



RADICALLY OPEN SECURITY

Penetration Test Report

Réseaux IP Européens Network
Coordination Centre

V 1.0
Diemen, July 23rd, 2021
Confidential

Document Properties

Client	Réseaux IP Européens Network Coordination Centre
Title	Penetration Test Report
Target	The externally (and transitively) accessible applications
Version	1.0
Pentester	Abhinav Mishra
Authors	Abhinav Mishra, Patricia Piolon
Reviewed by	Patricia Piolon
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	July 23rd, 2021	Abhinav Mishra	Initial draft
1.0	July 23rd, 2021	Patricia Piolon	Review

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Overdiemerweg 28 1111 PP Diemen The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Radically Open Security B.V. is registered at the trade register of the Dutch chamber of commerce under number 60628081.

Table of Contents

1	Executive Summary	4
1.1	Introduction	4
1.2	Scope of work	4
1.3	Project objectives	4
1.4	Timeline	4
1.5	Results In A Nutshell	4
1.6	Summary of Findings	5
1.6.1	Findings by Threat Level	6
1.6.2	Findings by Type	6
1.7	Summary of Recommendations	7
2	Methodology	8
2.1	Planning	8
2.2	Risk Classification	8
3	Reconnaissance and Fingerprinting	10
4	Findings	11
4.1	RIPE-009 — Unencrypted Communication	11
4.2	RIPE-008 — Cross Site Request Forgery	12
4.3	RIPE-007 — Improper session management	14
4.4	RIPE-006 — Unrestricted File Upload	16
4.5	RIPE-005 — Password Reset Token Does Not Expire on Email Change	18
4.6	RIPE-004 — Improper input validation	19
4.7	RIPE-003 — Open Redirect to Subdomains	21
4.8	RIPE-001 — Cross-origin Resource Sharing Policy Misconfiguration	22
5	Non-Findings	24
5.1	NF-002 — Non Finding Test Cases	24
6	Future Work	26
7	Conclusion	27
Appendix 1	Testing team	28

1 Executive Summary

1.1 Introduction

Between June 23, 2021 and July 22, 2021, Radically Open Security B.V. carried out a penetration test for Réseaux IP Européens Network Coordination Centre

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of work

The scope of the penetration test was limited to the following target(s):

- The externally (and transitively) accessible applications

The scoped services are broken down as follows:

- Pentest on the externally and transitively accessible systems: 8-10 days
- Reporting: 2 days
- **Total effort: 10 - 12 days**

1.3 Project objectives

ROS will perform a penetration test of the externally (and transitively) accessible applications with RIPE NCC in order to assess the security of the applications and data. To do so ROS will access and guide RIPE NCC in attempting to find vulnerabilities, exploiting any such found to try and gain further access and elevated privileges.

1.4 Timeline

The Security Audit took place between June 23, 2021 and July 22, 2021.

1.5 Results In A Nutshell

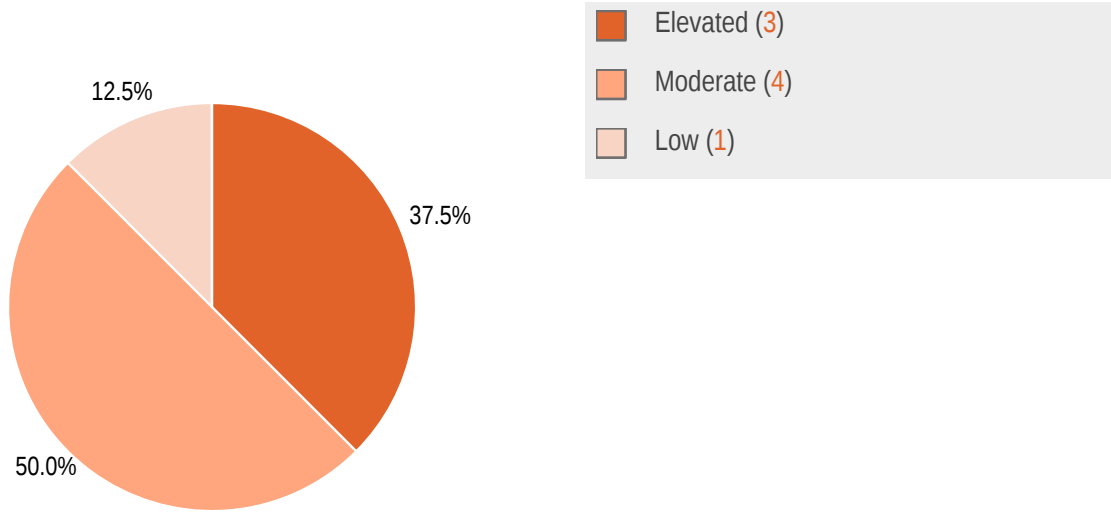
During this grey-box penetration test we found 3 Elevated, 4 Moderate and 1 Low-severity issue. The Elevated-severity issues are related to unencrypted communication, unrestricted file uploads and an open redirect. These vulnerabilities may be exploited to perform a man-in-the-middle (MitM) attack, upload malicious files to the server and/or trick users into visiting a phishing page.

The Moderate- and Low-severity issues arise from a lack of security controls against Cross-site Request Forgery (CSRF), the insecure handling of password reset links and problems with input validation, CORS configuration and session management. Fixing these issues will further improve the defense-in-depth of the applications, resulting in a more secure application suite.

