Red Bull | INFORMATION TECHNOLOGY

SECURITY ASSESSMENT

2024-001 - Red Bull Demo

05/09/2024
VERSION 1.0

Red Bull GmbH | Global Digital Security Department
Halleiner Landesstrasse 24, 5061 Elsbethen

CONTACT
John Doe
+00 000 0000 000
john.doe@example.com

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

The following chapter summarizes the scope and results of the security assessment. It provides estimated risk and outlines measures recommended by Red Bull Digital Security Department.

1.1 SCOPE

Red Bull Global Digital Security Department conducted a security test between Thursday 5 September 2024 and Thursday 12 September 2024 of the systems listed below.

1.1.1 SYSTEMS

Name	Domain
Demo Service	https://demoservice.redbull.com/

The pentest comprised the web application Demo Service and took place as a black box test. We had access to the version 13.230 which was released on the 1st September 2024. The test was carried out over a period of six person days.

For this test the application was cloned and hosted in a demo environment for us to test.

1.1.2 ROLES & RESPONSIBILITIES

Role	Name	Contact details
Organizational Lead	John Doe	security.officer@redbull.com
Service Owner	Bob Brown	service.owner@redbull.com

1.1.3 PROVIDED USER ACCOUNTS

The following accounts were provided for testing purposes:

- User 1 (Admin)
- User 2 (Normal User)
- User 3 (Guest User)

Please delete or deactivate mentioned accounts once the security test had been completed.

1.1.4 DEVICES USED FOR TESTING

The following ip addresses were used by the Security Assessor to test the application:

- 127.0.0.1

1.2 RESULT SUMMARY

During the 48 hours of testing, the following vulnerabilities were identified:

Vulnerability	Criticality	Remediation Status
Application takeover and database access by SQLi	Critical	Open
Reusable Password Reset Link	Medium	Open
Access to sensitive data from Elastic database	Medium	Open
Publicly accessible API Documentation	Medium	Open
Insecure HTTP cookie settings	Medium	Open

This is the space for the executive summary where you can provide a brief overview of the security assessment. Highlight key findings, vulnerabilities identified, and recommended actions. Ensure to summarize the overall security posture and any critical issues that need immediate attention.

Please be aware that a security test provides a snapshot at the time of the security assessment and does not indicate any state of security in the future. Therefore, it is highly recommended to repeat a security test on a yearly basis or at least after major updates.

1.3 ESTIMATED RISK

Based on evaluating all found vulnerabilities a risk score of "6.9" was estimated whereas a risk score of 0 means no risk and a risk score of 10 equals the highest possible risk. The risk score matches a risk level of "Medium".

Risk Score / Risk Level
6.9 / Medium Risk

The following figure shows the overall risk determined in an impact/likelihood diagram with a high-level view of the application. The likelihood axis describes how skilled an attacker must be, what resources are required, and how easy it is to discover and exploit vulnerabilities. Both values are based on facts, but also include a subjective assessment.

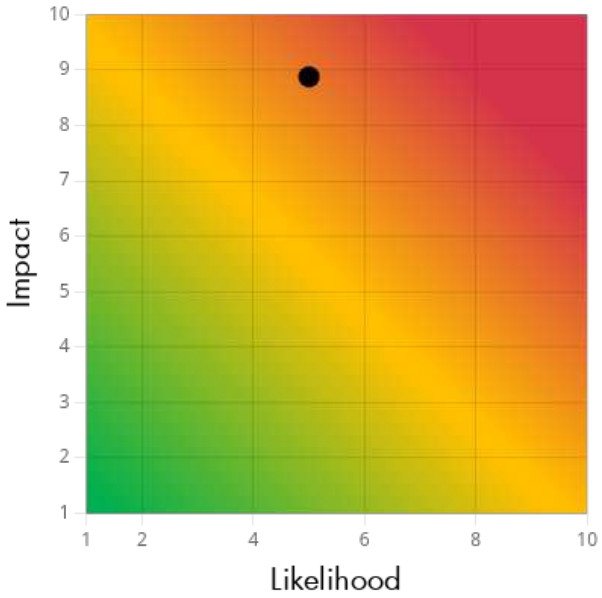

Figure 1 - Overall Risk

1.4 SUGGESTED MEASURES

Findings with severity "Critical" or "High" should be fixed as soon as possible. The decision about implementation of mitigations for all other findings with severity medium and below is up to the service owner and should depend on cost/benefit considerations.

This is the space for the suggested measures section where you can outline the recommended actions to improve security. Focus on practical steps that can be implemented to mitigate identified risks. Prioritize measures based on their impact and feasibility, and provide clear instructions for execution. Ensure to address any critical vulnerabilities and suggest best practices for maintaining a robust security posture.

