Fine penetration tests for fine websites

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

Pentest-Report Roonak App & Server 06.2017

Cure53, Dr.-Ing. M. Heiderich, Dipl.-Ing. A. Aranguren, Dipl.-Ing. A. Inführ, BSc. D. Weißer, BSc. C. Kean

Index

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

Introduction

"Roonak is an incident-report submission platform allowing users to send text and media files to the Hrana news organization's report database."

This report documents the findings of a security assessment performed by Cure53 team against the Roonak platform. The project was completed over the course of eight days in June 2017. Five members of the Cure53 team performed this assessment, which comprised a mobile penetration test, entailed an investigation of the server configuration, and, last but not least, incorporated a source code audit. It has to be noted that the test was curated by United4Iran (U4I). In January 2018, the Cure53 fix verification took place and this final report was created.

As for the approach, the test followed the so-called white-box methodology, which signifies access to various relevant items, including server, URLs, builds, sources and credentials, as well as other necessary entities covered by the scope. Detailed description of the covered aspects should mention that, besides source code audit of the mobile application's code, tests were carried out against the applications running on an emulator, notably Genymotion. The backend component was subject to configuration assessment and targeted source code audit of its API. At this stage a web backend was only examined to a certain extent, yet this realm will be covered within another follow-up project.

Full coverage was reached regarding the test items in scope and the assessment revealed eleven security-relevant issues. These findings could further be demarcated into two categories of six actual security vulnerabilities and five general weaknesses. It should be emphasized that one issue was flagged with the utmost "Critical" severity. The core problem discovered in the case of this vulnerability was a user's capacity to cause persistent XSS. This further not only occurred through a simple process of reporting news items and using the application as designed, but also affected the Admin backend. The problem yielded an option of having unintended admins, which is deemed critical under the application's goals and premise. On the positive note, this issue had been live-reported and could since be verified as fixed.

The report will now present and discuss each finding separately, providing mitigation advice in order to facilitate fixes. A conclusion delivered in the final section of this document additionally comments on the overall state of security at the Roonak application compound.

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

Scope & Test Parameters

- **Roonak Android App**
  - Sources and APK were shared with Cure53
- **Roonak Server Backend**
  - Sources & credentials were shared with Cure53
- **Roonak API & Web backend**
  - Sources & credentials were shared with Cure53

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

RNK-01-001 Android: Feed Manipulation via Lack of Cert Pinning *(Low)*

Note: This issue was declared to be of acceptable risk and no fix is planned for now.

It was found that the Roonak Android application fails to protect TLS communications with Pinning. This allows malicious attackers with a certificate trusted by the Android trust store (i.e. most governments, some companies) to observe and modify network communications. This attack is mitigated by the cryptographic protection implemented to protect user reports, but other traffic, such as the feed retrieval from *www.hra-news.org,* is unprotected. Therefore, a malicious attacker could modify the aforementioned traffic and could leverage this weakness to spoof news for the Roonak users.

The following request was captured intercepting TLS traffic with a certificate trusted by the Android trust store. A modification of this response allows spoofing of all news for the Roonak user.

Request:

```
GET /feed/ HTTP/1.1
Host: www.hra-news.org
[...]
```

Response:

```
HTTP/1.1 200 OK
[...]
```

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

```
<?xml version="1.0" encoding="UTF-8"?><rss version="2.0"[...]

<channel>
       <title>Ø®Ø¨Ø±Ú¯Ø²Ø§Ø±Û Ù‡Ø±Ø§Ù‡Ù†Ø§</title>
       <atom:link href="https://www.hra-news.org/feed/"
              rel="self" type="application/rss+xml" />
       <link>https://www.hra-news.org</link>
       <description>ارگان خبری مجموعه فعلان حقوق بشر در ایران</description>
       <lastBuildDate>Mon, 05 Jun 2017 06:54:49 +0000</lastBuildDate>
       <language>fa-IR</language>
       <sy:updatePeriod>hourly</sy:updatePeriod>
       <sy:updateFrequency>1</sy:updateFrequency>
       <generator>https://wordpress.org/?v=4.7.5</generator>
       <item>
              <title>با گشایش پرونده ای جدید، یوسف عمادی تفهیم اتهام شد</title>
              <link>https://www.hra-news.org/2017/hranews/a-10951/</link>
              <comments>https://www.hra-news.org/2017/hranews/a-10951/#respond
                     </comments>
              <pubDate>Mon, 05 Jun 2017 06:50:07 +0000</pubDate>
              <dc:creator><![CDATA[bank1]]></dc:creator>
                           <category><![CDATA[slide]]></category>
              <category><![CDATA[زندانیان]]></category>
              [...]
```

The root cause of this issue can be found on the following location.