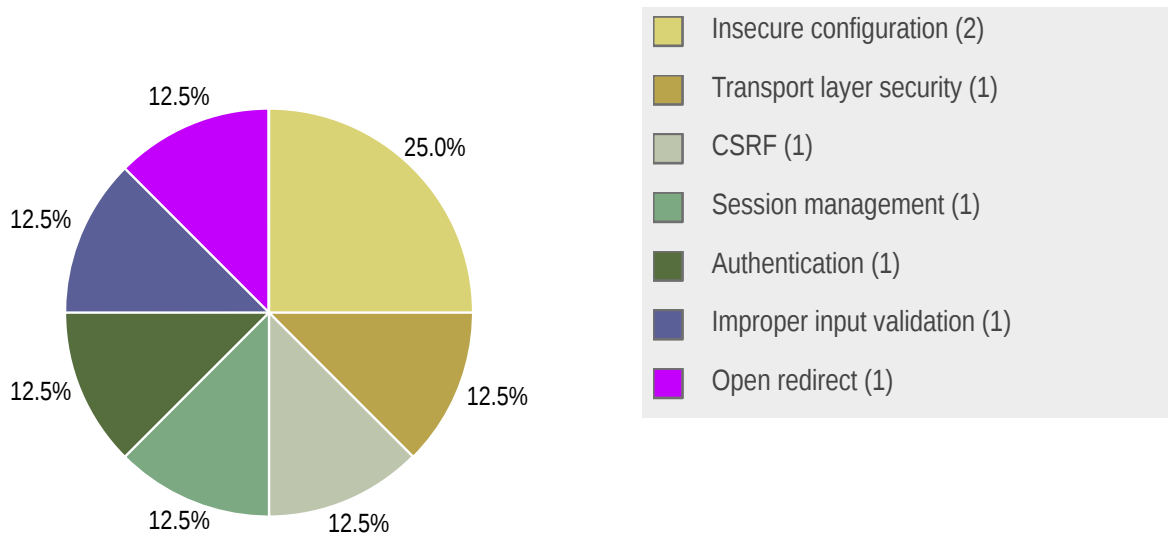
1.6 Summary of Findings

ID	Type	Description	Threat level
RIPE-009	Transport Layer Security	The applications at https://ba-apps.prepdev.ripe.net and http://core-1.rpki.prepdev.ripe.net:8080 work over HTTP, which is an unencrypted medium.	Elevated
RIPE-006	Insecure configuration	The endpoint https://my.prepdev.ripe.net/api/file-attachment allows the uploading of all files, irrespective of file type. Also, the endpoint does not implement any anti-automation.	Elevated
RIPE-003	Open Redirect	The endpoint https://access.prepdev.ripe.net/?originalUrl= allows redirection to any ripe.net subdomain.	Elevated
RIPE-008	CSRF	The application at https://lirportal.prepdev.ripe.net/ does not implement anti-CSRF controls on some of its endpoints.	Moderate
RIPE-005	Authentication	A password reset link sent to a user's email address is not invalidated if the user's registered email is changed afterwards.	Moderate
RIPE-004	Improper Input Validation	The application uses the user-submitted value of the JSON parameter and copies it to the HTML document as plain text between tags.	Moderate
RIPE-001	Insecure configuration	The HTML5 cross-origin resource sharing (CORS) policy for the request to endpoint https://access.prepdev.ripe.net/user/profile allows access from any domain.	Moderate
RIPE-007	Session management	The https://lirportal.prepdev.ripe.net/training/ endpoint requires the JSESSIONID cookie only, which does not expire on user logout. Moreover, the cookie is missing its secure flag.	Low

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



1.7 Summary of Recommendations

ID	Type	Recommendation
RIPE-009	Transport Layer Security	Applications should use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server. Set the Strict-Transport-Security HTTP header to ensure that clients will refuse to access the server over an insecure connection.
RIPE-008	CSRF	The most robust way to defend against CSRF attacks is to include a CSRF token in the relevant requests. The token should be: <ul style="list-style-type: none"> • Unpredictable with high entropy, as for session tokens in general. • Tied to the user's session. • Strictly validated in every case before the relevant action is executed. • Implemented and validated for all applicable requests.
RIPE-007	Session management	The secure flag should be set on all cookies that are used for transmitting sensitive data when accessing content over HTTPS. If cookies are used to transmit session tokens, then areas of the application that are accessed over HTTPS should employ their own session handling mechanism, and the session tokens in use should never be transmitted over unencrypted communications. Additionally, all of a user's session tokens should be set to expire as soon as they log out of the application.
RIPE-006	Insecure configuration	Implement the recommendations detailed in the full finding.
RIPE-005	Authentication	Invalidate all password reset links when the victim updates their registered email address.
RIPE-004	Improper Input Validation	An application that takes user-controllable data and copies it into application responses opens up the possibility for cross-site scripting attacks. This type of issue can be prevented by: <ul style="list-style-type: none"> • making sure that input is validated as strictly as possible on arrival given the kind of content that it is expected to contain. • HTML-encoding user input whenever it is copied into an application response. For more details please refer to: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
RIPE-003	Open Redirect	Applications should avoid incorporating user-controllable data into redirection targets, or should only allow domains from a whitelist.
RIPE-001	Insecure configuration	Specify trusted origins instead of allowing all domains.

2 Methodology

2.1 Planning

Our general approach during penetration tests is as follows:

1. **Reconnaissance**

We attempt to gather as much information as possible about the target. Reconnaissance can take two forms: active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection afforded to the app or network. This usually involves trying to discover publicly available information by visiting websites, newsgroups, etc. An active form would be more intrusive, could possibly show up in audit logs and might take the form of a social engineering type of attack.

2. **Enumeration**

We use various fingerprinting tools to determine what hosts are visible on the target network and, more importantly, try to ascertain what services and operating systems they are running. Visible services are researched further to tailor subsequent tests to match.

