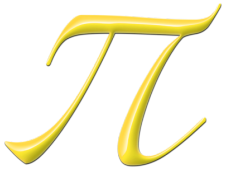

Paragon Initiative Enterprises

Bytejail Client

Source Code Audit
August 2015

I. Introduction

This document describes the security audit of the Bytejail client software by Paragon Initiative Enterprises for EAM Experience Area Münsingen GmbH.

Our audit targeted git commit `9271ed90d184be1749cfd1a461d79ea601c06520`, which was committed on August 20, 2015.

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

Audit Results Summary

After a thorough examination of the Bytejail client software, we found no security vulnerabilities or cryptographic weaknesses. Users of this software will also be pleased to hear that we did not find any backdoor-like functionality in the source code of this software.

II. Audit Scope

We have limited the scope of our audit to the contents of the Bytejail client source code.

We excluded the following dependencies from our scope (although, where appropriate, we did verify that their features were being used safely):

- Base58Check
- Bytejailcore (separately audited by Paragon Initiative Enterprises)
- EldoS SecureBlackbox
- The .NET bindings for Libsodium
- NaclKeys (previously audited by Paragon Initiative Enterprises)
- Protobuf-net
- StreamCryptor
- Zxcvbn

III. Issues

No security vulnerabilities were discovered in the Bytejail client software.

IV. Other Findings

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

1. Simplify log encryption with SealedPublicKeyBox

The current .NET bindings for libsodium expose the `crypto_box_seal()` and `crypto_box_seal_open()` features for .NET developers. Using this could result in simpler code (fewer lines of code usually means fewer places for an error to pop up).

Currently, an ephemeral keypair is generated and used with a hard-coded public key (for which I assume the associated curve25519 secret key is held by the Bytejail developers). The ephemeral secret key, static public key, and a nonce are used with PublicKeyBox to encrypt anonymized error messages.

That being said, the libsodium features are implemented safely.

2. The ExternalCommand feature might be unnecessary

Before we began our audit, we were informed of an External Command feature. This was intended or power users, and the README explicitly stated that these commands were not validated by the client software. Its main purpose was to allow users to facilitate secure file deletion in response to certain events (e.g. file upload completion). However, development continued while we were auditing Bytejailcore, and a secure deletion feature was implemented in the client software. (As of this writing, it is not enabled by default.)

Secure file deletion is a complicated topic. On consumer SSDs, software-implemented secure delete functionality comes with no guarantees. On HDDs, overwriting the file with pseudorandom bytes and then deleting it is usually enough to make forensic recovery difficult. Knowing whether or not overwriting the data will be effective isn't trivial. Hooking into Eraser makes sense for some hardware, but for others it's security theater.

Recommended: Sam Bowne's Investigation into Data Evaporating from SSDs.
https://samsclass.info/121/proj/ssd-evaporation.htm

If the new secure file deletion feature can be improved beyond what Eraser offers, then there might no longer be any need for external command execution.

That said, the commands must be specified and configured by a local user. The configuration files are protected by Microsoft's DPAPI, which we presume to be reasonably secure. More importantly, the way the external commands are implemented separates the command from the arguments so OS command injection from a malicious input isn't possible, barring an unknown vulnerability in the .NET Framework itself. This feature presently poses no security risk so long as users don't go out of their way to misuse it.

V. Conclusion

Our investigation of the Bytejail client software did not uncover any security vulnerabilities, cryptographic weaknesses, or anything resembling a backdoor. User's sensitive credentials are never transmitted, and even error messages are anonymized and encrypted before they are sent to the developers.

The Bytejail client software also does a lot of good things: It actively recommends DNSCrypt (huge privacy win for end users), which is also one of the supported DNS options. It used Zxcvbn to give the user realistic feedback about the strength of their two passwords for each Jail identity. If Paragon Initiative Enterprises were developing Bytejail, these are two decisions we would have made.

Combined with our knowledge from auditing of other components of the Bytejail infrastructure, we have confidence in the privacy and security of this service.