File:
Hrana/app/src/main/java/org/hrana/roonak/view/fragment/NewsFragment.java

Affected Code:

```
 41 import info.guardianproject.netcipher.client.StrongBuilder;
 42 import info.guardianproject.netcipher.client.StrongConnectionBuilder;
 43 import info.guardianproject.netcipher.proxy.OrbotHelper;
[...]
 54 public class NewsFragment extends Fragment implements IViewContract.RssView,
 55         StrongBuilder.Callback<HttpURLConnection> {
[...]
 59     private final String URL = "https://www.hra-news.org/feed",
 60             URL_EN = "https://www.en-hrana.org/feed", ENTRIES = "entries";
[...]
183     /*
184      * Load RSS Feed from Hrana-news website. If Orbot is installed and
Orbot
             use is enabled, attempts to load feed over Tor.
185      * If Orbot is not installed and user has not requested that access to
             Hrana-News website
186      * be restricted, loads feed without Orbot.
187      *
```

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

```
188      * @todo: currently automatic in-app Tor/Orbot detection and use is
disabled, but users can still configure Orbot themselves to tunnel apps such as
Roonak.
189      */
190     void loadFeed(String targetUrl) {
191         if (OrbotHelper.isOrbotInstalled(getContext()) && CAN_ATTEMPT_ORBOT)
{
192             try {
193                 java.net.URL url = new URL(targetUrl);
194                 StrongConnectionBuilder.forMaxSecurity(getActivity())
195                         .withTorValidation()
196                         .withWeakCiphers()
197                         .withBestProxy()
198                         .connectTo(url)
199                         .build(this);
200             } catch (Exception ex) {
201                 Log.e(TAG, ex.getMessage());
202                 showError();
203             }
204         } else if (!preferences.getBoolean
205                 (getString(R.string.restrict_access_tor_key), false)) { //
Try regular way if user allows non-Tor access
206             Log.d(TAG, "Loading news results without Tor according to user
preferences.");
207             loadFeedFallback(targetUrl);
208         } else {
209             showNoNetworkAccess();
210         }
211     }
```

The demonstration above shows that the feed will be loaded via *Tor* when Orbot[1] is installed. This mechanism increases the likelihood of the connection being intercepted by a government since many state-level entities are known to run *Tor* exit nodes[2]. However, regardless of the Orbot usage, there are no traces of Pinning protections for the entirety of the feed retrieval process.

It is recommended to protect TLS communications with Pinning. For this purpose, the *OWASP Pinning Cheat Sheet*[3] could be used. Note that guidelines provided in the specified documentation includes Android examples. In this particular case Pinning protections can be implemented with the *NetCipher* library using the *withTrustManagers* directive[4]. The proposed approach would leave all existing functionality unaltered.

[1] https://guardianproject.info/apps/orbot/
[2] https://hackertarget.com/tor-exit-node-visualization/
[3] https://www.owasp.org/index.php/Pinning_Cheat_Sheet
[4] https://github.com/guardianproject/NetCipher

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

RNK-01-002 Android: Phone calls via missing TapJacking protections *(Low)*

Note: This issue was fixed and the fix was verified by Cure53.

It was found that the Roonak application does not fully mitigate TapJacking attacks. The app only selectively implements protections on certain buttons, while others can still be exploited. This issue was verified with a PoC on the Android app which, despite not having phone permissions[5], could ring Roonak Legal advice numbers by relying on TapJacking only. The issue can be verified with the Cure53's PoC APK supplied alongside a short demo recorded for demonstration purposes[6].

