Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

Pentest-Report Briar Project App & Protocol 03.2017

Cure53, Dr.-Ing. M. Heiderich, Dipl.-Ing. A. Aranguren, Dipl.-Ing. A. Inführ,
BSc. F. Fäßler, MSc. C. Kean, N. Kobeissi

Index

- Introduction
- Scope
- Cryptography Review
- Identified Vulnerabilities
  - BRP-01-001 Mobile: Lack of screenshot protections (Medium)
  - BRP-01-004 Mobile: Panic app prompt bypass via Tapjacking (Medium)
  - BRP-01-005 Mobile: Arbitrary sign out via PanicResponderActivity (Low)
  - BRP-01-006 Mobile: DNS leak via RSS Import (High)
  - BRP-01-008 Mobile: User disruption via exposed activities (Medium)
  - BRP-01-009 Mobile: Possible Intent hijacking via PendingIntent (Medium)
- Miscellaneous Issues
  - BRP-01-002 Mobile: Possible logcat leakage via SDK version (Info)
  - BRP-01-003 Mobile: Possible Information Leakage via Debuggable Flag (Info)
  - BRP-01-007 Mobile: App Crash via unsupported Intent (Info)
  - BRP-01-010 Crypto: DoS in StreamEncrypterImpl.java via paddingLength (Info)
  - BRP-01-011 Crypto: Password strength indicator seems useless (Info)
  - BRP-01-012 Crypto: Unnecessary combination of two CSPRNGs (Info)
- Conclusions

Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

Introduction

"Briar is a messaging app designed for activists, journalists, and anyone else who needs a safe, easy and robust way to communicate. Unlike traditional messaging tools such as email, Twitter or Telegram, Briar doesn't rely on a central server - messages are synchronized directly between the users' devices. If the Internet's down, Briar can sync via Bluetooth or Wi-Fi, keeping the information flowing in a crisis. If the Internet's up, Briar can sync via the Tor network, protecting users and their relationships from surveillance."

From https://briarproject.org/how-it-works.html

This report documents the findings of a penetration test and source code audit carried out by Cure53 against the Briar secure messenger application. The project was completed in March 2017 and revealed the existence of twelve security-relevant issues. Six Cure53 testers were involved in this assignment, the completion of which took a total of thirteen days.

The core application in scope was the Briar messenger application for Android, which was complemented with the review of the protocols specified and used by the Briar product, notably BQP[1], BSP[2] and BTP. Methodology-wise, the assessment followed a white-box approach, meaning that the testing team had access to the Android application's full sources and could take advantage of the provided debug builds. The Briar team has further supplied the Cure53 testers with several APKs, specifically tweaked to enable more efficiency within the testing process, especially for the scenarios linked to the use of the Tor network. All components positioned in the scope of this test have undergone thorough reviews and audit of the code, while respective implementations were additionally examined when applicable.

The aforementioned total number of twelve different findings comprises six actual security vulnerabilities and six additional general weaknesses. The great majority of discoveries received a low-to-medium severity ranking, and some were documented for the sake of completeness and informational value rather than as actual and current risks.

Scope

- **Briar Project Android App APK was shared**
- **Briar Project Android App Sources were shared**

[1] https://code.briarproject.org/akwizgran/briar-spec/blob/master/protocols/BQP.md
[2] https://code.briarproject.org/akwizgran/briar-spec/blob/master/protocols/BSP.md

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

Cryptography Review

Three days were dedicated to a cryptography review of the Briar Android App. This included a specification review for the BQP[3], BSP[4] and BTP[5] protocols provided by the Briar team. While BQP was quickly deemed to be specified in a sound manner, the BSP and BTP specifications lacked details and required further communication with the Briar team. Ultimately, all three specifications could be considered appropriate.

While a full review of the Briar cryptographic code was conducted, the actual series of tasks began with the code located in the specific directory, notably *bramble-core/src/main/java/org/briarproject/bramble/crypto/*. All function calls were then traced from this location.

