Cryptonics

Audit Report

pNetworks Enclaves Integration

DRAFT

May 18, 2021

Table of Contents

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

This audit has been performed by

Philip Stanislaus

Cryptonics Consulting S.L.
Ramiro de Maeztu 7
46022 Valencia
SPAIN

https://cryptonics.consulting/
info@cryptonics.consulting

Introduction

Purpose of this Report

Cryptonics Consulting has been engaged to perform a security audit of the pNetworks encalves Integration for the pTokens 2-way asset transfer bridge (https://p.network/). The objectives of the audit are as follows:

1. Determine the correct functioning of the integration, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the changes between the following commits in the code provided by the developers in the following GitHub repositories:

https://github.com/provable-things/ptokens-strongbox-common

commit: `b3b1985809138edb6221d064c84383af328e7a73`

https://github.com/provable-things/ptokens-strongbox-pbtc-on-eth

commit: `bcf508664656fb36d341d676355d698d20efd9a9`

https://github.com/provable-things/ptokens-nitro-pbtc-on-eth-priv

commit: `d0a823ea3eed1af685029c440dbc530b686f528a`

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under- / overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

The submitted code implements integrations for executing the pNetworks relayer nodes within the context of trusted execution environments/secure enclaves. Google Android Strongbox and AWS Nitro are supported.

How to read this Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged** or **Resolved**. Informational notes do not have a status, since we consider them optional recommendations.

Note, that audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentatio**n, and **test coverage**. We include a table with these criteria for each module, in the corresponding findings section.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Summary of Findings

No	Description	Severity	Status
ptokens-strongbox-common			
1	Enclave allows signing of a subset of arbitrary bytes without prefix	Critical	Reported
2	Data written after the first transaction is lost	Critical	Reported
3	Validity of key attestation certificate chain is not verified	Major	Reported
4	JWS' signature of attestation response is not verified	Major	Reported
5	Written signed state hash may use stale data	Major	Reported
6	Enclave does not enforce absence of debug mode	Major	Reported
7	Unsafe mutation of global variables	Minor	Reported
8	Database wiring returns a value to signify an error	Minor	Reported
9	Sensitive information might be exposed through command argument logging	Minor	Reported
10	Deprecated dependency on trusted bytecode	Informational	-
ptokens-nitro-pbtc-on-eth-priv			
11	DEBUG command allows execution of any command	Critical	Reported
12	Sensitive data in database is not encrypted	Critical	Reported
13	Set KMS policy during key creation	Major	Reported
14	Parallel DB transactions lead to race conditions	Major	Reported
15	Running enclave in debug mode undermines PCR validation	Major	Reported
16	Enclave does not enforce absence of debug mode	Major	Reported
17	Dockerfile built from community image	Minor	Reported
18	Nitro Secure Module library built from fork	Minor	Reported

19	IAM credentials are stored in parent instance's filesystem	Minor	Reported
20	Manual building of dependencies	Informational	-

Code Quality Criteria

ptokens-strongbox-common

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium	-
Level of Documentation	Medium	-
Test Coverage	None	There are currently no tests in the repository.

ptokens-strongbox-pbtc-on-eth

Criteria	Status	Comment
Code complexity	Low-Medium	-
Code readability and clarity	Medium-High	-
Level of Documentation	Medium	-
Test Coverage	None	There are currently no tests in the repository.

ptokens-nitro-pbtc-on-eth-priv

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium	-
Level of Documentation	Medium	-
Test Coverage	None	There are currently no tests in the repository.

Detailed Findings

ptokens-strongbox-common

1. Enclave allows signing of a subset of arbitrary bytes without prefix

Severity: Critical

In `jni/src/bridges/pbtc_on_eth.rs:340` and `jni/src/bridges/peos_on_eth.rs:667`, the function `sign_ascii_msg_with_eth_key_with_no_prefix` allows arbitrary ASCII data to be signed. Since ASCII is a subset of arbitrary bytes, there may be a valid transaction that could be composed with that subset. As an example, the following bytes would be valid ASCII and not a hex string: `[32, 70, 127, 120, 60]`.

Recommendation

We recommend removing the ability to sign ASCII without a prefix.

Status: Reported

2. Data written after the first transaction is lost

Severity: Critical

The `endTransaction` function in `src/main/java/io/ptokens/database/DatabaseWiring.java:155` returns before writing data to the database if the instance variable `END_TX_IN_PROGRESS` equals `true`. When the `endTransaction` function runs the first time, that variable is set from `false` to `true`. However, after the data has been written to the database, `END_TX_IN_PROGRESS` is never reset to `false`. Consequently, any transaction that runs after the first will lead to data loss, which could lead to lost value (for instance if a private key was created or rotated).