2 VULNERABILITY DETAILS

2.1 APPLICATION TAKEOVER AND DATABASE ACCESS BY SQLI

10.0 Critical	Root Cause	Remediation Status
	Insecure Design	Open

OVERVIEW

This SQL Injection (SQLi) vulnerability allows attackers to query, modify and delete data in the SQL Database of the application. It would be possible for an attacker to combine this with other attacks and gain access to other systems.

IMPACT

The application did not sanitize the user input from parameter `orderBy`, which was used in an SQL statement.

By testing with the following String it was possible to determine that an SQLi Attack was possible: `'+OR+1=1--`.

RECOMMENDATION

- Use prepared statements or stored procedures wherever possible. Prepared statements are parameterized statements and prevent attackers from manipulating SQL statements.
- Validate all user input. Ensure that only expected and valid input is accepted. Do not sanitize potentially malicious input.
- For detailed information and assistance on how to prevent SQL Injection vulnerabilities, see the OWASP SQL Injection Prevention Cheat Sheet[1].

1. https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

2.2 REUSABLE PASSWORD RESET LINK

5.9 Medium	Root Cause	Remediation Status
	Insecure Design	Open

OVERVIEW

Users that reset their password get a password-reset-link which is still valid after a password reset using this link. This allows an attacker, who gained access to the link via a keylogger, log files or other measures, could change the password again without the user knowing.

IMPACT

We discovered that the password reset links generated by the application are reusable. An attacker with access to a reusable link can repeatedly reset passwords, which might lead to unauthorized account access. This could allow attackers to impersonate users and to perform unauthorized actions.

RECOMMENDATION

- Password-Reset Links should be generated randomly with sufficient entropy, so links cannot be guessed.
- Invalidate Password-Reset forms after usage. When requesting a new Password-Reset link, invalidate all previous links.
- The Password-Reset link should be invalidated after a certain period of time (e.g., after 1 hour).

2.3 ACCESS TO SENSITIVE DATA FROM ELASTIC DATABASE

5.8 Medium	Root Cause	Remediation Status
	Weak Credentials or Credential Management	Open

OVERVIEW

We were able to query customer data from the Elastic database due to a leaked API token in the HTML source.

IMPACT

The web application exposed an Elastic API token in its HTML source.

