BlindAI Core Security Assessment : Mithril Security Response

Mithril Security engaged with cybersecurity firm Quarkslab to perform a security assessment of BlindAI Core. The audit lasted 40 days and was conducted **from January to March 2023**

The audit was conducted on blindai-preview v0.0.2

The project has since been integrated into the main BlindAI repository as *BlindAI Core*. All future developments, and fixes in response to the audit will be made to BlindAI Core. Please note that the audit does not cover the BlindAI API or our on-going work to support AWS Nitro enclaves.

During the assessment, the auditors identified :

- 1 vulnerability of low severity
- 7 security weaknesses considered as informative only.

No vulnerability that could compromise the confidentiality or integrity of user data (model, tensors) were found. These results are very positive. This demonstrates the strength of our system and the effectiveness of our security design.

This response was prepared by the Mithril Security team to discuss the issues that were found during the assessment and their resolution steps.

The official Quarkslab report can be found here.

Issues

LOW1

Description

blindai-preview does not have a mechanism to offload to the disk when memory consumption is high, which makes a denial-of-service attack possible when multiple models are sent.

Resolution

There is now a dedicated port for model management. This port is intended to be exposed only to the solution administrators. Now that a standard user cannot upload models, the proposed attack scenario cannot be carried out.

Discussion

There is still no way to offload to disk when memory consumption is high, so other DoS attacks could still be carried out. That being said, availability attacks are explicitly outside of our threat model. This is because preventing DoS attacks is complex and requires the combination of different security measures such as network security, and proactive monitoring for instance.

It is important to understand that enclaves can only provide increased confidentiality and integrity compared to regular solutions, since availability is out of scope of the Intel SGX threat model. Of course, we understand that users might want enclave solutions to be as robust as regular solutions regarding availability. As BlindAI is still in beta, we prioritize confidentiality and integrity to leverage SGX promises. We leave addressing availability considerations for future work.

Also we want to point out that from a confidentiality perspective the fact that no data is persisted after the enclave has been terminated is a nice property to have. We would like to maintain this property in the future. This could be done by encrypting the offloaded data with a key only known to the enclave instance.

For now, we are studying how to better manage memory with regard to model management mostly for usability reasons. We want to prevent an administrator from accidentally crashing the server by uploading a model that is too large to fit in memory.

INFO1

Description

Enclaves are unsigned prior to running which is not compliant with Fortanix EDP documentation.

Resolution

Starting with blindai-preview v0.0.5, we are signing the enclave with a dummy key. This is mainly to circumvent the fact that the Fortanix EDP launcher will run an unsigned enclave in debug mode by default. As stated in the audit report, this was a usability issue, not a security issue because the client will ultimately check if the enclave is running in debug mode.

Discussion

The enclave is now signed with a dummy key. This might seem strange but this is because there are two ways to use SGX. You can either make trust decisions based on the enclave identity, the *MRENCLAVE* policy, or you can make decisions based on which software vendor signs the enclave, the *MRSIGNER* policy.

We don't use the *MRSIGNER* policy, since this would imply that our users would have to blindly trust all software we make (and that would defeat the transparency brought by the attestation). The decision to trust an enclave is made based on the MRENCLAVE. Thus, the enclave signing key does not play any role in security.

INFO2

One of the dependencies used by the enclave, rouille, seems to have some issues.

Comment

Usual HTTP software stacks are quite large. Even HTTP libraries written in Rust such as Hyper relies on a significant amount of unsafe code, which carries a security risk. Additionally, HTTP servers have been written with the (implicit) assumption that the OS is not malicious. This assumption is false in the case of SGX.

Another aspect to consider is the vulnerabilities associated with concurrency bugs. Those kinds of vulnerabilities require precise timing and coordination of multiple threads or processes to be exploited by an attacker. This is very hard to achieve remotely, and therefore HTTP libraries will consider them low-severity. However, because in the SGX

threat model, the untrusted OS controls the scheduling of the enclave threads, those vulnerabilities need to be taken seriously.