Affected XML Layouts:
src/Hrana/app/build/intermediates/res/merged/debug/layout/fragment_appintro.xml
src/Hrana/app/build/intermediates/res/merged/debug/layout/item_legalcontact.xml
src/Hrana/app/src/main/res/layout/fragment_appintro.xml
src/Hrana/app/src/main/res/layout/item_legalcontact.xml

Contents:
```
android:filterTouchesWhenObscured="true"
```

Affected Java Files:
src/Hrana/app/src/main/java/org/hrana/roonak/view/button/FilterTouchButton.java
src/Hrana/app/src/main/java/org/hrana/roonak/view/button/FilterTouchFAB.java
src/Hrana/app/src/main/java/org/hrana/roonak/view/button/FilterTouchFAMenu.java

Contents:
```
setFilterTouchesWhenObscured(true);
```

Implementing the *filterTouchesWhenObscured*[7][8] attribute at the Android WebView level[9] is recommended as a way to eradicate this issue. This will ensure that all taps from the potentially malicious apps rendered on top are ignored, thus eliminating the attack vector. Ideally, this should be implemented in a BaseView that other views inherit, thus reducing the potential for human error tied to leaving certain buttons unprotected, as illustrated in this finding. Please note that screenshot leakage mitigations are currently already implemented on the *BaseActivity*, which is also the location where TapJacking protections should be located.

[5] https://developer.android.com/reference/android/Manifest.permission_group.html#PHONE
[6] https://cure53.de/exchange/2437856924356/5437236856572.zip
[7] http://developer.android.com/reference/an...iew.html#setFilterTouchesWhenObscured(boolean)
[8] http://developer.android.com/reference/andr...View.html#attr_android:filterTouchesWhenObscured
[9] https://cordova.apache.org/docs/en/latest/guide/platforms/android/webview.html

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

File:
src/Hrana/app/src/main/java/org/hrana/roonak/view/activity/BaseActivity.java
Code:

```
20      protected void onCreate(@Nullable Bundle savedInstanceState) {
21          super.onCreate(savedInstanceState);
22
23          getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
24                  WindowManager.LayoutParams.FLAG_SECURE);
25      }
```

RNK-01-005 Web: Persistent XSS in Admin backend *(Critical)*

Note: This issue was fixed and the fix was verified by Cure53.

It was found that the Admin backend for reviewing user-submitted reports is vulnerable to persistent Cross-Site Scripting (XSS). This lets unauthenticated application users submit JavaScript that will execute in the security context of the logged in administrator. The process will take place immediately after a successful admin login. Building on this step, a malicious attacker could access other user-submitted reports and impersonate the administrator on any action available via the Admin backend.

The issue can be verified from the mobile app as one submits HTML characters and observes them resulting in an XSS on the backend. A report is supplied below to showcase how the submission occurs.

Sample Report:

```
testing please ignore <svg onload=alert(1)>
```

When an Admin user successfully logs in, the user is immediately directed to the *reports* endpoint. The XSS is triggered at this location.

URL:
https://dev1.qwp.symbiotic.coop/reports

Rendered HTML:

```
<div>testing please ignore <svg onload=alert(1)>
```

It is recommended to output-encode user-input. The encoding should take place in the security context of the location of the HTML page in which the user-input is rendered. For additional details and mitigation guidance on this matter, please consult the *OWASP XSS Prevention Cheat Sheet*[10]. In this specific situation, it is probably sufficient to

[10] https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

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

encode all HTML characters. Should it be necessary for users to submit HTML data, a sanitization strategy needs to be explored.

Note: This issue was live-reported and fixed when the penetration test was still ongoing. The proposed fix was positively verified by Cure53.

RNK-01-006 Server: Generally weak Configurations in place *(Medium)*

Note: This issue was fixed and the fix was verified by Cure53.

Most Linux default installations have several security options disabled due to requiring individual work or possibly affecting the system's usability for the majority of users. There are several configuration options listed below and proven to significantly raise the security-bar for a Linux server.