3. **Scanning**

Vulnerability scanners are used to scan all discovered hosts for known vulnerabilities or weaknesses. The results are analyzed to determine if there are any vulnerabilities that could be exploited to gain access or enhance privileges to target hosts.

4. **Obtaining Access**

We use the results of the scans to assist in attempting to obtain access to target systems and services, or to escalate privileges where access has been obtained (either legitimately through provided credentials, or via vulnerabilities). This may be done surreptitiously (for example to try to evade intrusion detection systems or rate limits) or by more aggressive brute-force methods. This step also consist of manually testing the application against the latest (2017) list of OWASP Top 10 risks. The discovered vulnerabilities from scanning and manual testing are moreover used to further elevate access on the application.

2.2 Risk Classification

Throughout the report, vulnerabilities or risks are labeled and categorized according to the Penetration Testing Execution Standard (PTES). For more information, see: <http://www.pentest-standard.org/index.php/Reporting>

These categories are:

- **Extreme**

Extreme risk of security controls being compromised with the possibility of catastrophic financial/reputational losses occurring as a result.

- **High**
High risk of security controls being compromised with the potential for significant financial/reputational losses occurring as a result.
- **Elevated**
Elevated risk of security controls being compromised with the potential for material financial/reputational losses occurring as a result.
- **Moderate**
Moderate risk of security controls being compromised with the potential for limited financial/reputational losses occurring as a result.
- **Low**
Low risk of security controls being compromised with measurable negative impacts as a result.

3 Reconnaissance and Fingerprinting

We were able to gain information about the software and infrastructure through the following automated scans. Any relevant scan output will be referred to in the findings.

- nmap – <http://nmap.org>
- Burp Suite Pro – <https://portswigger.net/burp/pro>
- Naabu – <https://github.com/projectdiscovery/naabu>
- Nuclei – <https://github.com/projectdiscovery/nuclei>

4 Findings

We have identified the following issues:

4.1 RIPE-009 — Unencrypted Communication

Vulnerability ID: RIPE-009

Vulnerability type: Transport Layer Security

Threat level: Elevated

Description:

The applications at <https://ba-apps.prepdev.ripe.net> and <http://core-1.rpki.prepdev.ripe.net:8080> work over HTTP, which is an unencrypted medium.

Technical description:

The application(s) allow users to connect over an unencrypted connection, i.e. HTTP. An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the application and obtain any information the user supplies.

The admin password is also submitted over HTTP, and is therefore also susceptible to theft.

```

Request
Pretty Raw Hex \n
1 POST /certification/?wicket:interface=:3:signInForm::IFormSubmitListener:: HTTP/1.1
2 Host: core-1.rpki.prepdev.ripe.net:8080
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:89.0) Gecko/20100101 Firefox/89.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 53
9 Origin: http://core-1.rpki.prepdev.ripe.net:8080
10 Connection: close
11 Referer: http://core-1.rpki.prepdev.ripe.net:8080/certification/?wicket:bookmarkablePage=:net.ripe.rpki.ui.admin.AdminLoginPage
12 Cookie: JSESSIONID=6545FF32F81428F28E500CF781033DA3; cookies-accepted=accepted; crowd.ripe.hint=true
13 Upgrade-Insecure-Requests: 1
14
15 signInForm2_hf_0=&password=B

Response
Pretty Raw Hex Render \n
1 HTTP/1.1 302
2 X-Content-Type-Options: nosniff
3 X-XSS-Protection: 1; mode=block
4 Cache-Control: no-cache, no-store, max-age=0, must-revalidate
5 Pragma: no-cache
6 Expires: 0
7 X-Frame-Options: DENY
8 Location: http://core-1.rpki.prepdev.ripe.net:8080/certification/SystemStatusPage
9 Content-Length: 0
10 Date: Thu, 22 Jul 2021 22:10:39 GMT
11 Connection: close
12
13

```

Impact:

To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. The attacker would be able to then capture all traffic in clear text, including the admin password.

Recommendation:

Applications should use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server. Set the Strict-Transport-Security HTTP header to ensure that clients will refuse to access the server over an insecure connection.

4.2 RIPE-008 — Cross Site Request Forgery

Vulnerability ID: RIPE-008

Vulnerability type: CSRF

Threat level: Moderate

Description:

The application at <https://lirportal.prepdev.ripe.net/> does not implement anti-CSRF controls on some of its endpoints.

Technical description:

It is possible to perform a CSRF attack on some of the endpoints through a POST request.

Create API Request

The screenshot shows a web browser's developer tools interface. The 'Request' tab is active, displaying a POST request to `/api/managementController/generateKey/Creating%20One/rpki` on the target `https://lirportal.prepdev.ripe.net`. The request headers include `Host: lirportal.prepdev.ripe.net`, `Cookie: cookies-accepted=accepted; activeMembershipId=21671; activeRegId=nl.amsix; JSESSIONID=ba-apps-3326ih3y0m6i11le8zm5rij0gk.ba-apps-3;`, `crowd.token_key=0rXNv_npyEZ3nBrdxNIoyQAAAAAAAAIABMG0YWMwZGVyLnVzZXJAZ21haWwVY29t;`, and `crowd.ripe.hint=true`. The request body is empty. The 'Response' tab is also active, showing a 200 OK response with headers including `Date: Thu, 22 Jul 2021 21:03:47 GMT`, `Server: Apache`, `Content-Security-Policy: frame-ancestors 'none';`, `X-Frame-Options: DENY`, `X-XSS-Protection: 1; mode=block`, `X-Content-Type-Options: nosniff`, `Strict-Transport-Security: max-age=15768000`, and `Content-Type: application/json`. The response body is a JSON object:

```

{
  "record": {
    "apiId": "rpki",
    "name": "Creating One",
    "value": "7532360415fa18e55ecf8ae12b2fdca7866380ff9eaf3f2aa62994283c911",
    "creationDate": "22 July 2021 at 11:03 hrs UTC",
    "creationFullName": "radopsec-1-test radopsec-1",
    "creationEmail": "0ctac0der.user@gmail.com"
  },
  "plaintext": "0c205d35-229c-439c-9882-dbc40fe3303b"
}

```

As shown in the above screenshot, the request does not contain any Anti-CSRF token, which makes it feasible to create a simple malicious HTML form and trick users into submitting it.

