



SAMPLE REPORT

REDPOINT SECURITY

APPLICATION SECURITY ASSESSMENT

Date xx/xx/2024

rdpt.io

EXECUTIVE SUMMARY

OVERVIEW

Client engaged Redpoint Security (“Redpoint”) to conduct an application security assessment of web and/or mobile applications by analyzing potential security flaws within the application in its running environment, the related backend web service, and associated source code.

The purpose of the assessment is to provide assurance that the application and its infrastructure has a strong security posture and to identify technical vulnerabilities which may expose the business and/or its customers to risk. To assess the security posture of the application, a security assessment was conducted from anonymous and authenticated perspectives, using multiple application roles (authenticated user and administrative user). Redpoint was provided with credentials, access to the application, and access to the source code to evaluate various security controls.

ANALYSIS

Redpoint identified a total of [#] findings during the assessment, ranging from high to low and informational severity. These findings relate to the confidentiality, integrity, and availability of the application, the environment, and the data contained within it.

The issues with critical and high severity risks which Redpoint found during the assessment included an Insecure Direct Object Reference (IDOR) finding as well as the discovery of a component using default credentials and stored cross-site scripting (XSS) vulnerabilities. A medium-risk vulnerability was also found that permitted the ability to enumerate valid usernames through the application responses to locked accounts.

The other low-severity issue was the use of outdated components that have known vulnerabilities. While these outdated libraries may not be used in an insecure manner, inclusion of this software increases the attack surface, invites attacks, and indicates an undisciplined approach to patching software.

The informational finding was a lack of the Content Security Policy (“CSP”) response headers on application pages. CSP headers are secondary controls that can be implemented to help mitigate impact from XSS attacks like HTML, DOM, JSON, and JavaScript injection vulnerabilities.

RATING

Overall, the application rates as insecure in an objective comparison to other applications assessed by Redpoint. Mitigation of the identified IDOR, default credentials, and username enumeration flaws and upgrades to known vulnerable components would strengthen the overall posture of the application. Additional implementation of the aforementioned CSP headers would add additional layers of security to the application.

FINDINGS CHART

ID	Severity	Category	Summary
1	CRITICAL	Broken Access Control	Insecure Direct Object Reference. An attacker can gain unauthorized access through manipulating references.
2	HIGH	Security Misconfiguration	Default Credentials. Application or one of its platform components operates with default credentials.
3	HIGH	Injection	Stored Cross-Site Scripting (XSS). The application vulnerable to Cross-Site Scripting (XSS) attacks because it does not properly output encode data user input before it is returned in the application response, allowing for the injection of HTML and script characters.
4	MEDIUM	Broken Authentication	Account Enumeration. Application usernames could be enumerated in multiple locations.
5	LOW	Security Misconfiguration	Using Components with Known Vulnerabilities. Application used outdated and unpatched components that contain published CVEs.
6	INFORMATIONAL	Security Misconfiguration	Content Security Policy (CSP) Not Implemented. The application server did not include CSP headers that limit page and site interactions with backend services.

FINDINGS

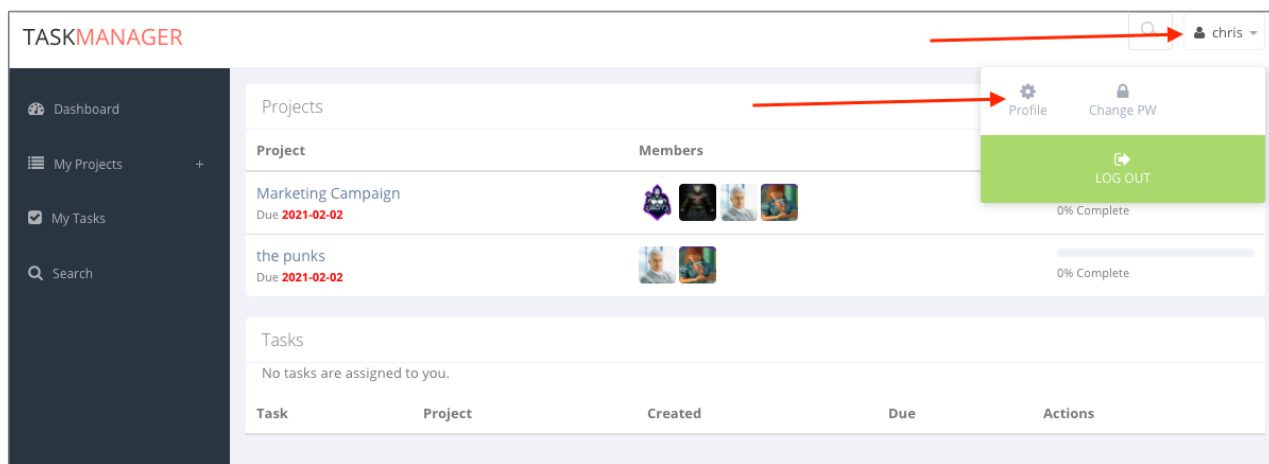
INSECURE DIRECT OBJECT REFERENCE

CRITICAL

DESCRIPTION

The application is exposing internal-object references to users. The object references could be a variety of things, such as files, directories, or numbers. These values can be modified in order to gain unauthorized access to resources.

