Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report Tresor Application Crypto 07.-09.2017

Cure53, Dr.-Ing. M. Heiderich, MSc. N. Kobeissi

Index

Introduction

This report documents the findings of a security assessment of the Tresor applications created by 1&1 Mail & Media GmbH, specifically focusing on the results pertaining to the cryptographic implementations. Carried out by Cure53, this project yielded two security-relevant discoveries in the realm of cryptographic security.

Notably, this cryptography assessment is a part of a previously agreed several rounds of testing in the wider realm of 1&1 Mail & Media GmbH’s cooperation with Cure53. Therefore, narrowing down the findings to the cryptography realm can help present a clearer verdict on this pivotal arena. Importantly, the focus on cryptography was explicit during the first phase of the assignment performed in July 2017. As far as methodological approach of this project is concerned, a white-box strategy has been selected. The initial round of testing looked at web applications besides being particularly invested in assessing cryptographic libraries. Two members of the Cure53 were specifically tasked with inspecting the three cryptographic libraries created and maintained by the Cryptomator team and used by the Tresor app entities. These were checked for correctness of implementation, as well as studied for presence of other security issues.

In the next sections, the report provides a case-by-case discussion for each of the two issues. Comprehensive mitigation advice is concurrently supplied for each finding whenever possible. The report closes with a conclusion about the general verdict about the cryptographic libraries undergirding the Tresor app operations.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Cryptomator Crypto Libraries**
  - https://github.com/cryptomator/cryptolib
  - https://github.com/cryptomator/cryptofs
  - https://github.com/cryptomator/siv-mode
  - https://github.com/cryptomator/cryptomator-objc-cryptor

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in a chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *1u1-22-001*) for the purpose of facilitating any future follow-up correspondence.

1u1-22-001 Crypto: Release Signing Private Key Available in Public (*Critical*)

Note: *The GPG key is used exclusively for the Maven repositories, is designed for signing only and is protected by a 30-character generated password (alphabet size: 96 chars). It is iterated and salted (SHA1 with 20971520 iterations). An offline attack is also very unattractive. Apart from that, this finding has no influence on the Tresor apps. This was not known to Cure53 at the time of reporting.*

The private release-signing PGP key, namely *34C80F11.gpg,* is publicly disclosed inside the project's *siv-mode*, *cryptolib* and *cryptofs* GitHub repositories. While the key is still protected with a seemingly strong passphrase, there are no circumstances which could warrant a full *release* private key to be leaked in public. An adversary willing to invest time could potentially bypass the passphrase protections imposed on the private key and compromise the *release* pipeline.

This issue can be verified by navigating to the Github URLs listed next.

URLs:
https://github.com/cryptomator/cryptolib/blob/develop/34C80F11.gpg
https://github.com/cryptomator/cryptofs/blob/develop/34C80F11.gpg
https://github.com/cryptomator/siv-mode/blob/master/34C80F11.gpg

Output:
```
-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: GnuPG v1
```

```
lQc+BFdtLX[...]
```

It is recommended to have the private key replaced, rotated, and removed from public repositories.

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

1u1-22-002 Crypto: JceAesBlockCipher leaks info via AES/ECB default *(Info)*

It was found that the *JceAesBlockCipher* implementation defaults to using AES in ECB mode, which is known to be semantically insecure[1]. This may result in unintended data leakage when the library is used without the default ECB setting being changed. The code snippet below can be reviewed as a confirmation of this problem.

File:
siv-mode-master/src/main/java/org/cryptomator/siv/JceAesBlockCipher.java

Affected Code:
```
class JceAesBlockCipher implements BlockCipher {

        private static final String ALG_NAME = "AES";
        private static final String KEY_DESIGNATION = "AES";
        private static final String JCE_CIPHER_NAME = "AES/ECB/NoPadding";
[...]
        public JceAesBlockCipher() {
                try {
                        this.cipher = Cipher.getInstance(JCE_CIPHER_NAME); // defaults to SunJCE but allows to configure different providers
                } catch (NoSuchAlgorithmException | NoSuchPaddingException e) {
                        throw new IllegalStateException("Every implementation of the Java platform is required to support AES/ECB/NoPadding.");
                }
        }
```

It appears that the only reason for the AES/ECB mode being currently included in the aforementioned adapter is to satisfy the Java compliance requirements for cryptographic modules. Furthermore, it seems that the *siv-mode* library is presently structured in such

[1] https://crypto.stackexchange.com/questions/20941/why-shouldnt-i-use-ecb-encryption

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

a way that referencing the *SIV* mode construction from within the library itself (instead of ECB mode) is not straightforward. Nevertheless, it is recommended to either:

- Refactor the library so that *SIV* mode can be called by default immediately from within the adapter specified in *JceAesBlockCipher.java,* instead of AES/ECB.
- Change the default behavior to a semantically secure alternative such as AES/CTR. A fallback to AES/ECB can remain as an option for when more secure alternatives are not available.

Conclusions

The results of this assessment against the cryptographic libraries employed by the Tresor apps by 1&1 Mail & Media GmbH are favorable for the examined items. The tested libraries generally held up to Cure53's scrutiny and stood fairly strong against a range of review approaches.

The cryptographic implementation exhibited a quite exceptional level of robustness, even though one finding was ultimately deemed as "Critical". Still, the assessment's findings are very few and far between, in addition demonstrating that no issues could be tied to threatening the Tresor apps' security and integrity in the long-run. While certain issues and imperfections were noticed and reported, the cryptographic strategies employed by the Tresor apps remain sound and adequate. In other words, the security in this realm is sufficient for the stated use-case, being further boosted by exposing a very small attack surface.

To conclude, the scoped relevant cryptographic libraries of the Tresor apps make a very good impression. The main recommendation is to continue on this path of security dedication, with additional piece of advice concerning documentation of development best practices.

Cure53 would like to thank Michael Ingelbach and Daniel Kefer of 1&1 Mail & Media GmbH for their excellent project coordination, support and assistance, both before and during this assignment.