NightVision Vulnerability Scan Report

2024-05-05

Created By: NightVision

Findings Overview:

Title	Severity	ID
SQL Injection - PostgreSQL	CRITICAL	2dc6ad2f-15d4-4798-ad18-bf7e5356ce53
Cross Site Scripting (Reflected)	HIGH	a7b88bd0-1511-4f27-b444-40ac8922fce5
Spring4Shell	CRITICAL	49decbc2-0ae9-452b-abc1-bdf4dfcdc6f9
Parameter Tampering	MEDIUM	e0ab10c6-c991-4f05-8c76-312cc2856397
Application Error Disclosure	LOW	e3727dc7-d674-491a-b37a-a3762be3763e
Application Error Disclosure	LOW	a31cce95-0dbc-4329-b1fd-d1f293987da5
Application Error Disclosure	LOW	9e416e37-cf63-4920-a48e-a786a25df97c
Missing HTTP Header - Access-Control-Allow-Origin	LOW	6cace365-811a-46a4-bbd1-81e95a893d31
Missing HTTP Header - Referrer-Policy	LOW	f3cda0be-5d04-4154-b2ca-ff6a11a0b9ce
Missing HTTP Header - Permissions-Policy	LOW	5ee0d878-b334-4d28-8944-17c7ec75612d
Missing HTTP Header - Cross-Origin-Resource-Policy	LOW	b3099aa5-8a71-4692-bada-9682131cdb63
Missing HTTP Header - Cross-Origin-Opener-Policy	LOW	0e9b65dc-ac9c-4267-b5a3-7afda325ee20
Missing HTTP Header - Cross-Origin-Embedder-Policy	LOW	392fa0f7-bf7c-4e7d-8315-3b03735ab45d
Missing HTTP Header - Content-Security-Policy	LOW	6d2ada71-6ff7-4fd8-9ccc-ff7701b24bab

Findings Details:

1. SQL Injection - PostgreSQL

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/search` URL path is vulnerable to **SQL Injection - PostgreSQL** via a `POST` request. The application declared the `/search` endpoint in the file `src/main/java/hawk/controller/SearchController.java` on **Line 36** .

Description : The vulnerability found in the application is a SQL Injection vulnerability, specifically targeting the PostgreSQL database. The payload used in the HTTP Request was a single quote ('). The parameter name affected by this vulnerability is `searchText` , and the HTTP method used is `POST` .

The evidence of this vulnerability is the error message returned by the application, which indicates that a PostgreSQL database error occurred. By manipulating the `searchText` parameter and injecting a single quote as the payload, the application's response confirms the existence of the vulnerability. This proof-of-concept attack demonstrates that an attacker can manipulate the input to execute arbitrary SQL queries on the database, potentially leading to unauthorized access or data manipulation.

The impact of this vulnerability is significant, as it allows an attacker to bypass the intended application logic and directly interact with the database. With the ability to execute arbitrary SQL queries, an attacker could extract sensitive information, modify or delete data, or even gain unauthorized access to other parts of the system. It is crucial to address this vulnerability promptly by implementing proper input validation and parameterized queries to prevent SQL injection attacks.

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/119

CRITICAL

2. Cross Site Scripting (Reflected)

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/search` URL path is vulnerable to **Cross Site Scripting (Reflected)** via a `POST` request. The application declared the `/search` endpoint in the file `src/main/java/hawk/controller/SearchController.java` on **Line 36** .

Description : The application security vulnerability that was found in the application is a Cross-Site Scripting (XSS) vulnerability. The payload `"><script>alert(1);</script>` was supplied in the HTTP Request parameter named `searchText` using the HTTP method `POST` . The vulnerability exists in the `/search` URL path on the application located at `127.0.0.1:9000` .

In this scenario, the value of the `searchText` parameter is being reflected in the application's response without proper sanitization or encoding. The payload `"><script>alert(1);</script>` demonstrates a proof-of-concept attack by injecting a malicious script into the application's response. This allows an attacker to execute arbitrary JavaScript code in the victim's browser, potentially leading to various malicious actions such as stealing cookies, performing phishing attacks, or gaining unauthorized access.

To mitigate this vulnerability, it is essential to implement proper input validation and output encoding to prevent the injection of malicious scripts. By sanitizing user input and encoding output, the application can ensure that any user-supplied data is treated as plain text and not executed as code. Regular security testing and code reviews should be conducted to identify and address such vulnerabilities proactively.

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/35

HIGH

3. Spring4Shell

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/search` URL path is vulnerable to **Spring4Shell** via a `POST` request. The application declared the `/search` endpoint in the file `src/main/java/hawk/controller/SearchController.java` on **Line 36** .

Description : The vulnerability found in the application is a **Spring4Shell** vulnerability. This vulnerability allows an attacker to execute arbitrary commands on the server by exploiting a command injection vulnerability in the Spring framework. The payload used in this attack is `class.module.classLoader.DefaultAssertionStatus=nonsense` .