EVIDENCE



Navigating to Profile page

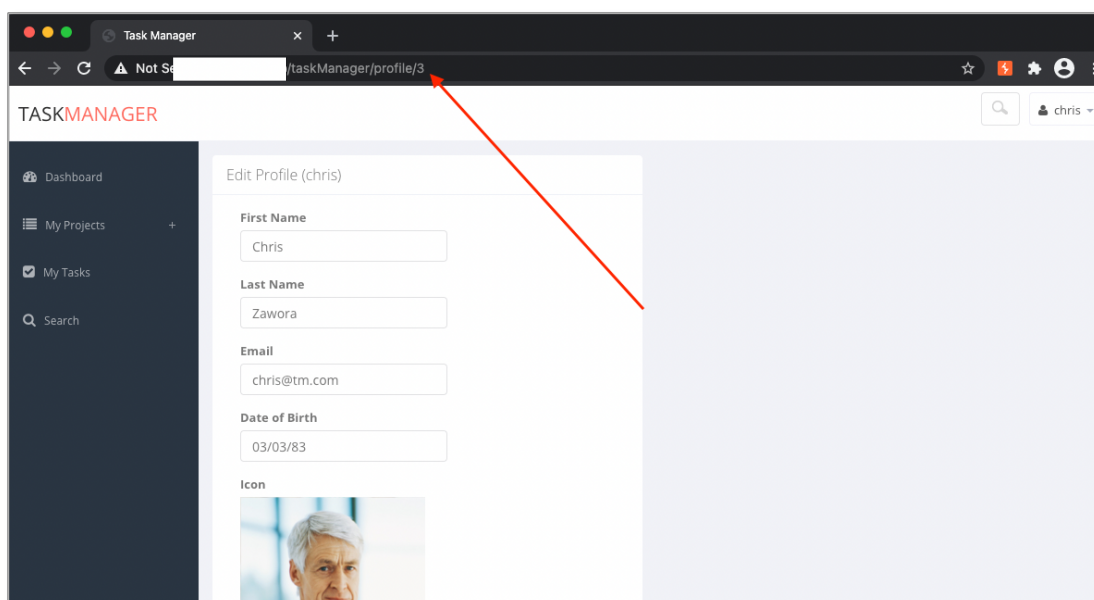


Figure 2: Viewing Direct Object Reference {profile ID} in URL

Endpoint	Parameter
url.url/taskManager/	profile

STEPS TO REPRODUCE

1. Login to any valid user account.
2. Check profile settings in drop-down menu.
3. View URL to see Insecure Direct Object Reference.
4. Manipulate value to see other users' information.

IMPACT

Difficulty	Likelihood
EASY	HIGH

An attacker can gain unauthorized access to resources by manipulating the reference values.

RECOMMENDATION

Access to resources should not be based solely on user-controlled values read directly from the request. Proper authorization should be implemented for user requests to ensure that users can only access resources they have permission to access.

REFERENCES

- OWASP - Broken Access Controls - https://owasp.org/Top10/A01_2021-Broken_Access_Control/
- Insecure Direct Object Reference Prevention - OWASP Cheat Sheet Series - https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