The code was found to be exceptionally clear and sound, with no vulnerabilities spotted within the scope of this two-day review. However, minor discoveries pertaining to the code have been noted and can be found as three miscellaneous issues documented in this report under BRP-01-010, BRP-01-011 and BRP-01-012.

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. *BRP-01-001*) for the purpose of facilitating any future follow-up correspondence.

BRP-01-001 Mobile: Lack of screenshot protections *(Medium)*

It was found that the Briar Android application currently fails to leverage the available Android protections to avoid side-channel data leakage via screenshots. This allows applications with either screen recording or root privileges to directly capture all information that is being displayed by the Briar app. The issue can be verified as one runs the following ADB Commands[6] while the application is open.

ADB Commands:

```
adb shell screencap -p /mnt/sdcard/screenshot1.png
adb pull /mnt/sdcard/screenshot1.png
```

[3] https://code.briarproject.org/akwizgran/briar-spec/blob/master/protocols/BQP.md
[4] https://code.briarproject.org/akwizgran/briar-spec/blob/master/protocols/BSP.md
[5] https://code.briarproject.org/akwizgran/briar-spec/blob/master/protocols/BTP.md
[6] https://developer.android.com/studio/command-line/adb.html

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

The resulting screenshot displays the information on the screen, as depicted below:

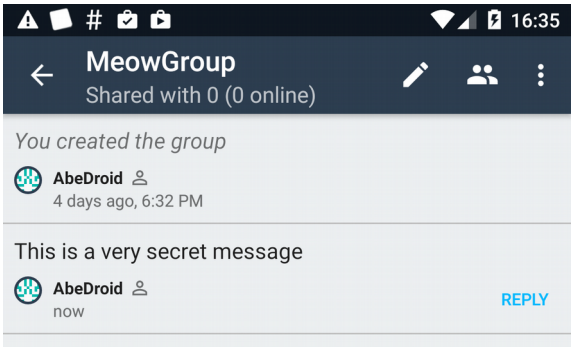

Fig.: Information leak via the screenshot API

It is recommended to ensure that all WebViews have the Android *FLAG_SECURE* flag[7] set. This will guarantee that even the applications running with root privileges are unable to directly capture information displayed by the Briar application on screen.

BRP-01-004 Mobile: Panic app prompt bypass via Tapjacking *(Medium)*

It was found that the Briar Android app fails to mitigate tapjacking attacks. This allows malicious apps to render an overlay, launch an instance of the Briar application in the background, and, ultimately, fool a user into performing actions on the Briar app while they can actually see something different altogether. Although all screens are affected, it was confirmed that the malicious app can also set itself as the *Panic* app. This means that it can set and trigger a *panic* event and effectively delete everything from the Briar app.

Scenario 1: App is unlocked. The *Panic* app prompts a bypass

To demonstrate how the issue occurs, a custom exploit APK was created. A demo was also recorded to highlight how the malicious app is set as the Panic app, then displaying how all of the *Panic* button setup settings are altered, including the consequences of the "*Delete Account*" action (which deletes all data), as well as "*Uninstall Briar*". While this is the main example, it has to be underscored that all other screens (i.e. chats, general settings, Tor configuration, etc.) are also affected by this problem.

https://cure53.de/exchange/792346243678/Tapjacking_PoC2.zip

Scenario 2: App is locked. The *Delete* button prompts a bypass

[7] http://developer.android.com/reference/android/view/Display.html#FLAG_SECURE

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

In the event of the Briar app being locked, a malicious application can still delete all user-information via two alternative buttons, namely "*I have forgotten my password*" and "*Delete*". These are also vulnerable to tapjacking:

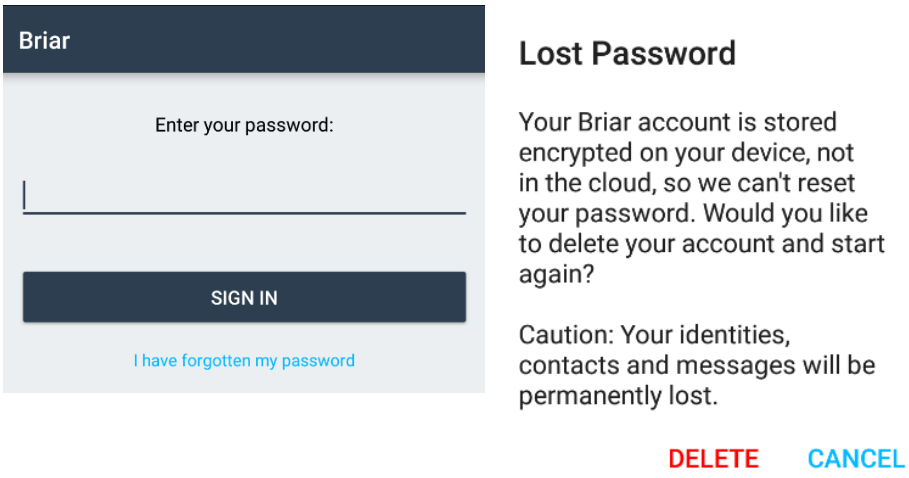

Fig.: User-data being wiped when the app is locked via tapjacking

It is recommended to implement the *filterTouchesWhenObscured*[8][9] attribute at the Android WebView level[10]. This will ensure that all taps from any potentially malicious apps rendered on top are ignored, thus eradicating this attack vector. Ideally, this should be implemented in a base view, meaning that other views are to inherit the protection. The proposed approach will reduce the risk of human error linked to leaving certain buttons unprotected.

BRP-01-005 Mobile: Arbitrary sign out via *PanicResponderActivity* *(Low)*

The Briar application exposes an activity responsible for handling *panic* situations. Although this functionality is intended to exclusively process actions from trusted applications, it was found that the *sign out* fragment of the processing code is executed regardless of the invoking app. As a result, a malicious app could leverage this weakness to continuously *sign out* the Briar user, sending and crafting intents in the background. Therefore, a legitimate app use would be prevented, while the user would additionally have no way of telling what has happened. The latter is due to the fact that the application's UI would only testify to a *loading* process of an unspecified type. Please note that this is the same intent that a malicious application could send to delete all application data as a result of exploiting BRP-01-004.

[8] http://developer.android.com/reference/androi....html#setFilterTouchesWhenObscured(boolean)
[9] http://developer.android.com/reference/android/...View.html#attr_android:filterTouchesWhenObscured
[10] https://cordova.apache.org/docs/en/latest/guide/platforms/android/webview.html

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

This issue was confirmed with the use of the following ADB Command. It sends an intent to trigger the *panic* response, which the app simply processes.

ADB Command:

```
adb shell am start -a "info.guardianproject.panic.action.TRIGGER" -n
"org.briarproject.briar/org.briarproject.briar.android.panic.PanicResponderActiv
ity"
```

Resulting Logcat trace:

```
03-03 18:54:27.477 I/ActivityManager(  966): START u0
{act=info.guardianproject.panic.action.TRIGGER flg=0x10000000
cmp=org.briarproject.briar/.android.panic.PanicResponderActivity} from uid 0 on
display 0
03-03 18:54:27.536 I/PanicResponderActivity( 3231): Signing out...
```

This issue would be likely to cause confusion among the affected users because the app's UI does not show that the user has been signed out. What seemingly transpires instead is the never-ending loading of all chats, forums, and similar components. The user can work around this problem by manually signing out and logging in again, but the malicious app only needs to send another intent to have the process repeated and occur once again.

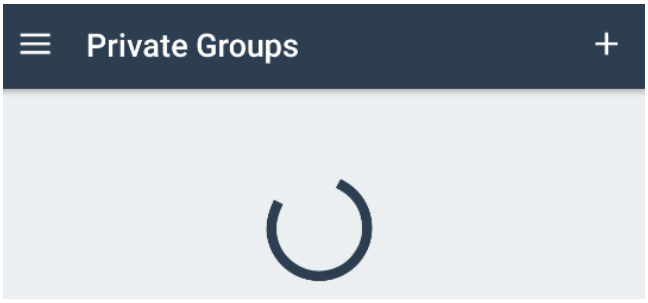

Fig.: All UI screens seem to be loading but the user is signed out.