CSRF form for API key creation

```

...
<html>
<body>
<script>history.pushState("", "");</script>
<form action="https://lirportal.prepdev.ripe.net/api/managementController/generateKey/Creating%20One/rpki" method="POST">
<input type="submit" value="Submit request" />
</form>
</body>
</html>
...

```

Impact:

This vulnerability could allow an attacker to induce users to perform actions that they do not intend to perform, such as creating API keys. It also allows an attacker to partly circumvent the same-origin policy, which is designed to prevent different websites from interfering with each other.

Recommendation:

The most robust way to defend against CSRF attacks is to include a CSRF token in the relevant requests. The token should be:

- Unpredictable with high entropy, as for session tokens in general.
- Tied to the user's session.
- Strictly validated in every case before the relevant action is executed.
- Implemented and validated for all applicable requests.

4.3 RIPE-007 — Improper session management

Vulnerability ID: RIPE-007

Vulnerability type: Session management

Threat level: Low

Description:

The `https://lirportal.prepdev.ripe.net/training/` endpoint requires the `JSESSIONID` cookie only, which does not expire on user logout. Moreover, the cookie is missing its `secure` flag.

Technical description:

When a user visits the endpoint `https://lirportal.prepdev.ripe.net/training/`, a new cookie with the name `JSESSIONID` is set, but its `secure` flag is missing.

When the user logs out of the application, `JSESSIONID` does not expire, and it remains possible to get authenticated responses from the endpoint:

```

Request
1 GET /training/ HTTP/1.1
2 Host: lirportal.prepdev.ripe.net
3 Cookie: JSESSIONID=ba-apps-3326ih3y0m6i11le8zm5rij0gk.ba-apps-3
4 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:89.0)
5 Gecko/20100101 Firefox/89.0
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
7 Accept-Language: en-US,en;q=0.5
8 Accept-Encoding: gzip, deflate
9 Referer: https://lirportal.prepdev.ripe.net/api/
10 Upgrade-Insecure-Requests: 1
11 Te: trailers
12 Connection: close
13

Response
1 HTTP/1.1 200 OK
2 Date: Thu, 22 Jul 2021 18:39:05 GMT
3 Server: Apache
4 Content-Security-Policy: frame-ancestors 'none';
5 X-Frame-Options: DENY
6 X-XSS-Protection: 1; mode=block
7 X-Content-Type-Options: nosniff
8 Strict-Transport-Security: max-age=15768000
9 Content-Type: text/html;charset=UTF-8
10 Content-Language: en
11 Pragma: no-cache
12 Cache-Control: no-cache, max-age=0, must-revalidate
13 Vary: Accept-Encoding,User-Agent
14 Connection: close
15 Content-Length: 27789
16
17 <!DOCTYPE html PUBLIC
18 "-//W3C//DTD XHTML 1.0 Transitional//EN"
19 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
20
21 <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
22
23 <head>
24 <!-- Matomo Tag Manager -->
25 <script type="text/javascript">
26   var _mtm = _mtm || [];
27   _mtm.push({
28     'mtm.startTime': (new Date().getTime()), 'event': 'mtm.Start'
29   });
30   var d=document, g=d.createElement('script'), s=d.getElementsByTagName(
31     g.type='text/javascript';
32     g.async=true;
33     g.defer=true;
34     g.src='https://www-analytics.ripe.net/js/container_6v8dyJ8K.js';
35     s.parentNode.insertBefore(g,s);
36   </script>
37 <!-- End Matomo Tag Manager -->
38 <meta charset="UTF-8">
39
40 <link href="https://www-static.ripe.net/static/common/font-awesome

```

Impact:

Because the cookie lacks a `secure` flag, it is susceptible to be leaked during a MitM attack. More in general, non-expiring cookies of this type result in an increased attack surface. An attacker with access to valid cookies from previous sessions can access information the endpoint that should be restricted to legitimate and authenticated users.

Recommendation:

The `secure` flag should be set on all cookies that are used for transmitting sensitive data when accessing content over HTTPS. If cookies are used to transmit session tokens, then areas of the application that are accessed over HTTPS should employ their own session handling mechanism, and the session tokens in use should never be transmitted over unencrypted communications.

Additionally, all of a user's session tokens should be set to expire as soon as they log out of the application.

