

Spring Security Advisories



cve-2026-22732: Under Some Conditions Spring Security HTTP Headers Are not Written

CRITICAL | MARCH 19, 2026 | CVE-2026-22732

Description

When applications specify HTTP response headers for servlet applications using Spring Security, there is the possibility that the HTTP Headers will not be written. This can open up applications to various attacks including exposing sensitive data via caching mechanisms.

Affected Spring Products and Versions

Spring Security Servlet applications using lazy (default) writing of HTTP Headers:

- 5.7.0 - 5.7.21
- 5.8.0 - 5.8.23
- 6.3.0 - 6.3.14
- 6.4.0 - 6.4.14
- 6.5.0 - 6.5.8
- 7.0.0 - 7.0.3
- Older, unsupported versions may also be affected

Mitigation

Reporting a vulnerability

To report a security vulnerability for a project within the Spring portfolio, see the [Security Policy](#)

Affected version(s)	Fix version	Availability
5.7.21	5.7.22	Enterprise Support Only
5.8.23	5.8.24	Enterprise Support Only
6.3.14	6.3.15	Enterprise Support Only
6.4.14	6.4.15	Enterprise Support Only
6.5.8	6.5.9	OSS
7.0.3	7.0.4	OSS

Workarounds

Applications can work around the issue by setting the

`HeaderWriterFilter.shouldWriteHeadersEagerly` property to `true`. However, it should be noted that this will change the

application behavior. For example, with

`shouldWriteHeadersEagerly=false` (default), then if any cache related headers are set by the application, then Spring Security will not write any cache related headers. However, with

`shouldWriteHeadersEagerly=true`, application specific headers that are written will only override that specific header with no way to remove any HTTP Headers that were explicitly written.

Java Based Configuration Workaround

If you are comfortable changing the application behavior as described above, then you can set the

`shouldWriteHeadersEagerly` using an `ObjectPostProcessor`:

```
@Bean
SecurityFilterChain springSecurity(HttpSecurity http) thr
    // @formatter:off
    http
```

```
// ...
    .addObjectPostProcessor(new ObjectPostProcess
        @Override
        public HeaderWriterFilter postProcess(HeaderWriterFilter filter) {
            filter.setShouldWriteHeadersEagerly(true);
            return filter;
        }
    });
return http.build();
// @formatter:on
}
```

XML Based Configuration Workaround

If you are comfortable changing the application behavior as described above, then you can set the

`shouldWriteHeadersEagerly` using a custom

`BeanPostProcessor`:

To start define a custom `BeanPostProcessor` that sets

`shouldWriteHeadersEagerly`:

```
public class EagerHeadersBeanPostProcessor implements BeanPostProcessor {

    @Override
    public Object postProcessAfterInitialization(Object bean, String beanName) throws BeansException {
        if (bean instanceof HeaderWriterFilter headerWriterFilter) {
            headerWriterFilter.setShouldWriteHeadersEagerly(true);
        }
        return bean;
    }
}
```

Then ensure to add the `BeanPostProcessor` as a Bean:

```
<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodProcessor" />
```

Credit

The issue was identified and responsibly reported by Wyfrel.

References

- <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N&version=3.1>

Get ahead

VMware offers training and certification to turbocharge your progress.

[Learn more](#)

Get support

Tanzu Spring offers support and binaries for OpenJDK™, Spring, and Apache Tomcat® in one simple subscription.

[Learn more](#)

Upcoming events

Check out all the upcoming events in the Spring community.

[View all](#)



Why Spring

- Generative AI
- Microservices
- Reactive
- Event Driven
- Cloud

Learn

- Quickstart
- Guides
- Courses
- Get Certified

Projects

Resources

- Blog
- Release Calendar

Community

- Events
- Authors

Enterprise

- Long-term Support

Serverless
Batch

Release
Highlights
Security
Advisories

Governance and
Compliance
Modern App
Development

Thank You

Get the Spring newsletter

Stay connected with the Spring newsletter

SUBSCRIBE