Hidepid:
Every user can see all of the processes and their respective parameters on a Linux server. Under certain premise, this behavior might leak information or point an attacker in the right direction when it comes to escalating privileges. *Hidepid* is an option that can be activated when the *procfs*[11] is mounted. This can be achieved with the following entry inside the server's *fstab*.

Command:
```
$ cat /etc/fstab
```

Output:
```
[...]
proc            /proc           proc        hidepid=2       0 0
```

If enabled, a non-root user can exclusively see the processes that were started by them and not by others.

Dmesg Restrict:
Dmesg[12] is a Linux command showing messages printed by the kernel. It contains information about the boot process and hardware. This means that, in some cases, it might disclose information to attackers. In this particular setup even the firewall messages were revealed. The core problem especially holds for an attacker who already has limited privileges on the server and can now escalate to root. There is no reason why a non-root user should see the output in question. It is recommended to restrict the

[11] https://en.wikipedia.org/wiki/Procfs
[12] https://en.wikipedia.org/wiki/Dmesg

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

access to kernel messages to root by adding the following line to the *sysctl* configuration:

```
kernel.dmesg_restrict = 1
```

Kptr restrict:
By default Linux discloses the location of its internal symbols via the */proc/kallsyms* file. Several Linux distros provide the same information via */boot/System.map.* It must be underscored that some privilege escalation kernel exploits rely on values from those files. In other words, it should not be allowed for the non-root users to obtain them. It is recommended to deny non-privileged access by changing the access permissions of */boot* to 700. What is more, the line supplied next should be added to the *sysctl* configuration.

```
kernel.kptr_restrict = 1
```

Remote Syslog:
Alongside local logging, it is advised to set up an external logging server. In case the server is compromised, an attacker can easily remove all evidence from the log files, thus making it difficult to even detect the attack, not to mention preventing the maintainers from understanding the attack that just took place. The consequences would be alleviated had the logs been stored on another server.

Weak file permissions:
It was found that the admin's home directory can be read by any other user on the system. This is a common issue as default directory access flags are typically very permissive. However, as home folders often contain sensitive data, only the respective owner should be allowed to access them.

RNK-01-007 Server: Kernel and Software not up to date *(Medium)*

Note: This issue was fixed and the fix was verified by Cure53.

While investigating the server configuration it was found that the installed kernel and application packages are not up to date. This poses a security risk since old software versions frequently suffer from the known security bugs.

The following packages need to be upgraded:

```
base-files, binutils, ca-certificates, git, git-man, initramfs-tools, libc-bin,
libc-dev-bin, libc6, libc6-dev, libdns-export100, libfreetype6, libgnutls-deb0-
28, libgnutls-openssl27, libicu52, libirs-export91, libisc-export95, libisccfg-
export90, liblcms2-2, libldap-2.4-2, libnss3, librtmp1, libssl-dev, libssl-doc,
```

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

```
libssl1.0.0, libsystemd0, libtasn1-6, libtiff5, libudev1, linux-compiler-gcc-
4.8-x86, linux-headers-3.16.0-4-amd64, linux-headers-3.16.0-4-common, linux-
image-3.16.0-4-amd64, linux-libc-dev, locales, login, multiarch-support,
openssl, passwd, perl, perl-base, perl-modules, python-acme, python-cffi,
python-cffi-backend, python-django-common, sudo, systemd, systemd-sysv, tzdata,
udev, vim-common, vim-tiny, wget
```

The impact of this issue has not been investigated any further, though old packages on a production machine are an indicator of negligence.

RNK-01-008 Server: App user can use sudo without password *(High)*

Note: This issue was fixed and the fix was verified by Cure53.

It was found that the user who runs the application has *sudo* privileges and can gain root access without providing a password. An attacker with command execution capabilities could therefore completely take over the server.

File:

```
cat /etc/sudoers.d/90-cloud-init-users
```

Sudo rule:

```
admin ALL=(ALL) NOPASSWD:ALL
```

It is recommended to run the application with a dedicated user account. Said account should not be granted with more than necessary list of privileges.

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.

RNK-01-003 Android: User disruption via unrequested tutorial launch *(Info)*

Note: This issue was fixed and the fix was verified by Cure53.