4.4 RIPE-006 — Unrestricted File Upload

Vulnerability ID: RIPE-006

Vulnerability type: Insecure configuration

Threat level: Elevated

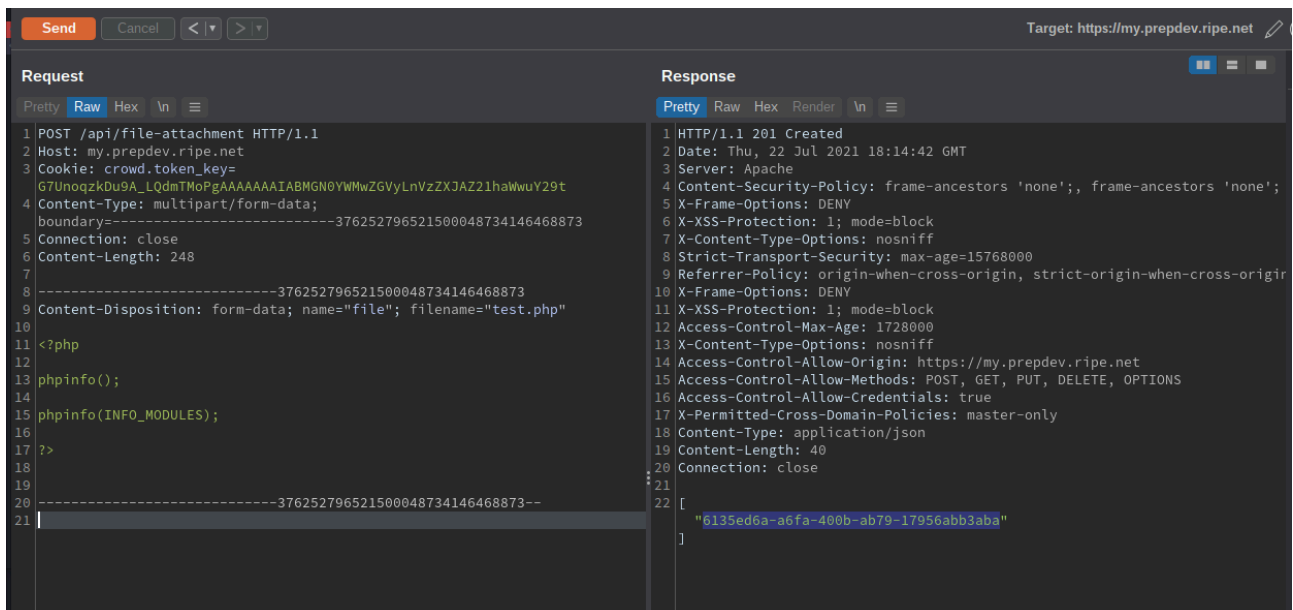
Description:

The endpoint `https://my.prepdev.ripe.net/api/file-attachment` allows the uploading of all files, irrespective of file type. Also, the endpoint does not implement any anti-automation.

Technical description:

The endpoint `https://my.prepdev.ripe.net/api/file-attachment` lets users upload files of any type, including `php`, `html`, etc.

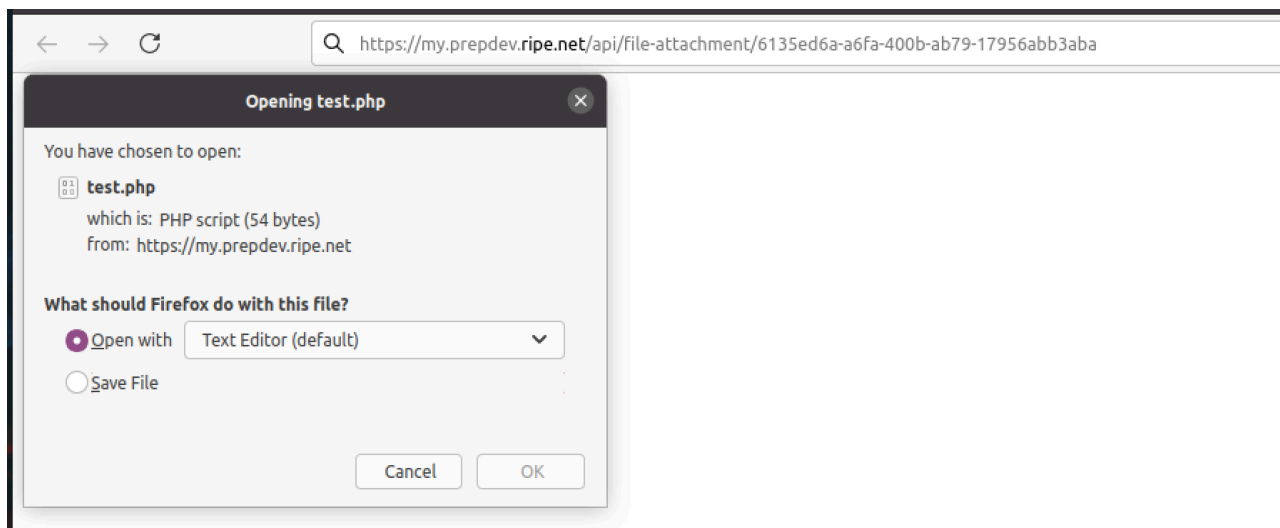
File Upload



```
Send Cancel < > Target: https://my.prepdev.ripe.net

Request
Pretty Raw Hex \n
1 POST /api/file-attachment HTTP/1.1
2 Host: my.prepdev.ripe.net
3 Cookie: crowd.token_key=
  G7UnqzkDu9A_LQdmTMoPgAAAAAATABMGNOYWMwZGVyLnVzZXJAZ21haWwUy29t
4 Content-Type: multipart/form-data;
  boundary=-----376252796521500048734146468873
5 Connection: close
6 Content-Length: 248
7
8 -----376252796521500048734146468873
9 Content-Disposition: form-data; name="file"; filename="test.php"
10
11 <?php
12
13 phpinfo();
14
15 phpinfo(INFO_MODULES);
16
17 ?>
18
19
20 -----376252796521500048734146468873--
21

Response
Pretty Raw Hex Render \n
1 HTTP/1.1 201 Created
2 Date: Thu, 22 Jul 2021 18:14:42 GMT
3 Server: Apache
4 Content-Security-Policy: frame-ancestors 'none';, frame-ancestors 'none';
5 X-Frame-Options: DENY
6 X-XSS-Protection: 1; mode=block
7 X-Content-Type-Options: nosniff
8 Strict-Transport-Security: max-age=15768000
9 Referrer-Policy: origin-when-cross-origin, strict-origin-when-cross-origin
10 X-Frame-Options: DENY
11 X-XSS-Protection: 1; mode=block
12 Access-Control-Max-Age: 1728000
13 X-Content-Type-Options: nosniff
14 Access-Control-Allow-Origin: https://my.prepdev.ripe.net
15 Access-Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS
16 Access-Control-Allow-Credentials: true
17 X-Permitted-Cross-Domain-Policies: master-only
18 Content-Type: application/json
19 Content-Length: 40
20 Connection: close
21
22 [
  "6135ed6a-a6fa-400b-ab79-17956abb3aba"
]
```

Because the endpoint is additionally lacking anti-automation, is also possible to automate the file upload request and upload numerous files through a script.