It is recommended to stop all processing when the intent arrives from an untrusted application.

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

BRP-01-006 Mobile: DNS leak via RSS Import *(High)*

An end-user expects all traffic issued by the application to be tunneled through Tor. It was discovered that the DNS lookup[11], linked to the domain specified in the RSS import, gets leaked on the local network. The reason behind this issue is not having it sent through Tor. An attacker who can sniff the victim's traffic can not only obtain the victim's real IP address, but may also replace the DNS response and point it to its own server.

Steps to Reproduce:

1. The attacker starts to sniff the victim's traffic, for example via a MitM (Man-in-the-Middle) attack;
2. The victim imports an RSS feed, for example http://whatever.com/rss.xml

The following traffic can be seen on the wire:

```
12:29:08.806309 IP 192.168.0.157.42166 > google-public-dns-a.google.com.domain:
55164+ A? whatever.com.
```

This issue can be verified by changing the Android DNS settings. Specifically, a DNS server on the local network can be employed. The traffic needs to be observed, for example with the use of *dnschef*[12]:

Command:

```
dnschef -i 192.168.7.231 --nameservers=192.168.7.1
```

Output:

```
[07:34:26] 192.168.7.193: proxying the response of type 'A' for whatever.com
[07:34:26] 192.168.7.193: proxying the response of type 'A' for whatever.com
```

It is recommended to use *SOCKS 4a* protocol for connecting to the Tor socket. This approach ensures that DNS lookups are protected via the Tor protocol too[13]. As Android lacks proper support for the *SOCKS 4a* protocol, it would be necessary to import another third-party library to rely on. As this could introduce new security issues, it is a priority for the necessity of reliable anonymity to be analyzed. Alternatively, it could be decided that the risk is acceptable, though the users must be informed about the matter at hand in that case.

[11] https://blog.udemy.com/dns-lookup-command/
[12] http://thesprawl.org/projects/dnschef/
[13] https://www.reddit.com/r/DarkNetMarkets/comme...opseccomputer_android_orbottorweb_socks/

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

BRP-01-008 Mobile: User disruption via exposed activities *(Medium)*

It was found that the Briar application exposes and processes intents sent to multiple activities. Although some of these do not process extras, they can still be valuable for a malicious application seeking to disrupt Briar users. More specifically, a malicious app running in the background could continuously send crafted intents to annoy Briar users until they decide to uninstall the app.

This issue can be confirmed when the following ADB Commands are run either on an emulator or on a real phone connected to a host system.

ADB Commands:

```
adb shell am start -a "android.intent.action.MANAGE_NETWORK_USAGE" -n
"org.briarproject.briar/org.briarproject.briar.android.settings.SettingsActivity
"
adb shell am start -a "info.guardianproject.panic.action.CONNECT" -n
"org.briarproject.briar/org.briarproject.briar.android.panic.PanicPreferencesAct
ivity"
adb shell am start -a "android.intent.action.MAIN" -n
"org.briarproject.briar/org.briarproject.briar.android.splash.SplashScreenActivi
ty"
```

The above sequence displays the Briar settings, then revealing also the *panic* settings. Ultimately, it shows the *Splash* screen logging the user out. Sending this sequence of intents in a constant manner will cease an option of a normal usage of the Briar application. Consequently, it would entice users to uninstall the Briar product.

It is recommended to reduce the number of exported activities so that less attack surface is exposed to untrusted applications. The remaining activities could then be protected with a permission or, where possible, the sender of the intent could be subjected to verification[14].

