approaches for automated collection and visualization of line- or region coverage for the fuzz tested codebase.

We proposed additional fuzz harnesses for the ntpd-rs project via PR678.

4.6 NF-045 — Correct the Minimal Supported Rust Version (MSRV) Documentation

The `ntpd-rs` documentation lists Rust 1.60.0 as the Minimal Supported Rust Version (MSRV):

> ntpd-rs is written in rust, and requires cargo 1.60.0 at a minimum to be built.

However, due to requirements by other dependencies, the actual MSRV at the time of reviewing is believed to be `1.65.0` due to requirements of Rust crate dependencies, specifically `Sentry 0.29.3`.

We recommend updating this information in the documentation.

4.7 NF-044 — Insufficient Range Checks for Some Configuration Parameters

The `ntpd-rs` configuration processing accepts several problematic values without warnings or errors for some variables.

`panic-threshold` value allowed to be negative:

Configuration client action via config socket:

```
ntp-ctl config --panic-threshold=-1
```

Server log:

```
INFO ntp_daemon::config::dynamic: dynamic config update operation=ConfigUpdate { log_filter: None,
 panic_threshold: Some(-1.0) }
```

This is also accepted via the `ntp.toml` configuration file:

```
[...]

[system]
panic-threshold = -1

[...]
```

Special values

Special values such as `nan` and `inf` are also accepted at configuration start for some parameters:

```
[system]
min-cluster-survivors = nan
panic-threshold = nan
startup-panic-threshold = { forward = nan, backward = nan }
```

`inf`, `-inf`, `+inf`, `+nan`, `-nan` in different case spelling are also accepted. This may lead to unexpected behavior at runtime.

Other variables

- `initial_poll` can be set to `initial_poll = -1` due to its `i8` value, which is unintended.

Other identified variables with benefits from logical range restrictions

- `rate-limiting-cutoff-ms = 0` is a special case that makes the rate limiting ineffective.

4.8 NF-042 — Consider Changing the Standard Configuration Filepath

We recommend a change from `/etc/ntp.toml` to a full ntpd-rs specific configuration subdirectory in `/etc/`:

- Example directory `/etc/ntpd-rs/`.
- Relevant for placement of additional files, such as NTS certificate file and NTS private key file.
- More flexible directory ownership when running under non-root user.
    - For example, related to limited read/write permissions on the NTS private key and certificate chain files.
- We recommend consulting a Linux distribution maintainer with insight into best practices and packaging requirements to help evaluate additional changes.

This was implemented by the ntpd-rs developers during the review via PR651.

4.9 NF-033 — Document Security Implications of Configuration Socket Access

Although the available functionality of the `ntpd-rs` configuration socket is still fairly limited at the time of review, we observed that it does provide sufficient control to permanently stop the daemon.

An attacker with write access to the configuration socket could set a `panic-threshold` value that is strict enough to lead to daemon shutdowns during normal operation, for example with `panic-threshold=0`. In other words, the clock safety mechanism can be used to perform a persistent Denial of Service, which is inherent in the `panic-threshold` design.

We recommend emphasizing the security implications of giving lower-privileged users control over the configuration socket in the documentation.

4.10 NF-031 — Extend Documentation for Running ntpd-rs as Non-Root

We recommend extending the documentation on how to operate and test the ntpd-rs programs without root permissions.

Particularly relevant:

- `CAP_SYS_TIME` permission.
- `CAP_NET_BIND_SERVICE` permission, if required by the daemon configuration to listen on standard server ports.
- Filesystem access to socket file targets and directories, if required by the daemon configuration.

We also recommend clarifying code documentation on root permissions:

```
// this binary needs to run as root to be able to adjust the system clock.
// by default, the socket inherits root permissions, but the client should not need
```

```
// elevated permissions to read from the socket. So we explicitly set the permissions
```

4.11 NF-030 — Encrypted TLS Private Key File not Supported

Due to limitations of the Rust TLS libraries in use (pemfile), the TLS private key for NTS usage cannot be stored and read in encrypted form from the filesystem by ntpd-rs.

We recommend briefly documenting this limitation in the NTS server configuration section.

4.12 NF-027 — Update ntpd-rs Threat Model Documentation

We recommend updating the existing THREATMODEL.md documentation to improve coverage of the following topics:

- Security considerations related to servers with active use of NTS, such as listing additional assets to map sensitive key material and the implications of cryptography-related failure cases.
- Temporarily sensitive information, such as security cookies used for communication with a specific peer.
- Information given out to peers that could be seen as internal but is required for the correct NTP operation.
- Security considerations and design limitations related to rate liming.
- If possible in the methodology, outline considerations for impacts beyond the scope of the `ntpd-rs` component, such as confidentiality / integrity / availability impacts on the underlying system environment.