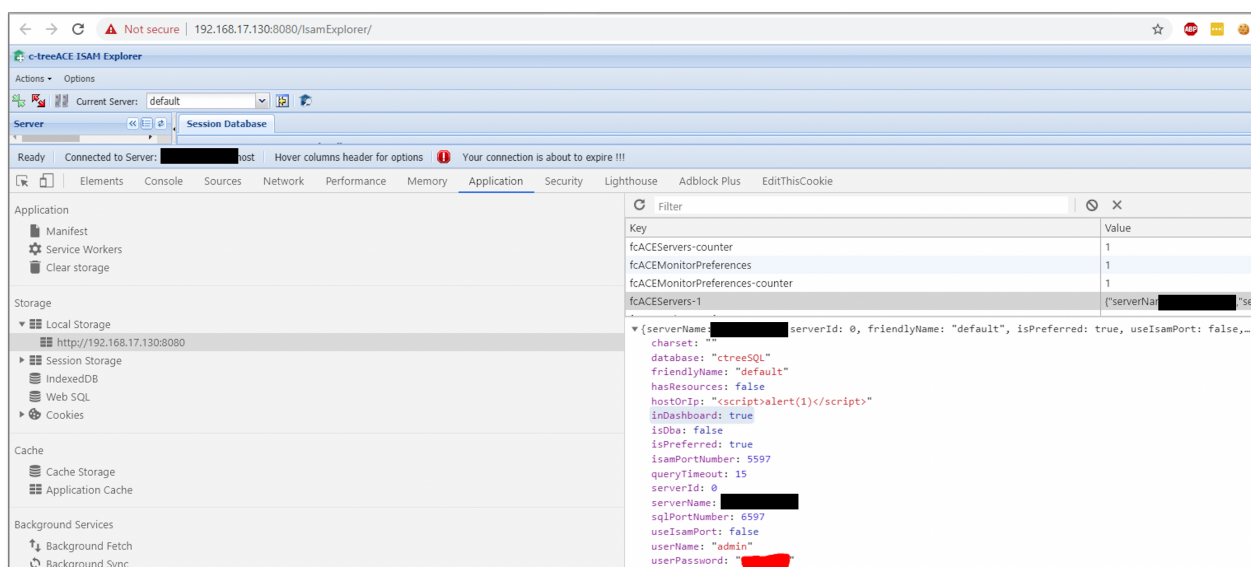
DEFAULT CREDENTIALS

HIGH

DESCRIPTION

The application or one of its platform components was operating with default credentials in place. Default credentials are well documented, and exploitable information is widely available not only in system documentation but in attacker web sites discussing known attack vectors.

EVIDENCE



View user-content fields populated with “default” and “admin” information.

Configuration	Default
Administrator UserName/Password	Admin:admin

STEPS TO REPRODUCE

1. Navigate to application component interface.
2. Use provided default credentials.
3. See successful login.

IMPACT

Difficulty	Likelihood
EASY	HIGH

An attacker can use default credentials to gain unauthorized access to the application or its platform. The identified credentials have elevated privileges and can cause a compromise of the entire application and its data.

RECOMMENDATION

Ensure that all default credentials are changed to a strong password. Implement a policy that identifies when the application, platform, or component is installed that part of the process is changing the password prior to deployment.

REFERENCES

- OWASP Top 10: 2021-A05-Security Misconfiguration - https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

STORED CROSS-SITE SCRIPTING (XSS)

