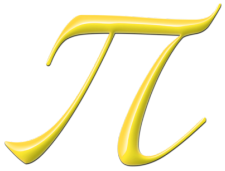

Paragon Initiative Enterprises

Bytejailcore

Source Code Audit
August 2015

I. Introduction

This document describes the security audit of the Bytejailcore library by Paragon Initiative Enterprises for EAM Experience Area Münsingen GmbH.

Our audit targeted git commit `68c96e3b4be645dfb816fd3101c455cc416156af`, which was committed on August 12, 2015.

This report was prepared by Scott Arciszewski, CDO, and reviewed by Robyn Terjesen, CEO.

Audit Results Summary

After a thorough examination of the Bytejailcore library, we found no security vulnerabilities or cryptographic weaknesses. We did find one source for potential bugs down the road, which was quickly remedied by Christian Hermann.

II. Audit Scope

We have limited the scope of this audit to focus specifically on the contents of the Bytejailcore source code.

We excluded the following from our scope (although, where appropriate, we did verify that their features were being used safely, in some cases assisted by ILSpy):

- EldoS SecureBlackbox
- The .NET bindings for Libsodium
- Protobuf-net
- StreamCryptor
- NaclKeys (previously audited by Paragon Initiative Enterprises)

III. Issues

No security vulnerabilities were discovered in Bytejailcore.

IV. Other Findings

Note: The findings in this section are not necessarily vulnerabilities.

1. (Resolved) Initial Concerns Over Null Byte Padding in BytejailUtilities

Before our investigation began, in an external method in StreamCryptor, `Utils.StringToPaddedByteArray()`, was padding the plaintext to exactly 256 bytes by appending null bytes. To strip padding, Bytejailcore was invoking `TrimEnd('\0')` on the plaintext after decryption.

At a glance, this looks like it could be vulnerable to padding oracle attacks. **This wasn't ever exploitable**, because libsodium's authenticated public key encryption (`PublicKeyBox` – which encapsulates Curve25519 + Xsalsa20 + Poly1305) saved the day. This also wasn't a big concern for non-security bugs, as there aren't many (any?) circumstances where you would need a filename to end in a null byte.

For general purposes, null byte padding is a bad idea (especially if you're encrypting already-encrypted data, which is the most likely scenario where your plaintext message will end in one or more null bytes). Blindly stripping null bytes might cut out part of your plaintext message, even if you're using an authenticated encryption construction (Encrypt Then MAC).

Our recommendation was to implement PKCS7 padding in StreamCryptor and remove the padding safely in Bytejailcore. Christian Hermann implemented these changes before our investigation was completed.

2. (Resolved) Code Duplication With Proxy Configuration

At the time of our audit, the code to configure the proxy settings for HTTPS connections was duplicated in four different places throughout the same file. While this wasn't a security risk today, it could lead to problems in the future if one of the places was overlooked for an important change to the configuration code.

Upon receiving our recommendation, Christian Hermann moved this code into a separate class to improve the maintainability of this library and prevent mistakes in the future.

3. StreamUtilities.ReadFully() Is Safe

At a glance, this method looked like a potential target for buffer overflow vulnerabilities since it operates on a fixed-size memory buffer. All it does is read data from the input stream into the buffer (up to the length of the buffer, and never beyond) and then write the buffer into a memory stream until the input stream is empty. There is no avenue for exploitation here.

V. Conclusion

The Bytejailcore audit is our most comprehensive investigation to date, and we are happy to say that we did not find any security-affecting vulnerabilities or cryptographic weaknesses.

We also did not find any code that could expose the user's credentials or encryption keys to an outside system, trusted or otherwise. (In other words, we did not find anything resembling an intentional or unintentional backdoor.)

Even though we did find a few very minor issues with some parts of the code, they were addressed correctly in less time than it took us to find and analyze them. This shows a commitment to secure software development and demonstrates the willingness to listen to security researchers. If more companies would follow the example set by Christian Hermann and the rest of the EAM GmbH team, the Internet would be a lot safer for end users.