It was found that the Roonak Android application will launch its tutorial when the corresponding main activity is invoked with a crafted intent action. A malicious app running in the background could leverage this weakness to repeatedly launch the Roonak tutorial, thus disrupting legitimate users. Since the tutorial forces the user to tap on certain locations before the possibility to conduct other actions is granted, a malicious

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

app continuously launching the tutorial from the background can effectively prevent Roonak users from taking advantage of the product.

This issue can be replicated using the Cure53 PoC app[13] developed for RNK-01-002, specifically with the use of the "*Launch Tutorial*" menu item. Alternatively, running the following ADB command is also possible for accomplishing the desired result. The latter will lock the user in the onboarding tutorial without an option to perform a different action until all tutorial taps are complete.

ADB Command:

```
adb shell am start -a "org.hrana.roonak.launchTutorial" -n
"org.hrana.roonak.debug/org.hrana.roonak.view.activity.MainActivity"
```

File:

src/Hrana/app/src/main/java/org/hrana/roonak/view/activity/MainActivity.java

Affected Code:

```
 75     protected void onCreate(Bundle savedInstanceState) {
 76         super.onCreate(savedInstanceState);
 77         setContentView(R.layout.activity_main);
[...]
150         // See if tutorial launch
151         checkIntent();
[...]
187     private void checkIntent() {
188         Intent intent = getIntent();
189         if (intent != null) {
190             if (intent.getAction() != null
191                     && intent.getAction().equals(INTENT_LAUNCH_TUTORIAL)) {
192                 preferences.edit()
193
.putBoolean(getString(R.string.onboarding_complete_homepage_key), false)
194                         .apply();[...]
```

It is recommended to move the tutorial check to a different and not-exported activity. This will ensure that third-party applications cannot change the preferences of the data logic. In effect, this will prevent the third-party apps from launching the onboarding tutorial in the future.

[13] https://cure53.de/exchange/2437856924356/5437236856572.zip

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

RNK-01-004 Android: Weak URL check on *LegalCaseFragment* *(Info)*

Note: This issue was fixed and the fix was verified by Cure53.

It was found that the *LegalCaseFragment* on the Roonak Android app contains a check to determine whether links on a page are internal or external. This check is weak and can be bypassed trivially by using traversal characters, namely dot and slash. Another option would also be to provide an SD Card path containing the expected *android_asset/web* string. Please note however that, at the moment, this is unexploitable since this page is delivered as static HTML from the app assets and does not accept any user-input.

File:
src/Hrana/app/src/main/java/org/hrana/roonak/view/fragment/LegalCaseFragment.java

Affected Code:

```
95    // @Todo can probably improve this checking method.
96    // Check if url is pointing to internal resource
97    private boolean isInternalUrl(String url) {
98       return url.startsWith("file://") && url.contains("android_asset/web");
99    }
```

Example bypass:
file:///mnt/sdcard/attacker/android_asset/web_xss.html

It is recommended to test if the provided URL starts with *file:///android_asset/web/* and that no "../" sequences are present. If they remain, they could allow an attacker to traverse from a different URL into this folder.

RNK-01-009 API: Multiple Infoleaks due to Django debug settings *(Medium)*

Note: This issue was fixed and the fix was verified by Cure53.

The tests revealed that the API on *api.roonak.org* has the debugging mode enabled. This results in unintended data leakage on every Python exception or error message that the server returns. For example, the API displaying a deviating 404 page for certain paths gives away information on server internals and server routes. This could be used to explore further endpoints, in turn potentially resulting in a possibly larger attack surface.

