SYNACKTIV

SECURITY ADVISORY

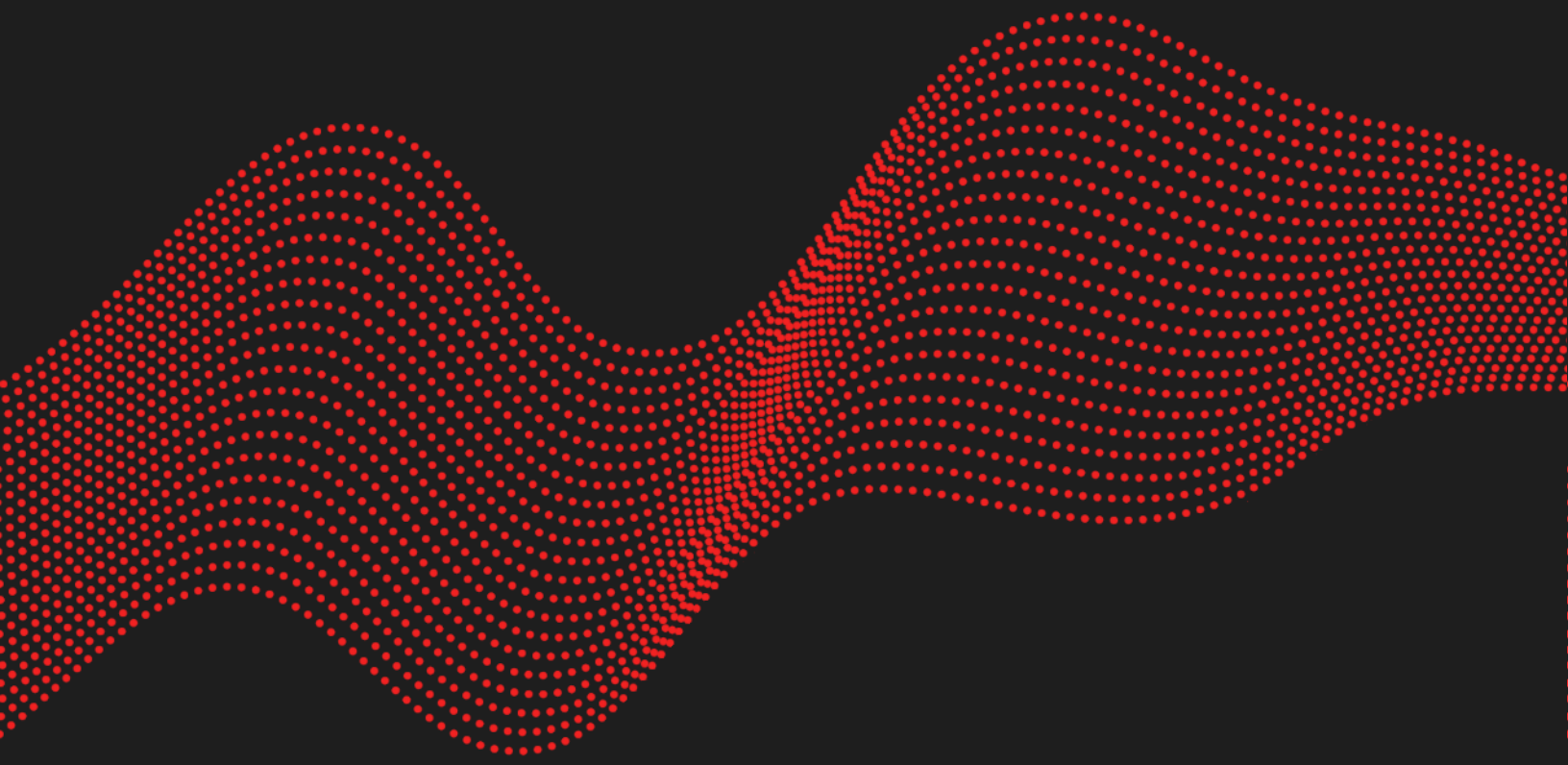
Improper Privilege Management in Grails Spring Security Core <= 5.1.0

CVE-2022-41923

2023.03.21

BENJAMIN SEPE

ADRIEN PETER



Vulnerability description

Presentation of Grails Spring Security Core

The open source Grails® Framework supported by the Grails Foundation is used to build web applications with the Groovy programming language. The core framework is very extensible and there are numerous plugins available that provide easy integration of add-on features.[1]

The Grails Spring Security plugin simplifies the integration of Spring Security into Grails Framework applications. The plugin provides sensible defaults with many configuration options for customization. Nearly everything is configurable or replaceable in the plugin and in Spring Security itself, which makes extensive use of interfaces.[2]

Issue

Synacktiv discovered an improper parsing of resource URIs in Grails Spring Security Core's authorization system that allows an authenticated user to bypass some applications' authorization requirements.

Mitigation

Upgrade to one of the following plugin versions:

- 3.3.2
- 4.0.5
- 5.1.1

Affected versions

The following plugin versions are affected:

- 1.x
- 2.x
- >=3.0.0 <3.3.2
- >=4.0.0 <4.0.5
- >=5.0.0 <5.1.1

1 https://grails.org/
2 https://grails.github.io/grails-spring-security-core/5.0.0-RC1/index.html

Timeline

Date	Description
2022.11	Advisory sent to Grails Foundation
2022.11.10	Patches deployed on GitHub.com
2022.11.22	Advisory published on the Grails Framework blog: https://grails.org/blog/2022-11-22-ss-plugin-auth-cve.html
2023.03.21	Public release

Special thanks

Synacktiv would like to thank the Grails Foundation team for their quick response to the vulnerability and their feedback for this advisory.