Recommendation

We recommend setting `END_TX_IN_PROGRESS` to `false` in line `195`, after data was written to the database.

Status: Reported

3. Validity of key attestation certificate chain is not verified

Severity: Major

In `src/main/java/io/ptokens/security/Strongbox.java:329`, the certificate chain associated with the hardware-backed key store is retrieved, but the validity of the certificates is not checked.

Recommendation

We recommend checking the validity of each certificate with `X509Certificate#checkValidity()`. Additionally, we recommend verifying certificate revocation status with a separate server, see https://developer.android.com/training/articles/security-key-attestation for detail.

Status: Reported

4. JWS' signature of attestation response is not verified

Severity: Major

In `src/main/java/io/ptokens/safetynet/SafetyNetHelper.java:183`, the `parseJsonWebSignature` method does not verify the signature in `jwtParts[2]`.

Recommendation

We recommend verifying the signature of the JWT.

Status: Reported

5. Written signed state hash may use stale data

Severity: Major

In `src/main/java/io/ptokens/database/DatabaseWiring.java:188`, the `writeSignedStateHash` method is called after the database transaction is ended in line `183`. If any data is written between that end of the transaction and the data retrieved within `writeSignedStateHash`, the signed state hash references stale data and cannot be trusted.

Recommendation

We recommend calling `writeSignedStateHash` before ending the transaction.

Status: Reported

6. Enclave does not enforce absence of debug mode

Severity: Major

The enclave currently supports a ptokens core that has the `debug` feature enabled. That allows enclave users to retrieve the private key from the enclave, for instance.

Recommendation

We recommend implementing a safeguard in the enclave that fails any command if the `debug` feature is enabled.

Status: Reported

7. Unsafe mutation of global variables

Severity: Minor

In `jni/src/bridges/pbtc_on_eos.rs:55` and `73`, `jni/src/bridges/pbtc_on_eth.rs:58` and `76`, `jni/src/bridges/peos_on_eth.rs:51` and `69` and `jni/src/bridges/perc20_on_eth.rs:52` and `70`, the global variables `CALLBACK_GLOBAL_REF` and `MAYBE_JNI_JAVAVM` are mutated, which is considered unsafe in Rust since it could introduce race conditions. Those variables are used to allow the native (Rust) code to call back into the managed (Java) code, e. g. for database access. When the native code is loaded, `MAYBE_JNI_JAVAVM` is set, and at the beginning of every method call that's coming through the JNI, `CALLBACK_GLOBAL_REF` is set. `MAYBE_JNI_JAVAVM` can then be used to retrieve the env to call back into the JVM and `CALLBACK_GLOBAL_REF` to determine which class to call.

Recommendation

We recommend changing the logic to instantiate a singleton on the database struct instead of using a global mutable variable. If a global variable is still necessary, a Mutex may be used to prevent race conditions and remove the usage of unsafe Rust.

Status: Reported

8. Database wiring returns a value to signify an error

Severity: Minor

In `src/main/java/io/ptokens/database/DatabaseWiring.java:96`, the value `0x0F` is returned to signify the error that a value was previously removed. This implies that an actual value of `0x0F` would always be interpreted as an error.

Recommendation

We recommend using errors or an extra error byte instead.

Status: Reported

9. Sensitive information might be exposed through command argument logging

Severity: Minor

The `src/main/java/io/ptokens/commands/Command.java:224`, the value argument of the command executed in the enclave is logged. That can lead to exposure of sensitive information, for instance when the core is run in debug mode and the private key is set with the `debugSetKeyInDbToValue` command.

Recommendation

While in normal operation commands arguments are not expected to contain sensitive information, we recommend not logging command arguments.

Status: Reported

10. Deprecated dependency on trusted bytecode

Severity: Informational

In `jni/src/bridges/pbtc_on_eth.rs:197` and `jni/src/bridges/perc20_on_eos.rs:177`, the `_bytecode_path` variable contains a path to trusted smart contract bytecode, which is no longer used.

Recommendation

We recommend removing the deprecated variable.

Status: Reported

ptokens-strongbox-pbtc-on-eth

No findings

ptokens-nitro-pbtc-on-eth-priv

11. `DEBUG` command allows execution of any command

Severity: Critical

The `DEBUG` RPC in `docker-enclave/enclave-server/enclave-server.py:21` allows the parent instance to run arbitrary code in the instance, which opens many security vulnerabilities, e. g. the ability to sign arbitrary data or rotate/delete private keys.

Recommendation

We recommend removing the `DEBUG` method.