BRP-01-009 Mobile: Possible Intent hijacking via PendingIntent *(Medium)*

It was found that the Briar Android app uses a local intent to create an instance of *PendingIntent* without passing an explicit *Intent Class*. This allows malicious apps to redirect or modify the intent in question while keeping up with the Briar app permissions. The issue can be found on the location provided next, wherein one can observe that the clear intent is never passed explicitly for creating a new instance of *PendingIntent*:

File:
src/main/java/org/briarproject/briar/android/AndroidNotificationManagerImpl.java

[14] https://dev.guardianproject.info/projects/trustedintents/wiki

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

Affected Code:

```
344   // Clear the counters if the notification is dismissed
345   Intent clear = new Intent(CLEAR_PRIVATE_MESSAGE_ACTION);
346   PendingIntent delete = PendingIntent.getBroadcast(appContext, 0,
347                          clear, 0);
[...]
452   // Clear the counters if the notification is dismissed
453   Intent clear = new Intent(CLEAR_GROUP_ACTION);
454   PendingIntent delete = PendingIntent.getBroadcast(appContext, 0,
455                          clear, 0);
[...]
548   // Clear the counters if the notification is dismissed
549   Intent clear = new Intent(CLEAR_FORUM_ACTION);
550   PendingIntent delete = PendingIntent.getBroadcast(appContext, 0,
551                          clear, 0);
[...]
644   // Clear the counters if the notification is dismissed
645   Intent clear = new Intent(CLEAR_BLOG_ACTION);
646   PendingIntent delete = PendingIntent.getBroadcast(appContext, 0,
647                          clear, 0);
```

The official Android documentation includes clear statement concerning the *getBroadcast* method of *PendingIntent*[15]:

> *"For security reasons, the Intent you supply here should almost always be an explicit intent, that is specify an explicit component to be delivered to through Intent.setClass"*

It is recommended to provide the target intent class explicitly to solve this problem. For additional background and guidance on this type of issues, please see the relevant secure coding rule.[16]

[15] https://developer.android.com/reference/andro...roid.content.Context, int, android.content.Intent, int)
[16] https://www.securecoding.cert.org/confluence/...s+pass+explicit+intents+to+a+PendingIntent

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

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.

BRP-01-002 Mobile: Possible logcat leakage via SDK version *(Info)*

It was found that the Briar Android app may inadvertently leak data to other apps via logcat messages. This is due to the fact that the application supports Android SDK version 14. Please note that applications without root privileges can read all logcat messages up to the Android 4.1 (API 16) version.

File:
AndroidManifest.xml

Affected Code:

```
<uses-sdk android:minSdkVersion="14" android:targetSdkVersion="22"/>
```

Example Tor startup information in logcat:

```
03-03 16:32:47.072 I/PluginManagerImpl( 3231): Starting plugin
org.briarproject.bramble.lan took 5 ms
03-03 16:32:47.585 I/TorPlugin( 3231): Mar 03 15:32:47.584 [notice] Tor v0.2.7.6
running on Linux with Libevent 2.0.22-stable, OpenSSL 1.0.2e and Zlib 1.2.8.
03-03 16:32:47.586 I/TorPlugin( 3231): Mar 03 15:32:47.586 [notice] Tor can't
help you if you use it wrong! Learn how to be safe at
https://www.torproject.org/download/download#warning
03-03 16:32:47.589 I/TorPlugin( 3231): Mar 03 15:32:47.588 [notice] Read
configuration file "/data/user/0/org.briarproject.briar/app_tor/torrc".
03-03 16:32:47.600 I/TorPlugin( 3231): Mar 03 15:32:47.600 [notice] Opening
Control listener on 127.0.0.1:59051
[...]
03-03 21:02:02.858 I/TorPlugin( 6878): WARN Received http status code 404 ("Not
found") from server '164.132.97.234:9001' while fetching
"/tor/keys/fp/585769C78764D58426B8B52B6651A5A71137189A+80550987E1D626E3EBA5E5E75
A458DE0626D088C".
```

It is recommended to consider increasing the minimum SDK version to the API 16 at the very minimum.