Impact:

Malicious users could upload any number of malicious files of any type to the application server. While the uploaded files are not executed on the server, the server may still be used to distribute malicious files, e.g. files containing malware.

Recommendation:

It is strongly recommended to verify both the content and type of a file before allowing it to be uploaded. These checks should always happen on the server side. Some important recommendations as per OWASP (https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload):

- Restrict allowed file types to those that are strictly necessary for business.
- Never accept a filename and extension directly; always apply an allowlist filter.
- The application should perform filtering and content checking on all files that are uploaded to the server. Files should be thoroughly scanned and validated before being made available to other users. If in doubt, the file should be discarded.
- The web application should provide a list of permitted extensions for a file type, and only allow users to select extensions from this list.
- All control and Unicode characters should be removed from filenames and extensions. Special characters such as `;`, `:`, `>`, `<`, `/`, `.`, `*`, and so on should be discarded as well. If there is no special need to have unicode characters, it is highly recommended to only allow alpha-numeric characters and a single dot as in the file name. Likewise, do not allow empty file names or extensions (e.g. regular expression `[a-zA-Z0-9]{1,200}` . `[a-zA-Z0-9]{1,10}`).

- Limit the filename length. For instance, the maximum length of the name of a file plus its extension should be less than 255 characters (not counting parent directories) on an NTFS partition.
- It is recommended to not let users name a file at all and instead use an algorithm; e.g. an MD5 hash of the name of file plus the current date.
- The directory to which the files are uploaded should not have any “execute” permission and all script handlers should be removed from these directories.
- Limit the file size to a maximum value in order to prevent denial of service attacks (on file space or other web application functions such as the image resizer).
- Also consider setting a minimum file size, again to prevent denial of service attacks.

Finally, it is also important to implement anti-automation on this endpoint, to prevent malicious actors flooding the server with numerous malicious uploads.

4.5 RIPE-005 — Password Reset Token Does Not Expire on Email Change

Vulnerability ID: RIPE-005

Vulnerability type: Authentication

Threat level: Moderate

Description:

A password reset link sent to a user's email address is not invalidated if the user's registered email is changed afterwards.

Technical description:

The application allows users to change their password using a password reset link sent to an email address that is no longer their registered address.

Steps to reproduce:

- User A is registered with the email `user@test.com` in the RIPE application.
- Noticing unauthorised access to their email inbox, User A changes the email linked to their account to `user_new@test.com`
- Before User A changed their registered email address, however, the attacker has already requested a password reset link to be sent to the hacked email address.

- The password reset link sent to older email address `user@test.com` is still valid and will allow the attacker to reset the account password. (Although do note that the attacker would still need to find the victim's new email address).

This opens a small attack surface for someone's RIPE application account to get compromised, even if they detect the compromise of their inbox and update their email address.

Impact:

The vulnerability allows an attacker with access to an unused password reset link sent to an older email to reset the victim's password even when the email linked to that account has been changed.

Recommendation:

Invalidate all password reset links when the victim updates their registered email address.

4.6 RIPE-004 — Improper input validation

Vulnerability ID: RIPE-004

Vulnerability type: Improper Input Validation

Threat level: Moderate

Description:

The application uses the user-submitted value of the JSON parameter and copies it to the HTML document as plain text between tags.

Technical description:

It is possible to inject an XSS payload into the JSON parameters `asn` and `prefix` in the POST request sent to `https://my.prepdev.ripe.net/api/rpki/alerts/suppress`. The user input is copied from a request and echoed into the application's immediate response.