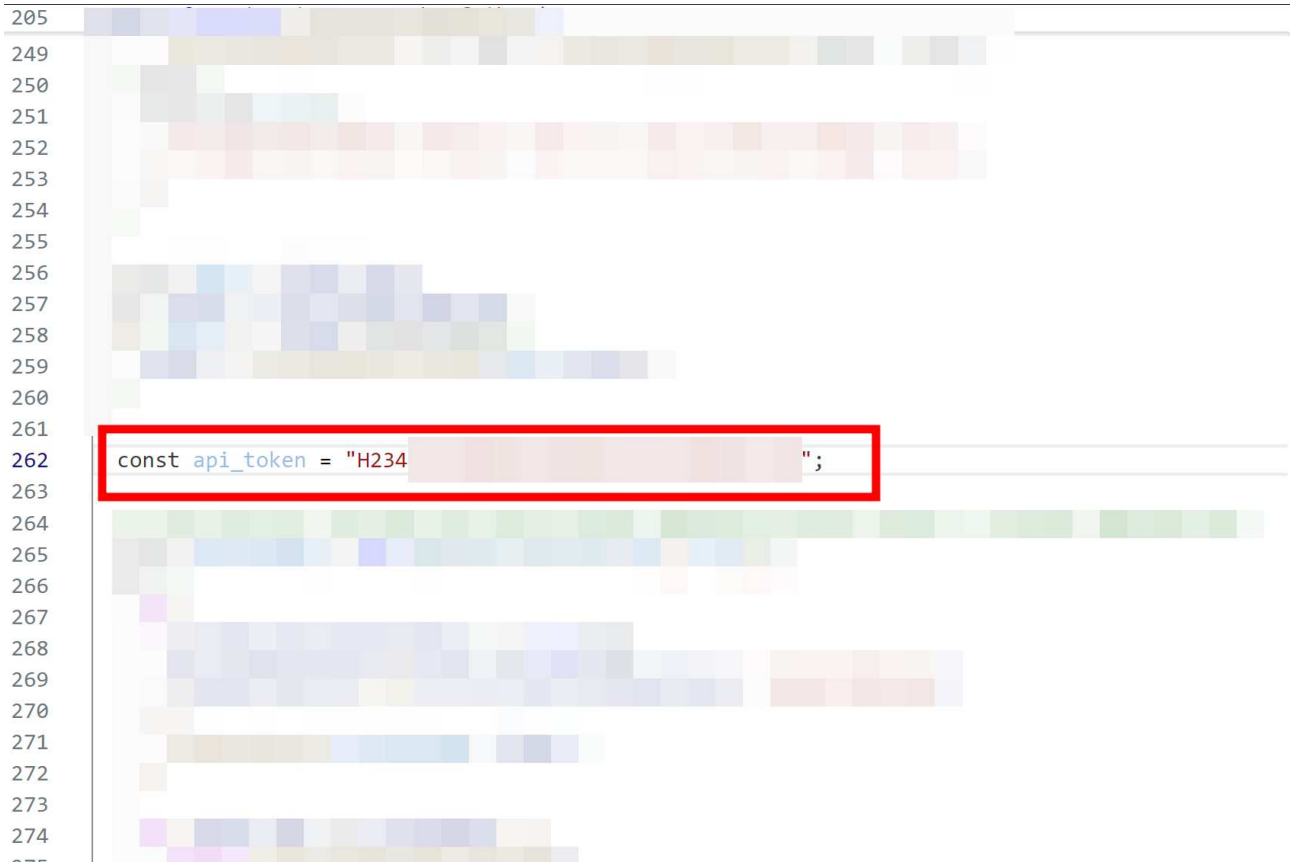

Figure 2 - API Token In Source Code

We were able to use the API token to query customer data from the Elastic database.

RECOMMENDATION

- Remove the API token from the HTML source

2.4 PUBLICLY ACCESSIBLE API DOCUMENTATION

5.3 Medium	Root Cause	Remediation Status
	No Root Cause	Open

OVERVIEW

A public API documentation exposes a comprehensive list of usable API endpoints which might increase the application's visible attack surface.

IMPACT

The backend of the web app has a publicly accessible OpenAPI and Swagger API specification. This could help attackers gain information about sensitive endpoints.

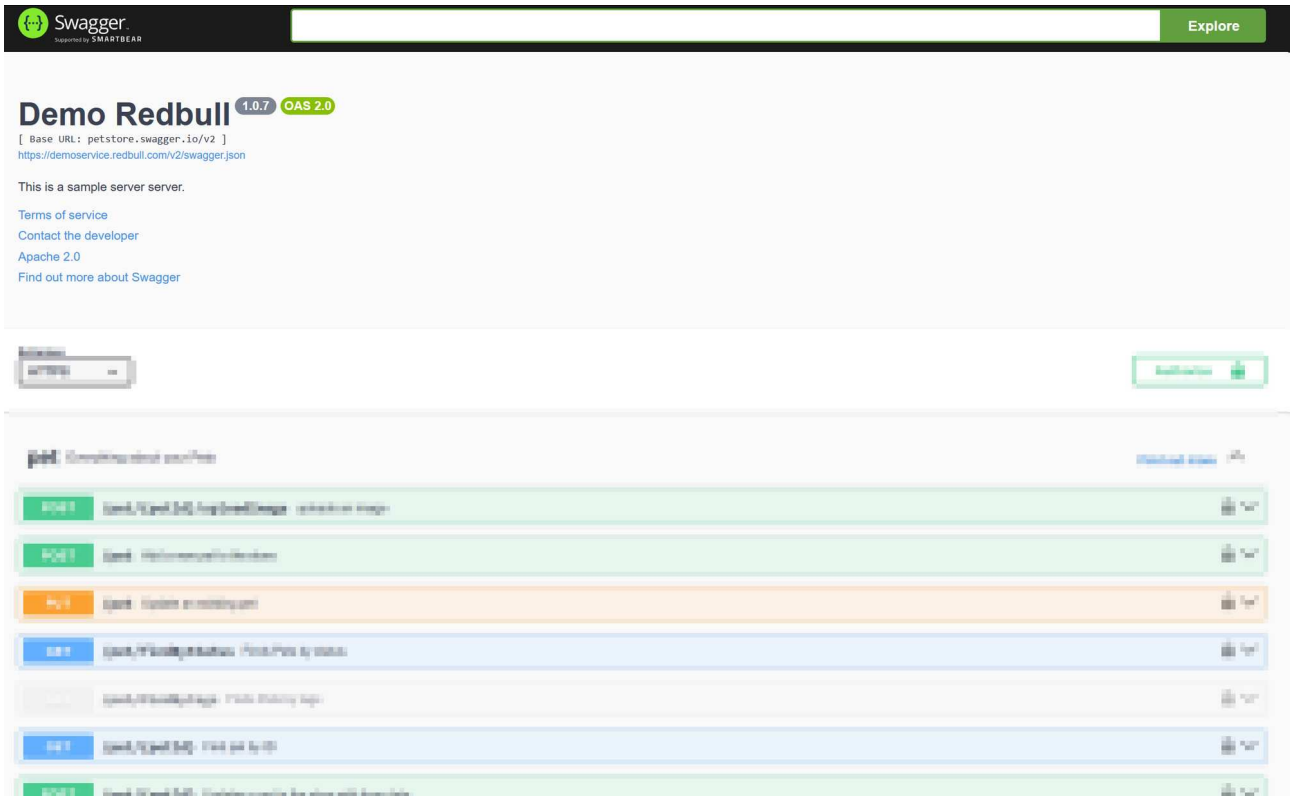

Figure 3 - Public swagger file

RECOMMENDATION

- Evaluate whether to restrict public access to the OpenAPI documentation.