Status: Reported

12. Sensitive data in database is not encrypted

Severity: Critical

In `docker-enclave/enclave-server/db.py:237`, sensitive values such as the private key used to sign transactions are not currently encrypted. Likewise, decryption has not yet been implemented in `docker-enclave/enclave-server/db.py:194`. This is a critical issue, since it allows the parent instance as well as any service having access to the database to read plaintext secrets.

Recommendation

We recommend implementing encryption of sensitive values.

Status: Reported

13. Set KMS policy during key creation

Severity: Major

One of the main benefits of using an enclave is the possibility to only grant access to key material if the enclave can prove to be trustworthy. In AWS Nitro enclaves, such trustworthiness can be proven through platform configuration registers (PCRs). PCRs are computed and provided in a signed attestation document by the Nitro Hypervisor. Using AWS KMS together with AWS Nitro enclaves allows conditions to be placed on key capabilities based on those PCRs. See https://docs.aws.amazon.com/kms/latest/developerguide/policy-conditions.html#conditions-nitro-enclaves for details. Currently, no policy is set during key creation in `docker-enclave/enclave-server/kms.py:88`.

Recommendation

We recommend setting up a policy with PCR conditions during key creation. Details can be found here: https://docs.aws.amazon.com/kms/latest/APIReference/API_CreateKey.html#API_CreateKey_RequestParameters.

Status: Reported

14. Parallel DB transactions lead to race conditions

Severity: Major

The `DB__start_transaction` method in `docker-enclave/enclave-server/db.py:122` resets the pending state of other open transactions. This behaviour leads to a race condition whenever multiple transactions could be open at the same time. The problem is mitigated by the fact that the enclave client employs a `FileLock` in `enclave-client.py:116`, but can still occur within the same command when the core opens multiple transactions in parallel.

Recommendation

We recommend using mutexes on either the whole database or the data they access for transactions. That will prevent any unintended state overwrites.

Status: Reported

15. Running enclave in debug mode undermines PCR validation

Severity: Major

In `build_and_run.sh:4`, the enclave is run with the `--debug-mode` flag. Attestation documents created by the hypervisor in debug mode will have zeroed platform configuration registers (PCRs), which will lead to the failure of any PCR conditions set in KMS.

Recommendation

We recommend removing the `--debug-mode` flag.

Status: Reported

16. Enclave does not enforce absence of debug mode

Severity: Major

The enclave currently supports a ptokens core that has the `debug` feature enabled. That allows the parent instance to retrieve the private key from the enclave, for instance.

Recommendation

We recommend implementing a safeguard in the enclave that fails any command if the `debug` feature is enabled. This check could be coupled to the enclave being in debug mode.

Status: Reported

17. Dockerfile built from community image

Severity: Minor

In `docker-enclave/Dockerfile`, a community image `frolvlad/alpine-glibc:alpine-3.12_glibc-2.32` is used as the base image. Using a community image here introduces an attack vector through the individual maintaining that image. Additionally, a community image often receives delayed security patches.

Recommendation

We recommend using an official base image. If changes are needed, we recommend that the team maintains their own fork that pulls in changes from upstream repositories.

Status: Reported

18. Nitro Secure Module library built from fork

Severity: Minor

In `README.md:11`, the link to the Nitro Secure Module library `libnsm` leads to the fork https://github.com/donkersgoed/aws-nitro-enclaves-nsm-api of the original https://github.com/aws/aws-nitro-enclaves-nsm-api repository. Using a fork here introduces an attack vector through the individual maintaining that fork. Additionally, a fork often receives delayed security patches.

Recommendation

We recommend building from the original repository. If changes are needed, we recommend that the team maintains their own fork that pulls in changes from upstream repositories.

Status: Reported

19. IAM credentials are stored in parent instance's filesystem

Severity: Minor

In `enclave-client.py:80`, IAM credentials are loaded from the parent instance's filesystem. Storing credentials in the filesystem can open security vulnerabilities (users/groups/others could have access to read them, they are included in backups with potentially other rights, they could be accidentally committed to repositories or copied).

Recommendation

We recommend passing credentials as environment variables instead.

Status: Reported

20. Manual building of dependencies

Severity: Informal

Dependencies such as `pbtc_app` and `libnsm` are currently built in manual steps before the Docker image is built. While the build process is trusted anyways, executing manual steps leave room for human error. Additionally, building dependencies outside of the docker container might be nondeterministic, since the host system may be set up differently during subsequent runs.

Recommendation

We recommend moving the building of dependencies into the Dockerfile. This ensures more deterministic builds. Multi-stage builds are a good way to keep the final image free of build artifacts.

Status: Reported