NETTITUDE

excellence as standard

Security Assessment Management Report

Prepared for: **The Linux Foundation**

System: **Hyperledger Sawtooth**

Type: Security Assessment

Author: Graham Shaw

Date: 13th December 2017

Version: 1.0

Report Contents

High Level Assessment

The Linux Foundation engaged with Nettitude in October 2017 in order to assess the overall security posture of their Hyperledger Sawtooth software product.

Based on The Linux Foundation's risk profile, primary security concerns and the vulnerabilities identified at the point of the engagement, Nettitude have found Sawtooth to require attention.

Nettitude were able to:

- Access the validator private key from an unprivileged account
- Show that the CHALLENGE authentication method was vulnerable to replay attack
- Cause connections to the REST API to hang

Overall Security Posture

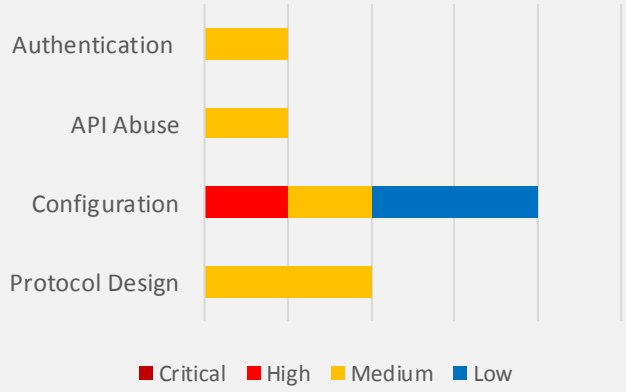
STRONG

MODERATE ATTENTION

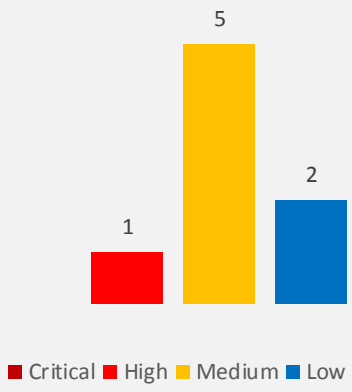
ATTENTION REQUIRED

IMIDIATE ATTENTION

Vulnerability category breakdown



Severity clasification



Limitations

Some limitations and constraints were encountered during the engagement. Please refer to the technical report for more details.

System Analysis

A default installation of Sawtooth allows the private key for the validator to be read by any other user of the system in question. On the face of it this is a potentially serious vulnerability, and it is certainly not good practice, however it is mitigated by the fact that most machines hosting Sawtooth validators are unlikely to have any users who are not also administrators.

Sawtooth supports two different authentication mechanisms, TRUST, and what is supposed to be a more secure method called CHALLENGE. For the latter, the server sends a challenge to the client, which the client is supposed to respond to in a manner that demonstrates it has possession of the authentication credentials.

Unfortunately, the server does not appear to keep a copy of the challenge that was issued, and instead trusts the copy that is returned by the client. This means that the client can substitute any challenge of its choosing, including one recorded from a previous session, therefore negating any advantage from having a two-way challenge-response protocol. However, a replay attack would be difficult to mount in practice if ZMQ encryption has been enabled (as it should be on production systems).

There are several classes of request which will cause connections to the REST API to hang until they time out. This is due to insufficient validation of the supplied parameters before they are forwarded to the validator. This does not prevent the REST API from accepting and processing other connections when done on a small scale, however each connection will consume resources on the server while it is open, therefore this behaviour would make it easier to mount a denial of service attack.

There does not appear to be any upper limit on the size of a block, opening the possibility of a malicious node performing a denial of service attack by adding arbitrarily large amounts of data to the blockchain.

For comparison, the Bitcoin blockchain grows at a very predictable rate due to fine-tuning of the difficulty of adding new blocks, and a hard, upper limit on the size of each block. This is an important consideration because, once they become a confirmed part of the chain, blocks must be stored indefinitely and are very difficult to remove. Adding a sufficiently large amount of data would deny service until a sufficient number of nodes were upgraded, or the blockchain was forked.

Session management for the ZMQ interface to the validator depends upon ZMQ providing encryption at the transport layer, however it defaults to unencrypted operation if the required key pair is absent from the configuration. It would be better for the validator to refuse to run unless the key pair was provided, or the user has explicitly requested unencrypted operation (which would only be appropriate for testing).

Sawtooth was found to be vulnerable to a method of attack known as log injection, which is made possible when non-validated inputs are written verbatim to the log. This would not have any direct business impact, however as part of a larger attack it might be used to fabricate log entries to mislead incident response efforts, or to corrupt the log to prevent it from being processed by automated monitoring systems.

The Ubuntu-packaged network services are all run from the same account, 'sawtooth'. Whilst this is considerably better than running them as root, it would be better still if a separate account were used for each subsystem. This will reduce the risk of a vulnerability in a lower-value subsystem being used to pivot to a higher-value subsystem.

Notwithstanding these findings, the software was found to be generally well written and documented. Particular points of note include:

- Good modularity and encapsulation of subsystems.
- Use of reputable third-party libraries to provide cryptographic primitives, which is (usually) less risky than re-implementing them.
- Good replay protection in PoET.
- Use of the Ironhouse pattern.
- Use of chained request handlers in the validator (reminiscent of the architecture of Apache), which makes it very clear what validation and authentication procedures are applied at each endpoint.

Next Steps

Nettitude recommends that The Linux Foundation perform the following post engagement activities in the order of priority indicated.

	Activity	Description	Priority
1	Debrief from Nettitude	Nettitude will deliver a formal debrief to The Linux Foundation in order to ensure that the findings of this engagement have been fully comprehended and to help assist in the formulation of a remediation plan.	++++
2	Validator private key	Restrict access to sensitive files, as per recommendation in technical report.	+++
3	CHALLENGE authentication	Do not trust the copy of the challenge returned by the client. Instead, keep a copy of the challenge as issued by the server, and use that for validation of the response.	++
4	Validation in REST API	Validation of arguments passed to the REST API should be improved, particularly in the case of data that will be forwarded to the validator.	+

Distribution List

Nettitude	Name	Title
	Graham Shaw	Security Consultant
	Richard Dennis	Security Consultant
	Richard Hicks	Security Consultant
	Kristopher Vasilik	Key Account Manager

The Linux Foundation	Name	Title
	David Huseby	Security Maven, Hyperledger

Revision History

Version	Issue Date	Issue By	Comments
0.1	30th November 2017	Graham Shaw	Initial Draft
0.2	6th December 2017	Richard Hicks	Quality Assurance
0.3	6th December 2017	Kristopher Vasilik	Quality Assurance
1.0	13th December 2017	Graham Shaw	Final