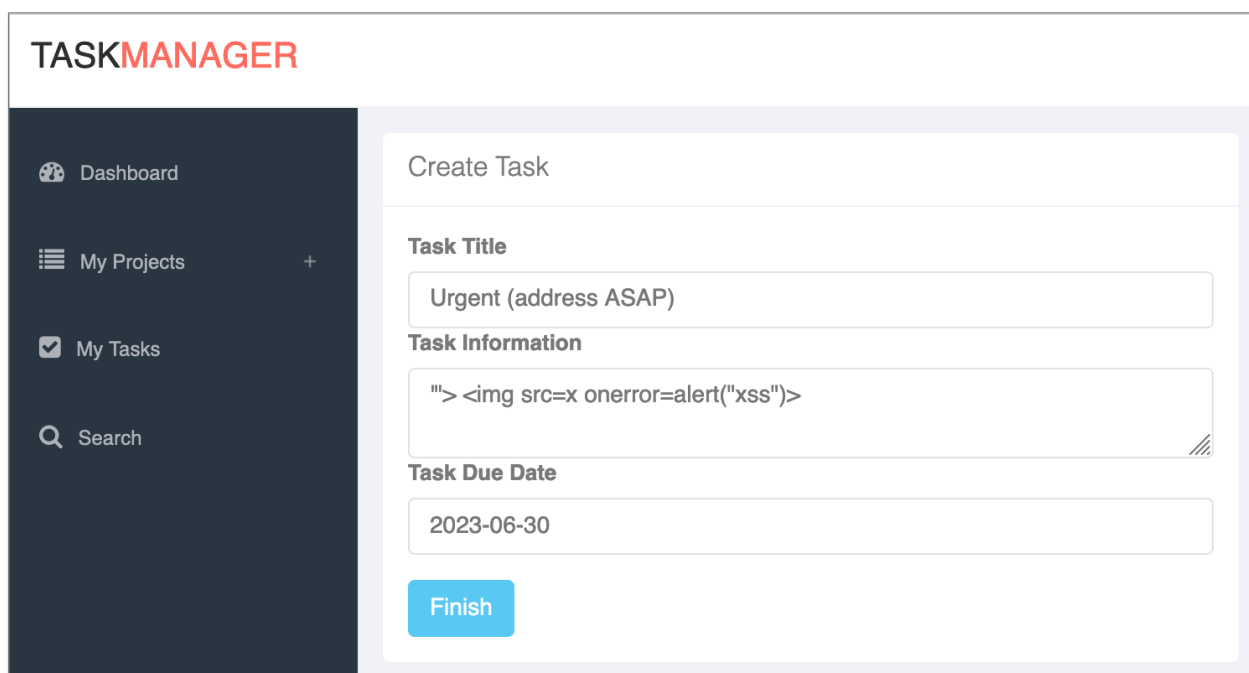
HIGH

DESCRIPTION

The application does not properly output encode data accepted from the user when it is returned in the application response. This allows for the injection of HTML and script characters making the application vulnerable to Cross-Site Scripting (XSS) attacks.

Stored XSS allows an attacker to store their attack, quite often in a database or other storage location, and it will be executed when a page is subsequently rendered. This malicious data persists and continues to execute until it is removed from the storage location or until proper output encoding is implemented.

EVIDENCE



The screenshot shows a web application interface titled "TASKMANAGER". On the left is a dark sidebar with navigation options: "Dashboard", "My Projects", "My Tasks", and "Search". The main content area is titled "Create Task" and contains the following fields:

- Task Title:** A text input field containing "Urgent (address ASAP)".
- Task Information:** A text area containing the XSS payload: `"> `.
- Task Due Date:** A date input field containing "2023-06-30".

At the bottom of the form is a blue button labeled "Finish".

Inserting XSS payload into "Task Information" field

TASKMANAGER

is one this one

Status: Active Created: 2023-06-14
Due: 2023-06-30

Participants: image

Tasks Add task

Title	Due Date	Status	Actions
<script>alert('tester')</script>	2023-06-21	In Progress	Edit Delete
Urgent (address ASAP)	2023-06-30	In Progress	Edit Delete

Viewing that Task with XSS payload has been added to Project List

TASKMANAGER

Urgent (address ASAP)

Created: 2023-06-15 Due: 2023-06-30

Participants: imag

Description: ">

Notes

Title	Text
-------	------

vtm.rdpt.dev

xss

OK

Pop-up activates indicating successful XSS injection

Endpoint	Parameter
url.url/taskManager/project_list	Task_Title, Task_Information

STEPS TO REPRODUCE

1. Log in as a user and navigate to “My Projects in dashboard.
2. Create a new task and add `'"> ` (or some other XSS payload) to each of the vulnerable form fields.
3. Save the form.
4. Visit the My-Projects page and select the created task.
5. View the pop-up indicating successful XSS injection.

IMPACT

Difficulty	Likelihood
EASY	HIGH

An attacker may set JavaScript in vulnerable form fields that will execute when users browse to a page where the script is stored or when an admin browses to their user page in the admin console. An attacker can further use this vulnerability steal user or admin credentials through crafting fake login pages and directing target victims to those pages.

RECOMMENDATION

The application should have an accept-list of characters that a user is allowed to input that is validated on the backend.

When possible, do not include user-supplied data in application responses. This would prevent an attacker from having the content rendered in the user's browser, thus mitigating the XSS vulnerability.

Output encoding should be implemented by the backend server so that any entities that an attacker would insert would be properly converted to their safe counterparts. The following are some examples of characters and their safe counterparts

- `< = <`;
- `> = >`;
- `& = &`;
- `/ = /`;
- `single quote = '`;
- `double quote = "`;

If possible, a security library should be used to sanitize these values or to utilize the output encoding features of the framework in use.

REFERENCE

- OWASP XSS Prevention Cheat Sheet – [https://cheatsheetseries.owasp.org/cheatsheets/Cross Site Scripting Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)
- OWASP XSS – <https://owasp.org/www-community/attacks/xss/>(<https://owasp.org/www-community/attacks/xss/>)
- OWASP-Top 10 2021:A03-Injection – https://owasp.org/Top10/A03_2021-Injection