The HTTP request was sent as a **POST** request to the `/search` URL path on the **127.0.0.1:9000** location. The evidence of the vulnerability is an HTTP response with a status code of 400.

In this proof-of-concept attack, the payload is injected into the application's input, which is then processed by the Spring framework. The vulnerability allows the attacker to execute arbitrary commands on the server, potentially gaining unauthorized access, manipulating data, or causing system disruptions. It is crucial to address this vulnerability promptly by applying security patches or updates provided by the Spring framework to prevent further exploitation and potential damage to the application and its underlying infrastructure.

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/137

CRITICAL

4. Parameter Tampering

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/search` URL path is vulnerable to **Parameter Tampering** via a `POST` request. The application declared the `/search` endpoint in the file `src/main/java/hawk/controller/SearchController.java` on **Line 36** .

Description : The vulnerability in question is a Parameter Tampering vulnerability, which occurs when an attacker is able to modify the parameters of an HTTP request. In this case, the payload used is represented as `sa0000` , and it was supplied in the `searchText` parameter of a POST request to the `/search` URL path on the application located at `127.0.0.1:9000` .

The impact of this vulnerability lies in the ability of an attacker to manipulate the application's parameters, potentially leading to unauthorized access or malicious actions. By tampering with the `searchText` parameter, an attacker could manipulate the application's behavior, bypass security controls, or gain access to sensitive information. The evidence of this vulnerability's existence is indicated by the provided text explanation, specifically the mention of the `javax.servlet.http.HttpServlet.service` method, which suggests that the application's servlet is vulnerable to parameter tampering. This proof-of-concept attack demonstrates the need for proper input validation and parameter sanitization to prevent such vulnerabilities and protect the application's integrity and security.

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/95

MEDIUM

5. Application Error Disclosure

🚨 **Exploitable Vulnerability Found** 🚨

Description : ## Summary

This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.

Solution

Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/7

LOW

6. Application Error Disclosure

🚨 **Exploitable Vulnerability Found** 🚨

Description : ## Summary

This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.

Solution

Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/7

LOW

7. Application Error Disclosure

🚨 **Exploitable Vulnerability Found** 🚨

Description : ## Summary

This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.

Solution

Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/7

LOW

8. Missing HTTP Header - Access-Control-Allow-Origin

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/` URL path is vulnerable to **Missing HTTP Header - Access-Control-Allow-Origin** via a `GET` request. The application declared the `/` endpoint in the file `src/main/java/hawk/controller/IndexController.java` on **Line 10** .

Description : The application is missing the `Access-Control-Allow-Origin` security header. This header is used to specify which external domains are allowed to access the website's resources. By not including this header, an attacker could bypass the browser's cross-domain policy and perform malicious actions, such as stealing cookies, performing phishing attacks, or other malicious activities.

To demonstrate this security misconfiguration, send an HTTP request to `None` with the following `curl` command: `curl -I None` . If the HTTP response does not include the `Access-Control-Allow-Origin` header in the response, then it does not have this layer of protection provided by the HTTP header.

References : - https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html#access-control-allow-origin - https://developer.mozilla.org/en-us/docs/web/http/headers/access-control-allow-origin

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/183

LOW

9. Missing HTTP Header - Referrer-Policy

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/` URL path is vulnerable to **Missing HTTP Header - Referrer-Policy** via a `GET` request. The application declared the `/` endpoint in the file `src/main/java/hawk/controller/IndexController.java` on **Line 10** .

Description : The site is missing the `Referrer-Policy` HTTP security header. This HTTP header controls how much referrer information (sent via the `Referrer` header) should be included with requests. If this is not included, and the user is leveraging an older browser, then the `Referrer` information can can contain the absolute or partial URL from which the resource was requested, potentially leading to information leakages offsite via the URLs in the `Referrer` header.

Note : Today, the default behavior in modern browsers is to no longer send all *referrer* information (origin, path, and query string) to the same site but to only send the origin to other sites. Therefore, the `Referrer-Policy` header is not as important as it used to be. Implementing this control could protect certain information leakage scenarios, but in most cases is not a critical security control.

References : - https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html#referrer-policy - https://developer.mozilla.org/en-us/docs/web/http/headers/referrer-policy

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/189

LOW

10. Missing HTTP Header - Permissions-Policy

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/` URL path is vulnerable to **Missing HTTP Header - Permissions-Policy** via a `GET` request. The application declared the `/` endpoint in the file `src/main/java/hawk/controller/IndexController.java` on **Line 10** .

Description : The application does not include the `Permissions-Policy` HTTP security header in the response. This could allow a successful XSS attack to leverage sensitive browser security features, like accessing a client's geolocation, microphone, or camera.

To demonstrate this security misconfiguration, send an HTTP request to `None` with the following `curl` command: `curl -I None` . If the HTTP response does not include the `Permissions-Policy` header, then it does not have this layer of protection provided by the HTTP header.