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

BRP-01-003 Mobile: Possible Information Leakage via Debuggable Flag *(Info)*

It was found that the APK supplied for testing has the *debuggable* flag[17] enabled. This could result in data leakage and should not be set in production builds.

```
<application android:allowBackup="false" android:debuggable="true"
android:icon="@drawable/ic_launcher" android:label="@string/app_name"
android:logo="@mipmap/ic_launcher_round"
android:name="org.briarproject.briar.android.BriarApplicationImpl"
android:supportsRtl="true" android:theme="@style/BriarTheme">
```

It is recommended to have the deployment strategy ensure that the *debuggable* flag is set to false.

BRP-01-007 Mobile: App Crash via unsupported Intent *(Info)*

It was found that links in manually created blog posts can specify any protocol. This can be used to specify the *intent://* protocol handler[18], which makes creation of intents possible. As soon as a victim clicks on a malicious *intent://* link, which points to the Briar app, the app will crash.

Steps to Reproduce:

1. The attacker creates a blog post with the following payload:

```
<a
href="intent://test#package=org.briarproject.briar;action=info.guardianpr
oject.panic.action.CONNECT;component=org.briarproject.briar.android.panic
.PanicPreferencesActivity">asdf</a>
```

2. A contact of the attacker reads the blog post and clicks on the link.
3. Confirm the dialog.
4. The Briar app crashes.

```
ACRA caught a ActivityNotFoundException for org.briarproject.briar
android.content.ActivityNotFoundException: No Activity found to handle
Intent { act=android.intent.action.VIEW dat=Internet://test }
at
android.app.Instrumentation.checkStartActivityResult(Instrumentation.java
:1798)
```

File:
briar-android\src\main\java\org\briarproject\briar\android\blog\WriteBlogPostActivity.java

[17] http://developer.android.com/guide/topics/manifest/application-element.html#debug
[18] https://developer.android.com/guide/components/intents-filters.html

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

Code:

```
public void onSendClick(String body) {
  // hide publish button, show progress bar
  [...]
  body = StringUtils.truncateUtf8(body, MAX_BLOG_POST_BODY_LENGTH);
  // <-- Apply HTML sanitization here
  storePost(body);
 }

private void storePost(final String body) {
  runOnDbThread(new Runnable() {
   @Override
   public void run() {
    long now = System.currentTimeMillis();
    try {
     LocalAuthor author = identityManager.getLocalAuthor();
     BlogPost p = blogPostFactory
       .createBlogPost(groupId, now, null, author, body);
     blogManager.addLocalPost(p);
     postPublished();
```

It is recommended to only allow certain protocols - like HTTP, HTTPS and FTP - in the links to blog posts. The same whitelist is already applied to the imported RSS feeds and should be equally employed for all manually created blog posts.

BRP-01-010 Crypto: DoS in StreamEncrypterImpl.java via paddingLength *(Info)*

During the crypto audit it was found that the *writeFrame* function in the *StreamEncrypterImpl.java* file could cause a Denial of Service (DoS) in the Briar's cryptographic subsystem, provided that it passes a negative *paddingLength* value. The explanation of this flaw is linked to a bounds check that is implemented for other input values, yet it is not implemented for this particular value.

File:
bramble-core/src/main/java/org/briarproject/bramble/crypto/StreamEncrypterImpl.java

Affected Code:

```
public void writeFrame(byte[] payload, int payloadLength,
 int paddingLength, boolean finalFrame) throws IOException {...}
```

It is recommended to check the length of the value and agree on throwing an *IllegalArgumentException* in case of an invalid value.

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

BRP-01-011 Crypto: Password strength indicator seems useless *(Info)*

During the crypto audit, doubts arose with regard to the indicator informing about the strength of a password. It is unclear whether the password strength mechanism defined in the file *PasswordStrengthEstimatorImpl.java* is of real value. As an example, it can be mentioned that the password "*Password1*" gets accepted as *strong* by the algorithm in place.