Request & Response

```
Request
1 POST /api/rpki/roas/stage HTTP/1.1
2 Host: my.prepdev.ripe.net
3 Cookie: cookies-accepted=accepted; crowd.token_key=pa5kUAE91jU8kwfusCCP
4 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:89.0) Gecko/20100
5 Accept: application/json, text/plain, */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://my.prepdev.ripe.net/
9 Content-Type: application/json;charset=utf-8
10 Content-Length: 1324
11 Origin: https://my.prepdev.ripe.net
12 Te: trailers
13 Connection: close
14
15 [
  {
    "asn": "AS0xbk2d<script>alert(1)</script>upv6k",
    "prefix": "2001:7f8:86::/48",
    "maxLength": 48,
    "_numberOfValidsCaused": 0,
    "_numberOfInvalidsCaused": 0
  },
  {
    "asn": "AS1200",
    "prefix": "80.249.208.0/21",
    "maxLength": 21,
    "_numberOfValidsCaused": 0,
    "_numberOfInvalidsCaused": 0
  },
  {
    "asn": "AS1200",
    "prefix": "91.200.16.0/22",
    "maxLength": 22,
    "_numberOfValidsCaused": 1,
    "_numberOfInvalidsCaused": 0
  },
  {
    "asn": "AS1200",
    "prefix": "91.236.189.0/24",
    "maxLength": 24,
    "_numberOfValidsCaused": 0,
    "_numberOfInvalidsCaused": 0
  }
],
]

Response
1 HTTP/1.1 400 Bad Request
2 Date: Thu, 22 Jul 2021 17:04:22 GMT
3 Server: Apache
4 Content-Security-Policy: frame-ancestors 'none'; frame-ancestors 'none';
5 X-Frame-Options: DENY
6 X-XSS-Protection: 1; mode=block
7 X-Content-Type-Options: nosniff
8 Strict-Transport-Security: max-age=15768000
9 Vary: Accept-Encoding
10 Referrer-Policy: origin-when-cross-origin,
strict-origin-when-cross-origin
11 X-Frame-Options: DENY
12 X-XSS-Protection: 1; mode=block
13 Access-Control-Max-Age: 1728000
14 X-Content-Type-Options: nosniff
15 Access-Control-Allow-Origin: https://my.prepdev.ripe.net
16 Access-Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS
17 Access-Control-Allow-Credentials: true
18 X-Permitted-Cross-Domain-Policies: master-only
19 Content-Type: text/plain; charset=UTF-8
20 Content-Length: 193
21 Connection: close
22
23 {"error": "New ROAs are not correct: ASN
'AS0xbk2d<script>alert(1)</script>upv6k' is invalid in
(ROA{asn='AS0xbk2d<script>alert(1)</script>upv6k',
prefix='2001:7f8:86::/48', maxLength=48})"}
```

Affected URL: <https://my.prepdev.ripe.net/api/rpki/roas/stage>

Affected Parameters: `asn` and `prefix`

Impact:

This issue demonstrates that it is possible to inject arbitrary JavaScript into the application's response. Although the request uses a Content-type header (`Content-Type: text/plain; charset=UTF-8`), and therefore the issue is only directly exploitable (as XSS) if a browser can be made to interpret the response as HTML, the issue might be indirectly exploitable if a client-side script processes the response and embeds it into an HTML context. This finding however only shows the improper input validation in the application.

Recommendation:

An application that takes user-controllable data and copies it into application responses opens up the possibility for cross-site scripting attacks. This type of issue can be prevented by:

- making sure that input is validated as strictly as possible on arrival given the kind of content that it is expected to contain.
- HTML-encoding user input whenever it is copied into an application response.

For more details please refer to: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

4.7 RIPE-003 — Open Redirect to Subdomains

Vulnerability ID: RIPE-003

Vulnerability type: Open Redirect

Threat level: Elevated

Description:

The endpoint `https://access.prepdev.ripe.net/?originalUrl=` allows redirection to any `ripe.net` subdomain.

Technical description:

The application incorporates user-controllable data into the redirection url in an unsafe manner. An attacker can construct a URL like `https://access.prepdev.ripe.net/?originalUrl=https://attacker.domain.ripe.net`, which would cause the user to be redirected to `attacker.domain.ripe.net`.

```

Request
-----
1 GET /?originalUrl=https://attacker.domain.ripe.net HTTP/1.1
2 Host: access.prepdev.ripe.net
3 Cookie: JSESSIONID=ba-apps-4efkv2r57rzmpia6yq1mq303313488.ba-apps-4; cookies-accepted=
  accepted; activeMembershipId=21671; activeRegId=nl.amsix; crowd.token_key=
  zdVyalW9VJaawtT51ZDLwAAAAAIABYWJoaW5hd1pc2hyVUByYWRpY2FsbHvcGVuc2VjdXJpdHkuY29t;
  crowd.ripe.hint=true
4 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:89.0) Gecko/20100101
  Firefox/89.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Upgrade-Insecure-Requests: 1
9 Te: trailers
10 Connection: close
11
12

Response
-----
1 HTTP/1.1 302 Found
2 Date: Mon, 12 Jul 2021 15:59:05 GMT
3 Server: Apache
4 Content-Security-Policy: frame-ancestors *.ripe.net;
5 X-Frame-Options: DENY
6 X-XSS-Protection: 1; mode=block
7 X-Content-Type-Options: nosniff
8 Strict-Transport-Security: max-age=15768000
9 Location: https://attacker.domain.ripe.net
10 Content-Type: text/html;charset=utf-8
11 Content-Length: 0
12 Set-Cookie: cookies-accepted=accepted; Domain=.ripe.net; Path=/
13 Connection: close
14
15

```

Note

A malicious user with access to RIPE Atlas probes can create a DNS record of a subdomain pointing to a host of their choice. This allows them to create a subdomain and control the content served on that domain.

Impact:

An attacker may target the users of the Ripe.net application in a phishing attack using the legitimate `https://access.prepdev.ripe.net/?originalUrl=` URL. The fact that the attack would be making use of an authentic application URL, target the correct domain and use a valid SSL certificate (if SSL is used) would lend credibility to the phishing attack because many users, even those actively verifying authenticity, would not notice the subsequent redirection to a different domain.

Recommendation:

Applications should avoid incorporating user-controllable data into redirection targets, or should only allow domains from a whitelist.

4.8 RIPE-001 — Cross-origin Resource Sharing Policy Misconfiguration

Vulnerability ID: RIPE-001

Vulnerability type: Insecure configuration

Threat level: Moderate

Description:

The HTML5 cross-origin resource sharing (CORS) policy for the request to endpoint `https://access.prepdev.ripe.net/user/profile` allows access from any domain.

Technical description:

An HTML5 cross-origin resource sharing (CORS) policy controls whether and how content running on other domains can perform two-way interaction with the domain that publishes the policy. Trusting arbitrary origins effectively disables the same-origin policy, allowing two-way interaction by third-party web sites.