Additionally, we recommend providing additional information on the used TRIKE threat model methodology and standard version in use. This was not fully available at the time of the review, which limited our evaluation of the threat model.

This general effort is tracked upstream as issue 618.

4.13 NF-026 — Document Sentry Telemetry Functionality

- The ntpd-rs project includes `sentry` and `sentry-tracing` dependencies.
- The sentry.io service is "an application monitoring solution designed to identify, monitor, and alert developers to errors, bugs, and other performance issues that are occurring in their applications".
- Monitoring solutions may represent an anti-feature to users.
- Crucially, this functionality is not enabled by default and no Sentry API key is included.
- The ntpd-rs development team uses the feature together with a dedicated Sentry log server for internal evaluation purposes.

Recommendation:

- Include Sentry-related information in public documentation.
- Outline the optional/opt-in/unconfigured nature of this functionality to clarify implications on production builds for end users.

4.14 NF-025 — Use of Ubuntu 20.04 in Continuous Integration

The older Ubuntu 20.04 Long Term Stable (LTS) release is in use in the Continuous Integration (CI) system:

- For production builds in .github/workflows/release.yaml.
- For some testing builds in .github/workflows/build.yaml.

Abstract security concerns:

- Software used in Ubuntu 20.04 LTS is several years old, except for selected patches.
- 20.04 LTS is two years behind the latest 22.04 LTS.
- 20.04 LTS will still be still officially supported by Canonical for several years, but may see less active security backporting or patch rollout.

Based on discussions with the `ntpd-rs` developers, 20.04 LTS was chosen intentionally over 22.04 LTS to build programs with an older libc version. This ensures a wider compatibility of the generated Linux binaries with other Linux systems.

4.15 NF-023 — MD5 Usage in Codebase

`ntpd-rs` uses the MD5 hash function via the `md-5` Rust crate. The MD5 hash function has known cryptographic weaknesses.

The description in Cargo.toml clarifies the usage reasons:

```
# Note: md5 is needed to calculate ReferenceIDs for IPv6 addresses per RFC5905
md-5 = "0.10.5"
```

Evaluation confirms that MD5 is only used in `identifiers.rs`, where it is required by the RFC5905 standard and cannot be replaced with a more modern hashing function.

4.16 NF-016 — Analyzed Random Number Generation Handling

We briefly evaluated which random number generator (RNG) or pseudorandom number generator (PRNG) mechanisms are used to provide entropy, since incorrectly used random number generators are a common source of security problems.

Observations:

- `rand` Rust crate in use to provide random values.
- `rand::thread_rng()` mechanism used for a thread-local generator.
- Documentation from ThreadRng, OSRng:

> ThreadRng uses the same PRNG as StdRng for security and performance and is automatically seeded from OsRng.

> Unlike StdRng, ThreadRng uses the ReseedingRng wrapper to reseed the PRNG from fresh entropy every 64 kiB of random data as well as after a fork on Unix (though not quite immediately; see documentation of ReseedingRng). Note that the reseeding is done as an extra precaution against side-channel attacks and mis-use (e.g. if somehow weak entropy were supplied initially). The PRNG algorithms used are assumed to be secure.

We have not discovered any security concerns.

During evaluation, we noticed a potential minor performance improvement and shared it with the `ntpd-rs` developers. ntpd-rs uses gen() to fill `[u8; 16]` and `[u8; 32]` arrays with pseudorandom data. The upstream documentation recommends the use of fill() over `gen()` for this purpose, which may result in minor optimizations.

4.17 NF-015 — Analyzed TLS Server Protocol Handling

We analyzed the NTS-KE server TLS on port 4460.

The observed TLS behavior is very good overall. Only modern `TLS 1.2` and `TLS 1.3` protocols and ciphers are offered. This can be attributed primarily to `rustls` design and configuration defaults, see https://github.com/rustls/rustls#current-features and https://github.com/rustls/rustls#non-features for high-level documentation and keyexchange.rs for code details.

At the moment, `ntpd-rs` does not allow operators to change TLS-related configuration, see https://github.com/pendulum-project/ntpd-rs/issues/560 . If functionality for this is implemented in the future, we recommend offering a mechanism to optionally run `TLS 1.3`-only servers.