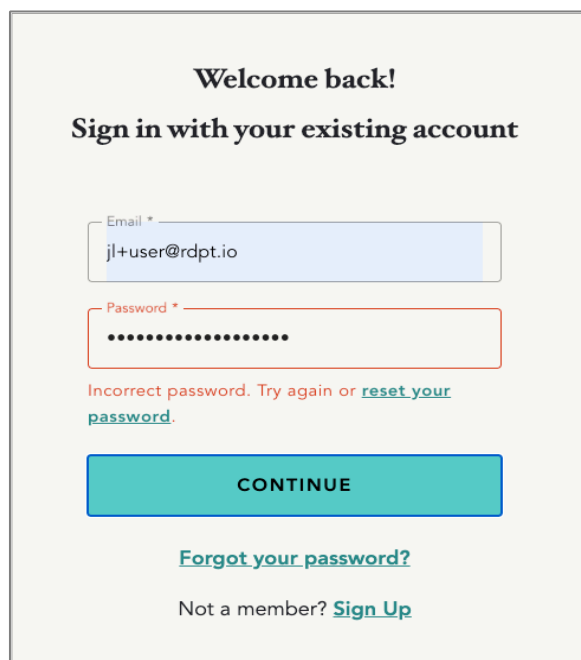
ACCOUNT ENUMERATION

MEDIUM

DESCRIPTION

The application responded to requests in a manner that allowed for the determination of valid accounts on the system. An attacker can use the discrepancies in responses between success and failure to determine valid accounts on the system. These responses can often be automated using iterating tools making attacks more effective.

EVIDENCE



Welcome back!

Sign in with your existing account

Email *

jl+user@rdpt.io

Password *

.....

Incorrect password. Try again or [reset your password](#).

CONTINUE

[Forgot your password?](#)

Not a member? [Sign Up](#)

Figure 4: Valid Account Returns Incorrect Password

Welcome back!

Sign in with your existing account

We couldn't find that email. If you're a [REDACTED] member, you'll need to [create a new \[REDACTED\] account](#) to access [REDACTED] clubs. This won't affect your [REDACTED] account.

Email *
jl@rdpt.io

Password *
•

CONTINUE

[Forgot your password?](#)

Not a member? [Sign Up](#)

Figure 5: Invalid Account Returns Different Response

STEPS TO REPRODUCE

1. Go to login page (<https://url.url/web/login>)
2. Sign in with `jl+user@rdpt.io` for the email and any string of characters for the password.
3. Submit.
4. View the error message stating "Incorrect password."
5. Go to login page (<https://url.url/web/login>).
6. Sign in with `jl@rdpt.io` for the email ID and any string of characters for the password
7. Submit.
8. View the error message stating, "We couldn't find that email."

IMPACT

Difficulty	Likelihood
MODERATE	HIGH

An attacker can use application responses to conduct attacks against valid users to gain unauthorized access.

RECOMMENDATION

Modify application responses to display generic messages that do not leak information about valid user accounts on the system. The same generic responses should be implemented in all locations where user account interaction happens. This includes login pages, password reset, and new user registration functionality.

REFERENCES

- OWASP Identification and Authentication Failures - https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

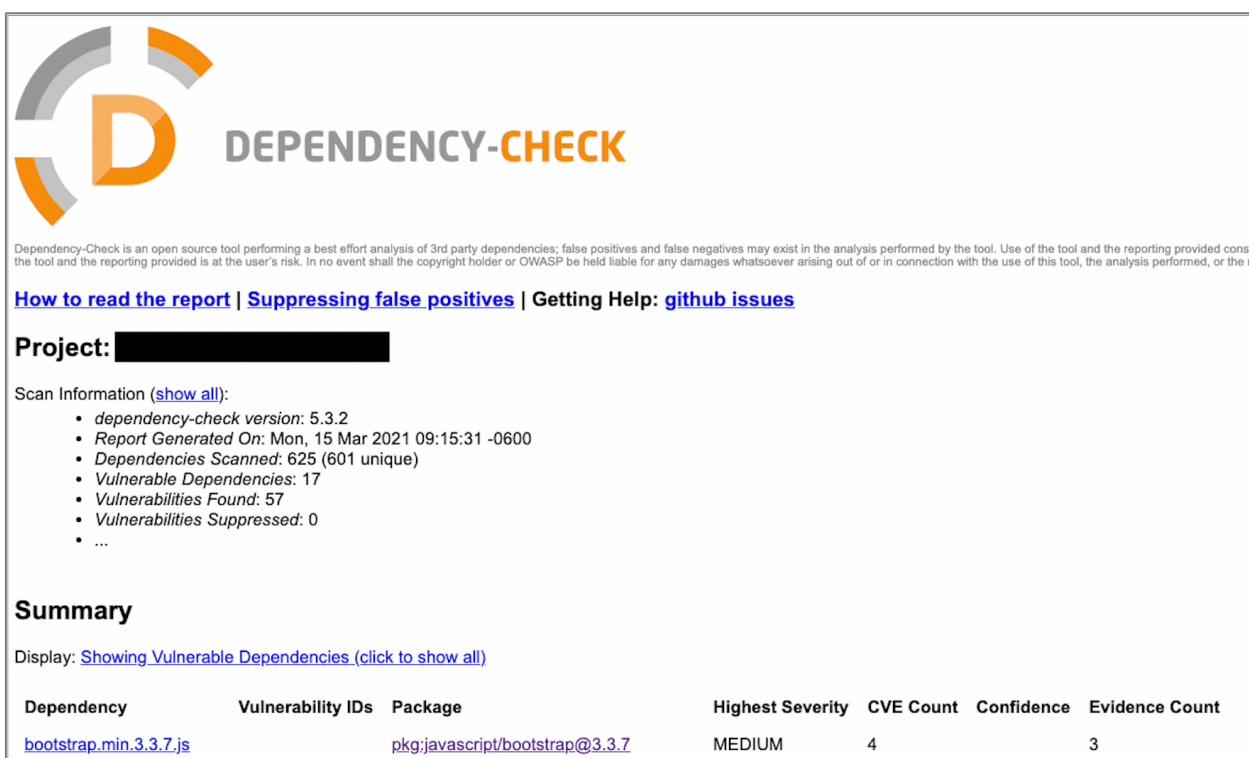
USING COMPONENTS WITH KNOWN VULNERABILITIES