Technical description

Description

The Grails Spring Security Core plugin allows the application's developer to define role requirements for each web application controller. In the following example, **ROLE_USER** and **ROLE_ADMIN** have been created, corresponding to two different level of privileges. A **SecureController** has been defined, containing a **/secure** path routing to the default **index** action and requires the **ROLE_USER** role, and **/secure/admin** that requires the **ROLE_ADMIN** role:

```
$ cat SecureController.groovy
@Secured('ROLE_USER')
class SecureController {
    def index() {
        render 'User access only'
    }
    @Secured('ROLE_ADMIN')
    def admin() {
        render 'Admin access only'
    }
}
[...]
```

When authenticated with the **ROLE_USER** role, access to the **/secure/admin** page is indeed denied:

```
$ curl -i http://localhost:8080/secure/admin -H 'Cookie: JSESSIONID=[...]'
HTTP/1.1 403
<div class="errors">Sorry, you're not authorized to view this page.</div>
[...]
```

However, inserting a semicolon (**;**) at a specific position in the path allows accessing the restricted page, bypassing the role requirement.

```
$ curl -i 'http://localhost:8080/secure;/admin' -H 'Cookie: JSESSIONID=[...]'
HTTP/1.1 200
[...]
Admin access only
```

The presence of a semicolon (**;**) character before the **/admin** endpoint allowed to bypass the endpoint's authorization requirements, and instead inherit the **ROLE_USER** requirement for the **/secure** endpoint.

Impact

An attacker with a valid, low-privileged account on a web application using the plugin could use this vulnerability to bypass its privilege requirements.

Indeed, exploiting this vulnerability grants the user access to all methods of a given controller, provided that they are authorized to access the controller's default action endpoint (which by default is the `index` action).

Detailed analysis

When used with some containers, the request URL received by the Grails Framework may include a semicolon to pass the `jsessionid` parameter. For example:

```
http://mywebsite.com/context/foo;jsessionid=A3294FBE42?a=b
```

This notation is called Matrix Parameters. A Matrix URI is written using semicolons to delimitate the path resources.

```
/context/foo;f1=WWW/bar;s1=1;s2=2/index.html;i1=1;i2=2
```

The top-level resource may contain parameters not applying to the sub-level resources.

To handle the case when a container passes the `jsessionid` parameter, the Grails Spring Security Core plugin strips everything after the first semicolon when extracting the context path of the request:

```
File: plugin/src/main/groovy/grails/plugin/springsecurity/web/access/intercept/
AbstractFilterInvocationDefinition.groovy@77132c1
176:     protected String calculateUri(HttpServletRequest request) {
177:           String url = request.requestURI.substring(request.contextPath.length())
178:           int semicolonIndex = url.indexOf(';')
179:           semicolonIndex == -1 ? url : url.substring(0, semicolonIndex)
180:     }
```

The `url` variable is then used to decide whether the user has the rights to access it or not.

Consequently, when an attacker includes a semicolon in the request's path, the controller authorization only applies to the first part of the URI instead of the whole path.

Thus, as demonstrated in the previous section, a request to the `/secure;/admin` path is parsed by the plugin as a request to `/secure`, which is in this case is configured to be accessible by an unprivileged user. The Grails Framework then serves the privileged `/secure;/admin` page.

Patch analysis

The following patches were committed to fix the issue:

- On 11/10/2022, included in release 5.1.1:
  https://github.com/grails/grails-spring-security-core/pull/806/commits/bf69e3676948a78e82df570b958aaba5cbcfa3c2
- On 11/13/2022, included in release 4.0.5:
  https://github.com/grails/grails-spring-security-core/pull/824/commits/e0795e1e197f28559e22cd42e3b6372ca7753cc3
- On 11/17/2022, included in release 3.3.2:
  https://github.com/grails/grails-spring-security-core/pull/837/commits/cf87f21a11b046ee5e734f8a4b1932fa4c2225ef

The code now uses the **`getRequestUri`** method of the **`urlPathHelper`** class from the Spring Framework library:

```
File: plugin/src/main/groovy/grails/plugin/springsecurity/web/access/intercept/
AbstractFilterInvocationDefinition.groovy@890ebac
18: import org.springframework.web.util.UrlPathHelper
[...]
96:       protected String determineUrl(FilterInvocation filterInvocation) {
97:          final HttpServletRequest request = filterInvocation.request
98:             lowercaseAndStripQuerystring
stripContextPath(urlPathHelper.getRequestUri(request), request)
99:       }
```

This method states in its documentation that “some containers [...] include ";" strings like ";jsessionid" in the URI. This method cuts off such incorrect appendices.”

Trying to reproduce the previous example on version 5.1.1 of the plugin demonstrates that the vulnerability is indeed fixed:

```
$ curl -i 'http://localhost:8080/secure;/admin' -H 'Cookie: JSESSIONID=[...]'
HTTP/1.1 403
[...]
<div class="errors">Sorry, you're not authorized to view this page.</div>
```

SYNACKTIV

01 45 79 74 75

contact@synacktiv.com

5 boulevard Montmartre

75002 – PARIS

www.synacktiv.com