To provide an example, the following paths created in *reportivist-v0.1.audit/reportivist-v0.1.auditsrc /server/hrvr/urls.py* are disclosed when visiting the following URL at *https://api.roonak.org/api-auth/.*

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

```
^ ^users/$ [name='user-list']
^ ^users\.(?P<format>[a-z0-9]+)/?$ [name='user-list']
^ ^users/(?P<pk>[^/.]+)/$ [name='user-detail']
^ ^users/(?P<pk>[^/.]+)\.(?P<format>[a-z0-9]+)/?$ [name='user-detail']
^ ^groups/$ [name='group-list']
^ ^groups\.(?P<format>[a-z0-9]+)/?$ [name='group-list']
^ ^groups/(?P<pk>[^/.]+)/$ [name='group-detail']
^ ^groups/(?P<pk>[^/.]+)\.(?P<format>[a-z0-9]+)/?$ [name='group-detail']
^ ^$ [name='api-root']
^ ^\.(?P<format>[a-z0-9]+)/?$ [name='api-root']
^api-auth/ ^login/$ [name='login']
^api-auth/ ^logout/$ [name='logout']
^ ^api/v1/submit-report/$
^ ^api/v1/submit-attachment/$
^ ^api/v1/s3/submit-report/$
^ ^api/v1/s3/submit-attachment/$
```

Sending an empty POST request to *https://api.roonak.org/api/v1/submit-attachment/* leaks the information as demonstrated. Among other data, the version number of the Django framework in use and several file paths (database location, base directory, etc.) are disclosed.

PoC:

```
curl -i -s -k  -X $'POST' \
    -H $'Upgrade-Insecure-Requests: 1' -H $'User-Agent: Mozilla/5.0 (X11; Linux
x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.81
Safari/537.36' \
    -b
$'csrftoken=Ggbgw0xTVspzUH9bEctCbpDoQRyjUTXqiJ4pUavxQbdCrwZo8fJ2j3uaZgacFGH4' \
    $'https://api.roonak.org/api/v1/submit-attachment/'
```

Response:

```
Django Version:     1.10.6
[...]
DATABASES [...] 'NAME': '/home/admin/reportivist/src/server/db.sqlite3',
[...]
BASE_DIR [...]  '/home/admin/reportivist/src/server'
```

It is recommended to set the *DEBUG* variable to *false* in the Django *settings* file, which will replace the leaks with a standardized 404 error page. It is assumed that this issue is just a test artifact and would not be present in the production-deployed version.

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

RNK-01-010 API: Outdated Django version in use *(Info)*

Note: This issue was fixed and the fix was verified by Cure53.

Another finding of the test highlighted that the Django framework employed on *api.roonak.org* uses an outdated version (1.10.6). They key information here is that this version suffer from publicly reported *Open Redirect* vulnerabilities[14] for certain functions.

Currently those functions are not in use in the context of the tested web application. By this logic, the issue is presently unexploitable. It is nevertheless recommended to update to the latest version of the Django framework.

RNK-01-011 API: Possible DoS via late message size check *(Info)*

Note: This issue was fixed and the fix was verified by Cure53.

It was found that the API blob decryption process performs a size check only after the decryption process is complete. This makes it possible for an attacker to attempt to DoS the server by sending multiple large requests and forcing the server to decrypt them. The problem can be observed in the following source code location.

File:
reportivist-v0.1.auditsrc/server/reportivist_rest/serializers.py

Affected Code:

```
 74     def create(self, validated_data):
 75         """
 76         Create report instance and submit it to civicrm server, given the
validated data.
 77         """
 78         instance = Report()
 79         try: #loading the encryption key
 80             report_decryptor =
ReportDecryptor(server_settings.RECEIVER_SECRET_KEY_FILENAME, receiver_key_id =
validated_data.get('encryption_key_id'))
 81         except RuntimeError as e:
 82             raise RuntimeError("unable to retrieve the report encryption
key")
 83
 84         #the correct way of doing this to invoke the serializer
 85         #of the report here for validation
 86         try:
```

[14] https://www.cvedetails.com/vulnerability-list/vendor...637/year-2017/Djangoproject-Django-1.10.6.html

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

```
 87             decrypted_report =
json.loads(report_decryptor.decrypt_report(validated_data.get('encrypted_blob'))
)
 88         except:
 89             raise RuntimeError("submitted Encrypted blob is badly formated")
 90         if ((not 'client_id' in decrypted_report.keys()) or
 91                 (not 'report_body' in decrypted_report.keys())):
 92             raise RuntimeError("both client id and report body are
required")
[...]
115         if (len(decrypted_report['report_body']) >
server_settings.MAX_SIZE_OF_SUBMISSION):
116             raise RequestDataTooBig("too big of a report")
117
118         instance.report_body = decrypted_report['report_body']
119         instance.save()
120
121         #counting client's daily submission
122         client_record[today_key] += 1
123         client_record.save()
124
125         return instance
```

It is recommended to perform an additional safety length check on the ciphertext, prior to having it processed. This will guarantee that obvious attacks, such as issuing very large encrypted reports, are not processed. In doing so, the processes cease to be the likely targets of a DoS attack.