Excerpt from testssl.sh tests:

```
Testing protocols via sockets except NPN+ALPN

 SSLv2      not offered (OK)
```

```
 SSLv3      not offered (OK)
 TLS 1      not offered
 TLS 1.1    not offered
 TLS 1.2    offered (OK)
 TLS 1.3    offered (OK): final
 NPN/SPDY   not offered
 ALPN/HTTP2 not offered

 Testing cipher categories

 NULL ciphers (no encryption)                      not offered (OK)
 Anonymous NULL Ciphers (no authentication)        not offered (OK)
 Export ciphers (w/o ADH+NULL)                     not offered (OK)
 LOW: 64 Bit + DES, RC[2,4], MD5 (w/o export)      not offered (OK)
 Triple DES Ciphers / IDEA                         not offered
 Obsoleted CBC ciphers (AES, ARIA etc.)            not offered
 Strong encryption (AEAD ciphers) with no FS       not offered
 Forward Secrecy strong encryption (AEAD ciphers)  offered (OK)

[...]

 Testing vulnerabilities

 Heartbleed (CVE-2014-0160)                not vulnerable (OK), no heartbeat extension
 CCS (CVE-2014-0224)                       not vulnerable (OK)
 Ticketbleed (CVE-2016-9244), experiment.  (applicable only for HTTPS)
 ROBOT                                     Server does not support any cipher suites that use RSA
 key transport
 Secure Renegotiation (RFC 5746)           supported (OK)
 Secure Client-Initiated Renegotiation     not vulnerable (OK)
 CRIME, TLS (CVE-2012-4929)                not vulnerable (OK) (not using HTTP anyway)
 POODLE, SSL (CVE-2014-3566)               not vulnerable (OK), no SSLv3 support
 TLS_FALLBACK_SCSV (RFC 7507)              No fallback possible (OK), no protocol below TLS 1.2
 offered
 SWEET32 (CVE-2016-2183, CVE-2016-6329)    not vulnerable (OK)
 FREAK (CVE-2015-0204)                     not vulnerable (OK)
 DROWN (CVE-2016-0800, CVE-2016-0703)      not vulnerable on this host and port (OK)
                                           [...]
 LOGJAM (CVE-2015-4000), experimental      not vulnerable (OK): no DH EXPORT ciphers, no DH key
 detected with <= TLS 1.2
 BEAST (CVE-2011-3389)                     not vulnerable (OK), no SSL3 or TLS1
 LUCKY13 (CVE-2013-0169), experimental     not vulnerable (OK)
 Winshock (CVE-2014-6321), experimental    not vulnerable (OK)
 RC4 (CVE-2013-2566, CVE-2015-2808)        no RC4 ciphers detected (OK)
```

4.18 NF-012 — Document ReferenceId as Potentially Sensitive Information

The NTP standards include a 32-bit `ReferenceId` value which encodes information about the primary time reference peer or mechanism of a timeserver. This value is communicated to peers to allow the identification and avoidance of unwanted dependency loops between timeservers.

By convention, this ID often includes direct (RFC1305) or hashed (RFC5905) information about the IPv4 and IPv6 addresses in use by the chosen upstream peer of a time server.

Since `ntpd-rs` both transmits its own `ReferenceId` value to peers when operating in a server role as well as showing ReferenceId values in DEBUG level log entries, we recommend documenting that this ID is not randomly generated

and can contain information that is potentially considered sensitive by the operator. This could help avoid accidental publishing of IP addresses via logs in bugtracker tickets or similar cases.

Example conversion: `ReferenceId(2817254435)` -> decimal `2817254435` -> hexadecimal `0xA7EBE423` -> `0xA7 0xEB 0xE4 0x23` -> IPv4 `167.235.228.35`

4.19 NF-003 — Clarify Security Reporting Policy

At the time of review, there is no public dedicated documentation on where and how to report suspected security issues in `ntpd-rs`. We recommend documenting how security issues should be reported to the `ntpd-rs` team. This should be mentioned in a common, highly visible place in the repository such as the README.md file or in a dedicated SECURITY.md file.

If confidential reporting is requested from submitters, a common practice is to have a dedicated `security@` email address on the main project domain that is read by a team of trusted core developers of the project. Optionally, a public PGP key for encrypted communication could be provided.

This is tracked upstream as issue 616.