Consider setting the Permissions policy to disable geolocation, camera, and microphone for all domains, unless the site requires these features. This will help protect the site from XSS vulnerabilities that could be used to access sensitive information. For example: `Permissions-Policy: geolocation=(), camera=(), microphone=()` .

References : - https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html#permissions-policy-formerly-feature-policy - https://developer.mozilla.org/en-us/docs/web/http/headers/permissions-policy

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/188

LOW

11. Missing HTTP Header - Cross-Origin-Resource-Policy

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/` URL path is vulnerable to **Missing HTTP Header - Cross-Origin-Resource-Policy** via a `GET` request. The application declared the `/` endpoint in the file `src/main/java/hawk/controller/IndexController.java` on **Line 10** .

Description : This site is missing the `Cross-Origin-Resource-Policy` HTTP security header. The `Cross-Origin-Resource-Policy` (CORP) header allows you to control the set of origins that are empowered to include a resource. It is a robust defense against attacks like Spectre, as it allows browsers to block a given response before it enters an attacker's process.

To demonstrate this security misconfiguration, send an HTTP request to `None` with the following `curl` command: `curl -I None` . If the HTTP response does not include the `Cross-Origin-Resource-Policy` header, then it does not have this layer of protection provided by the HTTP header.

Consider setting the `Cross-Origin-Resource-Policy` header to `same-site` to prevent cross-domain attacks.

References : - https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html#cross-origin-resource-policy-corp - https://developer.mozilla.org/en-us/docs/web/http/headers/cross-origin-resource-policy

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/187

LOW

12. Missing HTTP Header - Cross-Origin-Opener-Policy

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/` URL path is vulnerable to **Missing HTTP Header - Cross-Origin-Opener-Policy** via a `GET` request. The application declared the `/` endpoint in the file `src/main/java/hawk/controller/IndexController.java` on **Line 10** .

Description : The `Cross-Origin-Opener-Policy` HTTP header is missing from the site's HTTP response. The `Cross-Origin-Opener-Policy` header is used to ensure that the browser's cross-domain policy is enforced.

COOP will process-isolate your document and potential attackers can't access your global object if they were to open it in a popup, preventing a set of cross-origin attacks dubbed XS-Leaks, which includes Spectre, Meltdown, and Rowhammer.

Consider setting the `Cross-Origin-Opener-Policy` header to `same-origin` to prevent cross-domain attacks.

References : - https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html#cross-origin-opener-policy-coop - https://developer.mozilla.org/en-us/docs/web/http/headers/cross-origin-opener-policy - https://xsleaks.dev/

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/186

LOW

13. Missing HTTP Header - Cross-Origin-Embedder-Policy

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/` URL path is vulnerable to **Missing HTTP Header - Cross-Origin-Embedder-Policy** via a `GET` request. The application declared the `/` endpoint in the file `src/main/java/hawk/controller/IndexController.java` on **Line 10** .

Description : This site is missing the `Cross-Origin-Embedder-Policy` HTTP security header. The Cross-Origin-Embedder-Policy (COEP) header is used to specify a policy for which cross-origin resources are allowed to be embedded within a document. The lack of this header could allow an attacker to bypass the browser's cross-domain policy and gain access to sensitive information.

To demonstrate this security misconfiguration, send an HTTP request to `None` with the following `curl` command: `curl -I None` . If the HTTP response does not include the `Cross-Origin-Embedder-Policy` header in the response, then it does not have this layer of protection provided by the HTTP header.

References : - https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html#cross-origin-embedder-policy-coep - https://developer.mozilla.org/en-us/docs/web/http/headers/cross-origin-embedder-policy

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/185

LOW

14. Missing HTTP Header - Content-Security-Policy

🚨 **Exploitable Vulnerability Found** 🚨

ℹ️ The `/` URL path is vulnerable to **Missing HTTP Header - Content-Security-Policy** via a `GET` request. The application declared the `/` endpoint in the file `src/main/java/hawk/controller/IndexController.java` on **Line 10** .

Description : The application does not include the `Content-Security-Policy` HTTP security header in the response. The lack of this header could allow an attacker to inject arbitrary JavaScript into the page because browsers allow the loading of any resource, including scripts and stylesheets, when the `Content-Security-Policy` header is missing.

To demonstrate this security issue, send an HTTP request to `None` with the following `curl` command: `curl -I None` . The response does not include the `Content-Security-Policy` header.

If an attacker can control a script that is executed in the victim's browser, then the attacker could use this access to steal cookies, perform phishing attacks, or perform other malicious actions. This could have a serious impact on the security of the application and its users.

References : - https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html#content-security-policy-csp - https://developer.mozilla.org/en-us/docs/web/http/headers/content-security-policy

🔍 For more information see the issue on **NightVision** here: https://app.nightvision.net/scans/8ae4ffd7-1ee1-421e-8260-4543ce899f35/findings/184

LOW