LOW

DESCRIPTION

Outdated or weak components are in use by the application. These components may be part of a programming library or underlying platform. These weaknesses are commonly targeted by attackers because of the publicly available information on these vulnerabilities.

EVIDENCE



Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the reporting provided.

[How to read the report](#) | [Suppressing false positives](#) | [Getting Help: github issues](#)

Project: ██████████

Scan Information ([show all](#)):

- *dependency-check version:* 5.3.2
- *Report Generated On:* Mon, 15 Mar 2021 09:15:31 -0600
- *Dependencies Scanned:* 625 (601 unique)
- *Vulnerable Dependencies:* 17
- *Vulnerabilities Found:* 57
- *Vulnerabilities Suppressed:* 0
- ...

Summary

Display: [Showing Vulnerable Dependencies \(click to show all\)](#)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
bootstrap.min.3.3.7.js		pkg:javascript/bootstrap@3.3.7	MEDIUM	4		3

Vulnerable Components

bootstrap.min.3.3.7.js

CVE.1.2021.PleaseUpdate

CVE.2.2018.

IMPACT

Difficulty	Likelihood
MODERATE	HIGH

Weak and outdated components can be exploited by an attacker to gain unauthorized access to the application and its environment.

RECOMMENDATION

Update the component to a current, non-vulnerable version.

Implement security policies governing the use, tracking, and updating of platforms, libraries, and components. This will ensure that when future security updates are published, they can be worked into the application and environment.

REFERENCES

- OWASP Top 10 2021-A06:2021-Vulnerable and Outdated Components – https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

CONTENT SECURITY POLICY (CSP) NOT IMPLEMENTED

INFORMATIONAL

DESCRIPTION

The application does not send Content Security Policy (CSP) headers in server responses. CSP headers are secondary controls that can be implemented to help mitigate impact from XSS attacks like HTML, DOM, JSON, and JavaScript injection vulnerabilities.

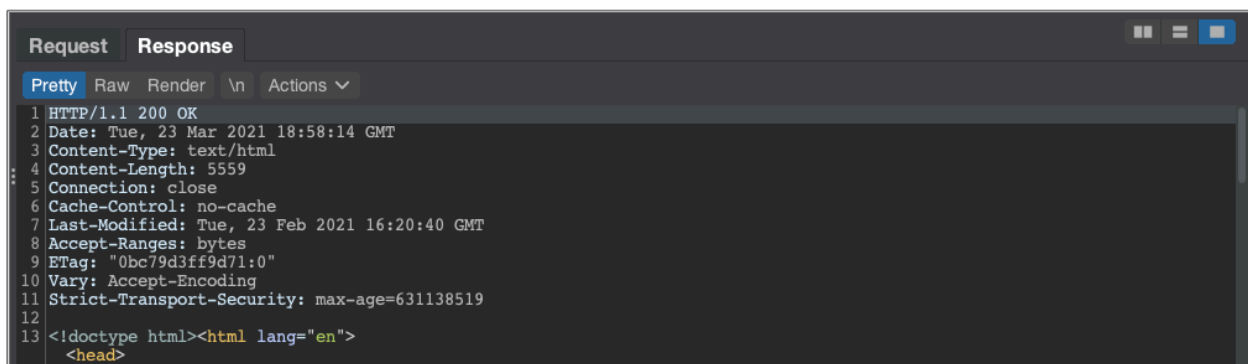
CSP can be enabled instructing the browser with a Content-Security-Policy directive in a response header:

```
Content-Security-Policy: script-src 'self';
```

or in a meta tag:

```
<meta http-equiv="Content-Security-Policy" content="script-src 'self';">
```