Accepting random domains

The screenshot displays the Network tab of a browser's developer tools. The target is `https://access.prepdev.ripe.net`. The request is a GET to `/user/profile HTTP/1.1`. The response is an HTTP/1.1 200 OK from Apache. The response headers include `Access-Control-Allow-Credentials: true` and `Access-Control-Allow-Origin: https://my.prepdev.ripe.net.attacker.com`. The response body is a JSON object containing user profile information.

```

Request
1 GET /user/profile HTTP/1.1
2 Host: access.prepdev.ripe.net
3 Cookie: 3SESSIONID=ba-apps-4yf2s5qj5b7c85uy2egnu4ldb151240.ba-apps-4; crowd.token_key=
4 Origin: https://my.prepdev.ripe.net.attacker.com
5
6
7

Response
1 HTTP/1.1 200 OK
2 Date: Mon, 05 Jul 2021 19:46:19 GMT
3 Server: Apache
4 Content-Security-Policy: frame-ancestors *.ripe.net;
5 X-Frame-Options: DENY
6 X-XSS-Protection: 1; mode=block
7 X-Content-Type-Options: nosniff
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 325
10 Connection: close
11
12 Access-Control-Allow-Credentials: true
13 Vary: Origin
14 Access-Control-Allow-Origin: https://my.prepdev.ripe.net.attacker.com
15
16 {
  "login": "abhinavmishra@radicallyopensecurity.com",
  "firstName": "radopsec-1-test",
  "lastName": "radopsec-1",
  "active": true,
  "twoFactorAuthEnabled": false,
  "uuid": "12821d5c-1531-460f-b00d-8d848f329ca0",
  "registries": [
    {
      "membershipId": 21671,
      "role": "ADMIN",
      "organisationName": "Alpha India Echo Bravo",
      "regId": "nl.amsix",
      "primary": true
    }
  ]
}

```

Impact:

An attacker with the ability to execute JavaScript in the context of on an external domain could try to access application-specific data and exfiltrate it.

Recommendation:

Specify trusted origins instead of allowing all domains.

5 Non-Findings

In this section we list some of the things that were tried but turned out to be dead ends.

5.1 NF-002 — Non Finding Test Cases

During the penetration test, we perform the following tests which did not result in any vulnerability.

- Verify if it is possible to bypass the authentication
- Verify if any of the endpoints used in the application, is missing access control
- Verify all pages and resources by default require authentication except those specifically intended to be public (Principle of complete mediation).
- Verify that all password fields do not echo the user's password when it is entered.
- Verify all authentication controls are enforced on the server side.
- Verify all authentication controls fail securely to ensure attackers cannot log in.
- Verify there are no default passwords in use for the application framework or any components used by the application (such as "admin/password").
- Verify that sessions are invalidated when the user logs out.
- Verify that the session id is never disclosed in URLs, error messages, or logs. This includes verifying that the application does not support URL rewriting of session cookies.
- Verify that all successful authentication and re-authentication generates a new session and session id.
- Verify that session ids are sufficiently long, random and unique across the correct active session base.
- Verify that session ids stored in cookies have their path set to an appropriately restrictive value for the application, and authentication session tokens additionally set the 'HttpOnly' and 'secure' attributes.
- Verify that access to sensitive records is protected, such that only authorized objects or data is accessible to each user (for example, protect against users tampering with a parameter to see or alter another user's account).
- Verify that directory browsing/indexing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.
- Verify that the application is not susceptible to OS Command Injection, or that security controls prevent OS Command Injection.
- Verify that the application is not susceptible to Remote File Inclusion (RFI) or Local File Inclusion (LFI) when content is used that is a path to a file.
- Verify that the application is not susceptible to common XML attacks, such as XPath query tampering, XML External Entity attacks, and XML injection attacks.
- Verify that all forms containing sensitive information have disabled client side caching, including autocomplete features.
- Verify that all sensitive data is sent to the server in the HTTP message body or headers (i.e., URL parameters are never used to send sensitive data).

- Verify the application code does not execute uploaded data obtained from untrusted sources.
- Verify the use of session-based authentication and authorization.
- Verify that the application is protected against cross site scripting, SQL injection, Insecure direct object reference etc.

6 Future Work

- **Retest of findings**

When mitigations for the vulnerabilities described in this report have been deployed, a repeat test should be performed to ensure that they are effective and have not introduced other security problems.

- **Regular security assessments**

Security is an ongoing process and not a product, so we advise undertaking regular security assessments and penetration tests, ideally prior to every major release or every quarter.

7 Conclusion

3 Elevated, 4 Moderate and 1 Low-severity issue were found during the penetration test. The implementation of access control and authorisation was found to be strong throughout the application and no vulnerability was discovered in those aspects. The security issues we did encounter are related to unencrypted communication, unrestricted file upload, open redirect, input validation, and session management. The end-user security could, and should, be improved by fixing all of the discovered weaknesses. Although not critical, fixing the low and moderate-rated issues will still help to reduce the applications' attack surface.

We therefore recommend fixing all of the issues found and then performing a retest in order to ensure that mitigations are effective and that no new vulnerabilities have been introduced.

Finally, we want to emphasize that security is a process – this penetration test is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end.

Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

Appendix 1 Testing team

Abhinav Mishra	Abhinav has 10+ years of experience in the penetration testing of web, mobile, infrastructure, API and other fields. He has received numerous accolades from multiple organizations for responsible disclosure of vulnerabilities. He is also known for providing trainings on web, mobile and infrastructure security.
Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.

Front page image by Slava (<https://secure.flickr.com/photos/slava/496607907/>), "Mango HaX0ring",
Image styling by Patricia Piolon, <https://creativecommons.org/licenses/by-sa/2.0/legalcode>.