File:
bramble-core/src/main/java/org/briarproject/bramble/crypto/PasswordStrengthEstimatorImpl.java

Affected Code:

```
private static final int LOWER = 26;
private static final int UPPER = 26;
private static final int DIGIT = 10;
private static final int OTHER = 10;
private static final double STRONG = Math.log(Math.pow(LOWER + UPPER +
     DIGIT + OTHER, 10));
```

It should be considered to replace the currently used implementation with a different approach. The proposed alternative should rely on the password length to a greater degree, and, conversely, become less preoccupied with the combination of all-case alphanumeric and special characters. Simply requiring a minimum length of twelve characters for each password would likely constitute a better policy for having passwords of a certain strength and quality enforced in the application.

BRP-01-012 Crypto: Unnecessary combination of two CSPRNGs *(Info)*

The design of the code in the *CombinedSecureRandom.java* file may not cause any problems *per se* at the moment, but nevertheless seems ill-advised. It is unclear what additional security guarantees are obtained from the process of XORing[19] together outputs from two different CSPRNGs[20], as these are seeded from and operating on the same device.

File:
bramble-core/src/main/java/org/briarproject/bramble/crypto/CombinedSecureRandom.java

Affected Code:

```
class CombinedSecureRandom extends SecureRandom {...}
```

[19] https://en.wikipedia.org/wiki/Exclusive_or
[20] https://en.wikipedia.org/wiki/Cryptographically_secure_pseudorandom_number_generator

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

A single CSPRNG should suffice. The code excerpt this issue extends to seems to be adding unnecessary complexity. In other words, it should probably be removed before becoming a liability.

Conclusions

The results of this Cure53 penetration test and audit of the Briar secure messenger application for Android points to an overall good handling of matters linked to security and privacy in the tested product.

To nevertheless begin with the areas to improve and issues calling for urgent attention, a reaction is needed with reference to the vulnerability marked as “High” in terms of severity and impact. The problem - documented in BRP-01-006 - can clearly lead to the anonymity promise being broken. This is because a user might assume to be protected when performing certain activities via the Tor, yet data leakage remains a real concern in this scenario. The bigger problem is that the issue must be seen as fairly hard to address and fix properly. For that reason, it might be the case that, despite considerable risk, the potential harmful effects are being accepted by the application’s maintainer. In that case, it is still mandatory that the users are informed about the matter at hand. Moving on, a noticeable pattern of insecurity, which could additionally impede user-experience and discourage people from continuing on with the Briar application, can be linked to intents. More specifically, many of the exposed intents failed to properly verify the sender. As a result, an attacker with a rogue app gains an array of possibilities when seeking to interfere with the Briar application. This includes a total wipe of the application and its data by combining some intents with a tapjacking attack. The flaws in this realm require consideration.

With reference to the positive factors, contributing to the final verdict, it has to be underlined that a grand total standing at twelve security-relevant findings is more than satisfactory and acceptable. Even more reassuring is the fact that all but one issue were marked with medium and low rankings, or classified even as informational only. For the six Cure53 testers who completed this assessment, the overall low severity translates to an application with a good understanding of vulnerability patterns and threats. What is more, the majority of issues classified as non-threatening could be seen as being quite easy to fix and unlikely to affect the user-experience in a profound way. Commenting on broader impressions, the testing team stated very that the quality and readability of the app’s source code was rather exceptional, thus making this assessment pleasant and efficient. The communication with the development team during the project was professional and timely, and the questions and requests communicated by the Cure53 team during the test were usually responded to with precision.

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

Last but not least, it must be noted that the tested Briar application is at a very early stage of development and present. Consequently, once the development is completed, a second audit is highly advisable. Still, provided that the documented issues get fixed properly, the application is able to offer a good level of privacy and security. In other words, the Briar secure messenger can be recommended for use.

Cure53 would like to thank Michael Rogers of Briar Project for his excellent project coordination, support and assistance, both before and during this assignment. We would like to further express our gratitude to the Open Technology Fund in Washington D.C., USA, for generously funding this and other penetration test projects, as well as enabling us to publish the results.