EVIDENCE



```
Request  Response
Pretty  Raw  Render  \n  Actions
1 HTTP/1.1 200 OK
2 Date: Tue, 23 Mar 2021 18:58:14 GMT
3 Content-Type: text/html
4 Content-Length: 5559
5 Connection: close
6 Cache-Control: no-cache
7 Last-Modified: Tue, 23 Feb 2021 16:20:40 GMT
8 Accept-Ranges: bytes
9 ETag: "0bc79d3ff9d71:0"
10 Vary: Accept-Encoding
11 Strict-Transport-Security: max-age=631138519
12
13 <!doctype html><html lang="en">
    <head>
```

Server Response without Content Security Policy Headers

Host

pentest1.app.url.url

IMPACT

There is no direct impact of not implementing CSP. However, if an application is vulnerable to a Cross-Site Scripting attack, CSP can prevent successful exploitation of that vulnerability. These headers provide an extra layer of security.

RECOMMENDATION

Enable CSP for the application by sending the `Content-Security-Policy` in HTTP response headers that instruct the browser to apply the specified policies. Apply a limited accept list and policies that are as strict as possible.

REFERENCES

- OWASP Content Security Policy - https://owasp.org/www-community/controls/Content_Security_Policy
- OWASP Content Security Policy Cheat Sheet - https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
- Content Security Policy Reference - <https://content-security-policy.com/>

APPENDIX: DEFINITIONS

Redpoint Security levels are determined by the evaluation of multiple factors surrounding the vulnerability and the environment where the vulnerability was identified. The rating of a vulnerability changes based on these factors. In general, the factors that make up the risk of an identified vulnerability include:

- Likelihood of vulnerability being identified
- Public availability of technical details related to vulnerability
- Attack tools
- Requisite attacker skill
- Potential and likelihood for successful exploitation
- Attack surface and users affected

SEVERITY

Severity levels as well as specific factors documented with levels in the finding are further defined in the tables below. While these definitions encompass common scenarios, a vulnerability's Severity level may be adjusted based on the specific circumstances in which it was encountered.

CRITICAL – The exposure may be exploited resulting in outcomes such as system compromise, authentication bypass, or unauthorized data access by users without privileges or existing user access. These findings are typically exploitable without authentication and should be addressed immediately.

HIGH – The exposure may be exploited resulting in outcomes such as system compromise, privilege escalation, or unauthorized data access by users with access to the system. These findings are exploitable by existing users and should be addressed as soon as possible.

MEDIUM – The exposure may be exploited resulting in outcomes such as system compromise, or privilege escalation where some user interaction is required for the attack to be successful. These findings should be remediated once all critical and high severity findings are remediated.

LOW – The exposure may provide information or access, which, while not exposing the system to current risk, may expose the system to risk in the future. These findings should be addressed but can be remediated over a longer timeline.

INFORMATIONAL - Controls that could be implemented to further enhance the security posture of the application, or, when based on business decision of company's or its customers' particular needs, a business could forego these controls. Redpoint recommends a wide range of preventative controls to help stop vulnerabilities before they can be exploited. Implementing these controls with a robust SDLC program and regular reviews can greatly increase an application's security posture.

DIFFICULTY OF EXPLOIT

The Difficulty of Exploit is the level of effort that is required for a threat agent to successfully exploit vulnerabilities identified in the target. This measurement takes into account compensating controls or other conditions that may augment the effort required to exploit the vulnerability.

EASY – The level of effort or sophistication required to exploit vulnerabilities present in the target is minimal or zero. This vulnerability may be exploited by an attacker with little or no intelligence or access. Compensating controls may not protect the target from attack. For example, a device that is exposed to the Internet that exhibits a vulnerability for which there is readily available exploits, tools, and/or techniques.

MODERATE – A moderate level of effort or sophistication is required to exploit the vulnerability. Compensating controls or conditions raise the level of effort required to attack the target. A threat must bypass compensating security controls to successfully exploit vulnerabilities. For example, a vulnerability that is present on a device but protected by a firewall requires a higher level of effort to exploit than if the vulnerable device was exposed directly to the Internet.