Conclusion

The findings stemming from this Cure53 investigation of the Roonak platform point to a rather good overall state of security at the tested items. Commissioned by United4Iran (U4I), this assignment involved five members of the Cure53 team. Cure53 assessed the Roonak platform over a period of eight days in June 2017. All fixed were verified by Cure53 in January 2018.

Discussing the stages and components of the test reaching the desired level of coverage has to first emphasize that the preparation phase completed by U4I had been done in an expert manner. The Cure53 team was given all necessary information and details at the very beginning of the assignments, which means that technical hindrances that commonly impede the test's pace could be avoided. This pattern continued throughout the test as all Cure53's requests were handled promptly and in an accurate manner.

As already mentioned, the number of the spotted issues is actually low in the context of the extensive scope. This impression is further complimented by the relatively

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

uncomplicated nature of the flaws. It is nevertheless worth commenting on the matters pertinent to the items in scope in a more targeted manner. The specific dimensions comprise conclusions regarding the Android application, server, web and API.

The Android application was quickly assessed as exposing a very limited attack. Presently, its singular exposed activity encompasses no extra activities, no content providers, no services, and no intent extras. It is also a good practice that the application opens links externally on the Android browser. Instead of rolling its own cryptographic strategies, which tend to be faulty, the Roonak Android app leverages *libsodium* to ensure meeting its cryptographic goals. Since the news feed is loaded in a TextView, no XSS attacks are possible. The application further manages to correctly validate the TLS certificates. In the realm of the Android app, the findings revolved around the lack of Pinning protections (RNK-01-001), as well as pointed out suboptimal handling of TapJacking issues by deploying incomplete protections (RNK-01-002). The flaws made it possible for third-party apps to send intents launching the tutorial (RNK-01-003), and resulted in an unexploitable but improvable URL check (RNK-01-004).

Moving on to the next component it was clear that the server constitutes an area where improvements should be made. Server hardening efforts must respond to the challenges brought on by the spotted weak configurations (RNK-01-006), and must urgently focus on making sure that the currently out of date kernel and system are upgraded (RNK-01-007). What is more, as it was found that an application user can gain root access without a password (RNK-01-008), more attention is needed in this realm.
The results concerning the web component are somewhat ambivalent. Notably, all user-controlled input-fields were tested for XSS including filename XSS. On the one hand, the single “Critical”-level vulnerability discovered within all tests was linked to web arena. On the other hand the aforementioned and potentially very harmful XSS issue (RNK-01-005) was fixed very quickly by the third-party provider. In this case live-reporting and prompt handling mitigated the perceptions of the final severity of this issue.

The API area boasts an array of security-relevant best practices. Among other realms, the *Settings* files are encrypted with the use of *git-crypt* on the server side. The solution does not roll its own crypto and uses libsodium a strong crypto library instead. Further, no usage of sensitive Python sinks could be discerned during testing. Similarly, close to no string concatenations were found on the API’s source code. In other words, the assessment unveiled well-structured and simple code. For that reason, following the process of handling an incoming request could be carried out with ease. The findings within this area were tied to the *Debug* mode being enabled on the server (RNK-01-009), suggested addressing the outdated Django version (RNK-01-010), and drew attention to the lack of checks on the encrypted blobs prior to processing (RNK-01-011).

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

In conclusion, the results put the Roonak suite in a good light from a security standpoint. It is strongly believed that none of the identified issues should prove overly difficult to fix. Therefore, provided that fixes are deployed and verified, the applications, server and API can be considered production-ready.

Cure53 would like to thank the entire team over at U4I for their excellent project coordination, support and assistance, both before and during this assignment.