2.5 INSECURE HTTP COOKIE SETTINGS

4.2 Medium	Root Cause	Remediation Status
	Insecure Configuration	Open

OVERVIEW

Cookies without the Secure flag can be intercepted via man-in-the-middle attacks, compromising session confidentiality. Without the HttpOnly flag, cookies are vulnerable to theft through XSS attacks, allowing session hijacking. Improper SameSite settings make the application susceptible to CSRF attacks, leading to unauthorized actions. Persistent cookies without expiry increase the risk of prolonged exposure and unauthorized access.

IMPACT

The session cookies `advertisementId` had the following attributes set:

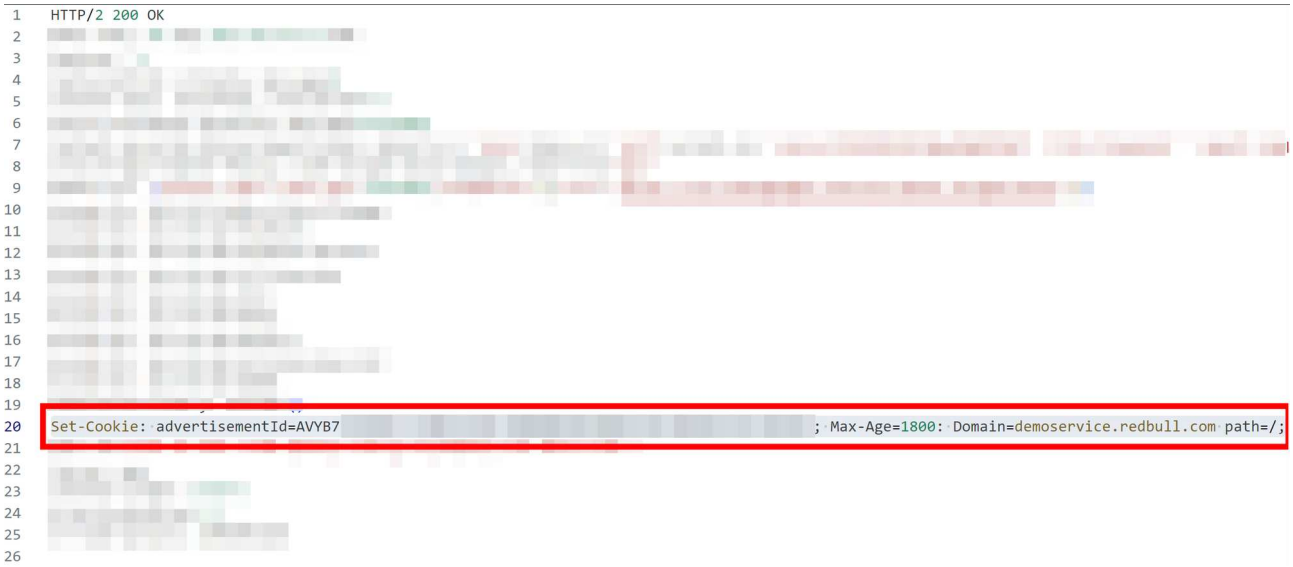

Figure 4 - Set Cookie Header for Advertisement Id

- JavaScript can access the cookies (missing `HttpOnly` flag)
- Browsers might send the cookies over unencrypted connections (missing `Secure` flag)
- Browsers might send the cookies in cross-site requests (`SameSite` setting is not `Strict`)

RECOMMENDATION

✔️ `HttpOnly` (helps mitigate the impact of XSS attacks)
✔️ `Secure` (prevents the browser from sending the cookie over unencrypted connections)
❗ `SameSite=Strict` (helps mitigate the impact of CSRF attacks)

3 PERSISTENT FILES

The following files were created on the server during the penetration test and may contain malicious payloads:

- RBDS2024-001_shell.php (could not be deleted)
- RBDS2024-001_xxe.xml
- RBDS2024-001_revshell.php

The following files were uploaded, but it is not clear under which filename they were written to disk (the names listed here are the ones they were being uploaded under). Therefore, they have not been removed from the system:

- RBDS2024-001_pwn.exe (Windows executable with basic Metasploit Reverse Shell functionaliy on IP address 1.1.1.1)
- RBDS2024-001_test.html (Hello world HTML file with no security impact)

4 METHODOLOGY OVERVIEW

The web security testing methodology by Red Bull Digital Security Department has been derived from OWASP guidelines:

OWASP Top 10

Red Bull Web&App Security Requirements