The silver lining : network stacks were never designed with the enclave threat model in mind. Operating them inside SGX is therefore hazardous, it might be possible but it requires a careful review of the codebase and generally some changes. This motivated us to search for simpler HTTP libraries. We chose tinyhttp and rouille for the following reasons:

- they contain no unsafe code. So we don't have to worry about a memory vulnerability and thanks to the Fortanix EDP "LibOS" like approach this should also protect us from Iago attacks, which have in the past been a source of numerous vulnerabilities against network stacks. Let's not repeat the same mistakes ! For example of such vulnerabilities, you can refer to TeeRex: Discovery and Exploitation of Memory Corruption Vulnerabilities in SGX Enclaves,
- their codebase is small, which allowed us to analyze the parts that might pose a security risk in the SGX threat model. This work would have been a lot harder to carry on a complex stack like Hyper.

However, it is important to keep in mind that it is tradeoff. tinyhttp and rouille are not as battle-tested as hyper. We expect that they contain bugs that are not present in hyper. They also have less features and less performance. We believe it's okay since those are not our priorities, our priority is to ensure the confidentiality and integrity of user data processed in the enclave, and for that tinyhttp and rouile are better.

INFO5

Description

When the runner requests a quote from the quoting enclave, it uses a nonce value set to 0 which is not a good practice.

Discussion

While reusing a nonce can be a cryptographic weakness, there are use cases where it is perfectly fine. For instance if you generate a new AES key for each message, using AES GCM with a nonce set to 0 is not a problem, at least if you use 256-bits keys.

Going back to our case, we can find in Intel source code :

> The caller can request a REPORT from the QE using a supplied nonce. This will allow the enclave requesting the quote to verify the QE used to generate the quote. This makes it more difficult for something to spoof a QE and allows the app enclave to catch it earlier. But since the authenticity of the QE lies in the knowledge of the Quote signing key, **such spoofing will ultimately be**

detected by the quote verifier. QE REPORT.ReportData = SHA256(*p_{nonce}||*p_{quote})||0x00)

Clearly from that paragraph we see that setting and checking the nonce would add no measurable security benefit in our context, so there is no reason to do so. It would only add complexity.

Resolution

We have added a comment in our code to justify that the nonce is set to 0. See https://github.com/mithril-security/blindai/pull/177

INFO6

Description

Client uses `rapidjson` from Tencent which is known for not taking security issues

Discussion

This is one of the concerns related to the use of Intel official QVL to verify attestation evidence. The QVL is written in C++. It is also using OpenSSL which arguably has somewhat of a poor security track record. As suggested by the auditors, we are evaluating how we could switch to a Rust based implementation. Of course the benefits of such change, needs to be balanced with the advantage of relying on an official Intel implementation. Since people trust Intel SGX, they will be more inclined to trust an official Intel library than our own implementation in Rust.

Remediation plan

This issue is mostly informative, so it is not a priority. Still we are considering rewriting the client attestation verification in Rust to have better control over dependencies and avoid memory safety bugs.

INFO7

Usage of CBOR to encode data sent and received by the enclave which is not maintained anymore

Response

We are aware that the project maintainer no longer maintains the crate. In response, we maintain a fork of `serde_cbor` at https://github.com/mithril-security/cbor, where we do basic maintenance (updating dependencies if needed).

Discussion

This decision was taken by our team after reviewing the project's code and looking at alternatives. In short, the `serde_cbor` code base is mature and we think that `serde_cbor` is a case of boring software (we say that in a good way!). The decision was made considering that :

- the code quality is good
- the code size is relatively small
- it has few (3) dependencies
- we do not expect that we will need to add features since it implements a standardized format

We will continue to keep the project's dependencies up to date, as we have done in the past https://github.com/pyfisch/cbor/compare/master...mithril-security:cbor:master

Finally as the past maintainer said himself :

> If the crate works for you there is no need to switch to another implementation.
> ~~ Pyfisch, August 2021

Miscellaneous

INFO3, INFO4 are related to code quality, and are mostly the result of BlindAI still being in beta. We expect these informative-only issues to resolve as the project becomes more mature.