HARD – Significant effort is required to attack the device. Compensating controls significantly shield the target from attack by threats. Threats must circumvent security controls in order to exploit vulnerabilities present in the target. For example, the device is only accessible by an authenticated internal user with elevated access rights.

LIKELIHOOD

Redpoint evaluates the Likelihood of exploit based on criteria regarding whether application privileges are required for the exploit, user interaction is required to successfully use the attack, or, especially, whether there is significant exploit code maturity, i.e., attackers are commonly using attack pathways that align with the vulnerability.

HIGH – The vulnerability exists in a critical attack path which attackers commonly exploit. There is a minimal need for interaction with user or application admins or maintainers or an internal privileged role within the organization to successfully perform the exploit.

MEDIUM – The exploit code maturity is less developed than is the case with HIGH rated likelihood exploits, but there still is awareness of the attack vector. There may need to be some trivial user interaction or successful phishing to perform the exploit.

LOW – The vulnerability requires innovation or edge case testing to successfully perform an exploit. The attacker would have to be internal to the organization or work to compromise a hard human target to successfully exploit the vulnerability.

OVERALL APPLICATION RATING

Redpoint evaluates the reviewed application in the context of other assessments we have performed. This rating strives to answer the common questions we receive from clients wondering how their applications compare to others. As such, the overall ranking reflects our researchers' qualitative evaluation of an application's security posture relative to the security posture of other applications we have reviewed. The following is a rough guide regarding the meaning of these qualitative evaluations.

- **Critically Insecure** – The application has several critical and high-severity vulnerabilities that are accessible to unauthenticated attackers with well-established methods and vectors for exploits. Confidentiality, Integrity, and Availability of the application are all at risk, and remediation solutions require potentially difficult implementations. Redpoint recommends immediate remediation of specified findings to protect your customers and your organization's security reputation.
- **Insecure** – The application has high-severity vulnerabilities that threaten the Confidentiality, Integrity, or the Availability of the application. However, mitigations should not present too many implementation challenges. The high-risk vulnerabilities should be remediated expeditiously to protect customers and your organization's security reputation.
- **Moderately Insecure** – The application may have one or two high-risk vulnerabilities and several medium-risk findings that threaten Confidentiality, Integrity, or Availability. Mitigation should proceed according to priority of threat.

- **Average** – The application presents vulnerabilities that are commonly found in application testing, but none are especially severe. Mitigation should proceed according to priority of threat.
- **Moderately Secure** – Redpoint did not find any vulnerabilities that were concerning, and generally the application handled authentication, authorization, user input, data transfer and storage in a secure manner. The report includes a couple mitigation recommendations to improve the application's overall security posture or to align with security best practices.
- **Secure** – Redpoint evaluates the application as demonstrating a secure posture relative to Confidentiality, Integrity, and Availability. Findings are largely related to best security practices.

APPENDIX: METHODOLOGY

APPROACH

Redpoint utilizes a standard methodology for assessments that is comprised of three phases: information gathering, vulnerability identification, and reporting. Each phase feeds the next, but any activity in later phases may inform additional research and testing. The activities are cyclical to provide the analyst with working knowledge of the targeted properties for additional threat vectors.

INFORMATION GATHERING

Redpoint starts by reviewing application endpoints based on availability, application use-cases, developer documentation, and application source code. These endpoints are analyzed for use, potential parameters, additional attack surface, and possible threats. Applications are reviewed during this phase from multiple points of view, including an anonymous, un-authenticated user, an authenticated user, and an authenticated partner.

Redpoint analyzes available endpoints and source code during this phase for controls that affect security posture, including authentication and authorization controls, logging behavior, communication protocols, input handling, encryption settings, and other application behavior.

VULNERABILITY IDENTIFICATION

Redpoint uses the identified endpoints and controls of the identified assets to identify and explore possible security vulnerabilities across applications based on our expertise in assessing application flaws. Special attention will be paid to possible fraud and business logic flaws that could affect the Client, its partners, or its customers.

Redpoint utilizes industry-standard vulnerability lists for assessment purposes, including OWASP's Application Security Verification Standard, the OWASP Top 10 Security Risks, and the SANS CWE Top 25 Software Errors. These vulnerabilities are assessed across various security domains as they apply to the targeted application. Additional attack surfaces and weaknesses may be noted during this portion of the assessment for further research.

REPORTING

To finalize the assessment activity, Redpoint documents the assessment vulnerabilities, endpoints, and findings in a report that summarizes the results into actionable items for remediation by the Client. Each finding documents the steps required to reproduce identified vulnerabilities and includes recommendations for remediating or mitigating the threat.