5 Future Work

- **Extend Fuzz Testing Efforts**
  We recommend extending the existing fuzz testing coverage and automation to increase the detection capabilities of bugs and regressions.
- **Establish more Beta-Testing**
  During this review, we encountered a number of bugs, documentation issues and enhancement opportunities that could be identified and reported by beta-testers of `ntpd-rs` during experimental usage, including some with relevance to security. We recommend investigating possibilities for collaborating with technically capable beta-testers from different communities that have an interest in time synchronization, for example volunteer NTP timeserver operators of the pool.ntp.org time server pool.
- **Perform more Testing Under Adverse Network Conditions**
  Several of the issues discovered in this review could be triggered by network problems. To ensure the detection of future network-related bugs or vulnerabilities and identification of non-optimal `ntpd-rs` behavior under unfavorable network conditions, it could be helpful to establish regular manual or automated testing across deliberately unreliable networks, for example via specially configured firewall software.
  Some examples of problematic network behavior:
  - Unavailable network at startup.
  - Temporary connection losses.
  - Temporarily unavailable individual hosts (NTP peers with outages).
  - Randomly rejected or dropped packets (misconfigured firewalls, overloaded networks).
  - Duplicated, corrupted or reordered packets.
  - Domain Name Service (DNS) resolution failures.
  - Selective failure of IPv6 or IPv4 in hybrid setups.

- **Improve ntpd-rs Deployment Documentation**
  As ntpd-rs moves towards wider adoption, the availability of documentation to guide operators on how to set up and configure the software for their particular use cases becomes more essential.
  We recommend extending the available documentation with well-tuned practical examples for common operating systems and different `ntp-daemon` roles, including information on how to run the daemon with the minimum required privileges, configure strict limits, handle TLS certificates and make other security-related configuration decisions.
- **Retest of Findings**
  When mitigations for the vulnerabilities described in this report have been deployed, a repeat test should be performed to ensure that they are effective and have not introduced other security problems.
  Notably, a significant number of findings were resolved and retested during the engagement.

- **Regular Security Assessments**

  Security is an ongoing process and not a product, so we advise undertaking regular security assessments and penetration tests, ideally prior to every major release or every quarter.

6 Conclusion

We discovered 1 Elevated, 4 Moderate, 12 Low and 3 N/A-severity issues during this code audit.

Our overall impression of `ntpd-rs` is positive. Crucially, the review has not identified any security issues that violate memory safety guarantees, which is a declared design goal and core motivation for the `ntpd-rs` project.

Several of the discovered code issues represent denial of service risks, either via problematic exception handling that leads to program crashes, or via insufficient logical handling or lack of fallback mechanisms. The most severe weakness was in the Network Time Security (NTS) client logic, where a Machine-in-the-Middle (MitM) attacker was able to permanently crash the ntp-daemon by interfering with the TCP connection. In this area, `ntpd-rs` still needs some additional attention to ensure all remote and local network interaction is robust and that misbehaving NTP peers are handled appropriately under all circumstances.
One of the problem spots that emerged during the review relates to the rate limiting implementation of ntpd-rs. Beyond two specific issues that made the existing defenses easier to circumvent, there is the more general challenge on how to implement effective rate limiting defenses for ntpd-rs based timeservers on IPv6 networks without excessive overhead.
Additionally, we identified several security concerns relating to local configuration file handling, socket file handling and log handling that could help local attackers but had a fairly limited impact. This security area will likely benefit from upcoming software packaging efforts and the associated focus on user separation and file ownership within the local system.

To summarize, the `ntpd-rs` project's overall security posture is good, especially considering its current alpha status. However, we still recommend working towards a wider base of experimental testing, testing of attack conditions and improvement of security-related documentation to ensure secure and correct operation of future stable releases.

We would like to thank the `ntpd-rs` team for their support and feedback during the review, which contributed significantly to our ability to identify practical security concerns and make appropriate improvement recommendations. We would also like to compliment the `ntpd-rs` team on their swift remediation actions. During the review, they patched and improved many indicated software issues within days of discovery, which is excellent.

We recommend fixing all of the remaining issues and then performing additional retesting in order to ensure that mitigations are effective and that no new vulnerabilities have been introduced.

Finally, we want to emphasize that security is a process – this code audit is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your project's information security. We hope that this audit report and the detailed explanations of our findings will contribute meaningfully towards that end.

Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

Appendix 1 Testing team

Christian Reitter	Christian is an IT Security Consultant with experience in the area of software security and security relevant embedded devices. After his M.Sc. in Computer Science, he has worked as a developer and freelance security consultant with a focus on fuzzing research. Notable published research includes several firmware vulnerabilities in popular cryptocurrency hardware wallets, including remote code execution, remote theft of secret keys and circumvention of 2FA protection. He has also discovered multiple memory safety issues in well-known smartcard driver stacks.
Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.